School of Information Systems
Curtin Business School

A Defeasible Logic Programming-based Framework to Support Argumentation in Semantic Web applications

Naeem Khalid Janjua

This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

February 2013

DECLARATION

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Naeem Khalid Janjua

Signature: . . . .

Date: . . . .

Table of Contents

List of Figures

List of Tables

Preface

Using ontologies, the Semantic Web provides structure and meaning to the vast amount of available information on the World Wide Web (WWW) and enables machines and/or computers to utilize, process, reason and discover knowledge from it. The logic layer of the Semantic Web stack provides a set of logic-based rule languages to perform automated reasoning over such information, produce results and assist the decision maker in the decision making process. Initial efforts in the literature for reasoning in Semantic Web applications have focused on the use of monotonic logic. However such efforts lack the capability to represent and reason when the underlying information is incomplete and/or contradictory. To overcome this problem, defeasible reasoning-based Semantic Web applications have been proposed that are capable of representing and reasoning over incomplete and/or contradictory information after defining the priorities between them. However their drawback is that they can only represent and reason over information coming from a single source. In scenarios where the decision maker is interested in considering information from multiple sources (such as information from collaborating enterprises or the feedback from customers) and where such information is incomplete and/or contradictory, current Semantic Web-based approaches do not provide any solution to represent, reason, resolve conflicts and integrate it to assist in the decision making process. This is in contrast to the approaches proposed in the literature in Artificial intelligence, where argumentation formalisms have been used to reason over contradictory information and produce a justifiable, tractable conclusion.

Therefore, to overcome such limitations in the Semantic Web discussed above, in this thesis a generic defeasible logic programming-based framework is proposed to support argumentation in Semantic Web applications (GF@SWA). GF@SWA enables Semantic Web applications to represent both structured and unstructured information and/or translate the existing information into a defeasible logic programming (DeLP) format, perform hybrid reasoning for arguments construction, identify and resolve conflicts among arguments, integrate them and produce their graphical representation in the form of reasoning chains. The GF@SWA also provides a solution to integrate the reasoning chains produced by different Semantic Web applications and assists the decision maker in the decision making process. For validation and evaluation of GF@SWA, three Semantic Web applications are developed using GF@SWA to provide decision support to an enterprise to achieve business intelligence. The functionality and features of each Semantic Web application are validated and evaluated to highlight the effectiveness of GF@SWA in addressing the decision making requirements of an enterprise.

Acknowledgements

First of all, praise and thanks to Almighty Allah for his uncountable blessings bestowed upon me and for enabling me to complete my Doctoral dissertation under the supervision of Dr. Omar Khadeer Hussain and Dr. Farookh Khadeer Hussain.

My greatest gratitude goes to my family. I thank my mother, Gul-e-Shareen, who taught me the principles of morality, determination and perseverance in life. I thank my father, Khalid Hussain Janjua, who always inspired me, encouraged me to embark on higher studies, despite the enormous financial difficulties and supported my travel adventure to Australia. I thank my sisters and brothers especially Faheem and Waseem for being my best friends and for being a constant source of support for me throughout my studies. I thank my wife, Naila and my son, Haris who have been a great strength throughout my PhD. I also thank the members of my extended family in Pakistan for their emotional support.

I would like to thank my supervisors, Dr. Omar Khadeer Hussain and Dr. Farookh Khadeer Hussain, for their continued support, excellent guidance and encouragement throughout my research. This thesis is as much their effort as it is mine. I thank Dr. Farookh and Professor Tharam Dillon for their help and guidance in assisting me to enrol at Curtin University and supporting me during the course of my studies.

I would like to express my appreciation to Dr. Alejandro J. Garcia and Mr. Sebastian Gottifredi from the "Artificial Intelligence Research and Development Laboratory (LIDIA)", Argentina, for providing the DeLP Server for our research. I also want to thank Dr. Iyad Rehwan from the "Masdar Institute" for answering my emails and providing valuable feedback for my research.

I would like to thank my friends, as they deserve a big 'thank you' for sharing with me the simple joys along the way. I thank Jamshaid, Atif, Zia, Adil, Raza, Bambang, Mohammad, Ali Reza and Hai Dong for helping me to have such a wonderful time at Curtin University.

I would like to thank my colleagues, especially Dr Hafiz Farooq, Dr. Amir Hayat and Dr. Arshad Ali at the School of Electrical Engineering and Computer Science, National University of Technology, Pakistan for their help in making the transition from SEECS to the commencement of a PhD at Curtin University so smooth.

List of Publications arising from this thesis

Referred Journal Articles

1. Naeem Khalid Janjua and Farookh Khadeer Hussain "Web@IDSS: Argumentation-enabled Web-based IDSS for reasoning over incomplete and conflicting information", Knowledge-Based Systems, Volume 32, August 2012, Pages 9-27, doi:http://dx.doi.org/10.1016/j.knosys.2011.09.009 **(Excellence in Research for Australia (ERA) rank: B, Impact Factor: 2.422)**

2. Naeem Khalid Janjua, Farookh Khadeer Hussain and Omar Khadeer Hussain "Semantic information and knowledge integration through argumentative reasoning to support intelligent decision making", Information Systems Frontiers, July 2012, Pages 1-26, doi:http://dx.doi.org/10.1007/s10796-012-9365-x **(Australian Business Deans Council (ABDC) rank: A, ERA rank: B, Impact Factor: 0.912)**

Referred Conference Articles

3. Naeem Khalid Janjua and Farookh Khadeer Hussain. Development of a Logic Layer in the Semantic Web: Research Issues. In 6^{th} International conference on Semantics, Knowledge and Grids, Pages 367-370, Nov. 1-3, 2010, Ningbo, China.

4. Naeem Khalid Janjua, Farookh Khadeer Hussain. Defeasible Reasoning based Argumentative Web-IDSS for Virtual Team (VT). In 2011 International conference on Web Intelligence and Intelligent Agent Technology, Pages 330-334, August 22 - 27, 2011, Lyon, France.

5. Naeem Khalid Janjua, Farookh Khadeer Hussain. Rule-based business policies representation, reasoning and integration in an enterprise. In IEEE 6^{th} International conference on Broadband and Wireless Computing, Communication and applications, Pages 51-56, October 26-28, 2011, Barcelona, Spain.

Chapter 1 - Introduction

With recent technological developments, the World Wide Web (WWW) is no longer simply a medium for sharing information over the Internet, but has become one of the sources for generating new knowledge by using existing information for commercial, social, educational, business and research-related activities. As a result, software applications and information services have become the real wealth of a knowledge-based society. However, the explosion of information on the WWW poses great challenges in the design and development of software systems to exploit this information, extract new knowledge autonomously and facilitate decision making processes. To address such challenges, the concept of the Semantic Web has been proposed in the literature. The Semantic Web aims to be a universal medium for data exchange i.e. classifying, packaging and semantically enriching information to support data automation, integration, and reuse across various applications and extract knowledge from it (Torroni et al., 2009; Suguri et al., 2008).

In the next sub-section, the important characteristics of the Semantic Web are discussed.

1.1 The Semantic Web

"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation" (Lee et al., 2001).

The Semantic Web (Daconta et al., 2003) initiative was proposed by the inventor of the WWW, Tim Berners Lee, to enable the sharing of information beyond the boundaries of applications and websites. The vision of the Semantic Web is to transform the current state of the Web which is confined to human readability to a machine understandable Web. This is achieved by the use of semantic annotations, also known as meta-data, to describe the meaning/context of information on the Web in order

to make them programmable by Semantic Web applications and software agents who have no prior knowledge of them. This will facilitate automated information extraction, reasoning, knowledge generation and the integration of knowledge from diverse sources by Semantic Web applications to facilitate the decision making process. In order to elucidate the vision of the Semantic Web, consider the example where an electronic company's website has annotated Web contents (also known as Web resources) to distinguish between the company's name, its location and the products on the Web. Such annotation of the website's contents helps other Semantic Web applications to search for specific products and related information such as available locations, brands and prices, compare and contrast them and provide customized results to the decision maker.

Although the annotations of a website's contents allow the sharing of annotated information beyond its boundaries, the challenging aspect before the research community is how the annotations of different websites can be aligned or combined if everyone uses their own terminologies. The solution lies in the organization of shared vocabularies, so-called ontologies (Fensel, 2003), and using the references of these ontologies in order to bring inter-operability between different Web resources and software applications. For example, a hotel ontology can be used to relate the rating of certain hotels in a given country. Similarly, a countries ontology could be used to determine that WA is an Australian state and Perth is a city in WA. Such information (i.e. a hotel ontology and a countries ontology) is crucial for Semantic Web applications that make reservations for people by establishing a connection between the decision maker's requests for accommodation in WA, and a hotel advertisement specifying Perth as the hotel location.

In order to realize the sharing and use of ontologies between different Semantic Web applications, considerable progress has been made towards the development and use of standards, languages, technologies and applications. The Semantic Web stack as shown in Figure 1.1, illustrates the hierarchy of languages layered in the form of a cake,

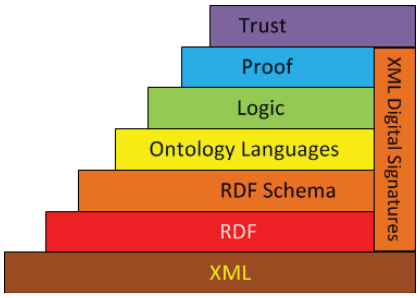


Figure 1.1: The Semantic Web stack (reproduced from Horrocks et al. (2005))

where each layer exploits and uses the capabilities of the layers below it (Antoniou and Van Harmelen, 2004). The layers in the Semantic Web stack are as follows:

1. The bottom layer is the eXtensible Markup Language (XML). It is a language for marking Web contents with tags that make it simpler for software applications to parse the data and process it. The XML standard supplies a grammar and syntax for tagging contents and also a behavioural standard for parsing those tags.

2. The Resource Description Framework (RDF) layer is located above the XML layer. RDF is a basic data model for writing simple statements about Web objects in the form of so-called triples. The RDF data model does not rely on XML, but RDF has an XML-based syntax.

3. The next layer is the RDF Schema Layer (RDFS) which provides basic vocabulary for RDF to create hierarchies of classes and properties.

4. The ontology languages layer provides a set of languages such as the Web Ontology Language (OWL), OWL 2, Web Service Modeling Ontology (WSMO) etc. for knowledge representation on the Semantic Web. OWL extends RDFS and is an advanced, computationally stable way of defining highly complex and interdependent data models in the Semantic Web.

5. The next layer is the logic layer which is used to extend the ontology language further with application-specific knowledge in some declarative language.

6. The proof layer involves the representation of proofs in Web languages and proof validation.

7. Finally, the top layer is trust which emerges through the use of digital signatures, and other kinds of knowledge, based on recommendations by agents they trust, or rating and certification agencies and consumer bodies.

The first four layers in the Semantic Web stack i.e. XML, RDF, RDFS and ontology languages, have reached maturity, resulting in a number of standard ontology languages being defined. Current research in the area of the Semantic Web stack focuses on the logic layer for the realization of advanced reasoning capabilities in Semantic Web applications. In the next sub-sections, the ontology languages layer and the logic layer is overviewed in detail and the advancements made to them are explained.

1.1.1 Ontology languages layer

The ontology languages layer comprises a set of ontology languages. Each ontology language is capable of the formal and explicit specification of a certain domain using a combination of classes, their relationships or properties, instances and axioms. To elucidate this with an example, consider a simple ontology named *Person* as depicted in Figure 1.2. The person ontology comprises a class *'person'* with two subclasses i.e. *'student'* and *'researcher'*. The ontology has one object property i.e. *hasSupervisor*.

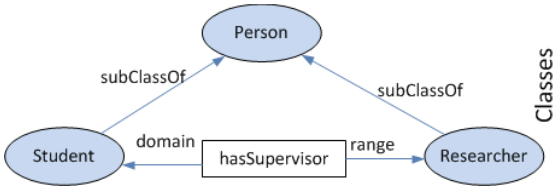


Figure 1.2: Pictorial representation of the person ontology

Consider information such as '*Naeem has a supervisor named Omar*'. In order to make this information understandable by different Semantic Web applications, we need to annotate the information with the person ontology. As depicted in Figure 1.3, *'Naeem'* and *'Omar'* are depicted as the instances of the class *student* and *researcher*, respectively. Similarly, *'Naeem'* has an object property i.e. *hasSupervisor* which relates him to his supervisor. As previously stated, ontologies are shared vocabularies, therefore the information annotated with the person ontology can be understandable by other Semantic Web applications by using the person ontology.

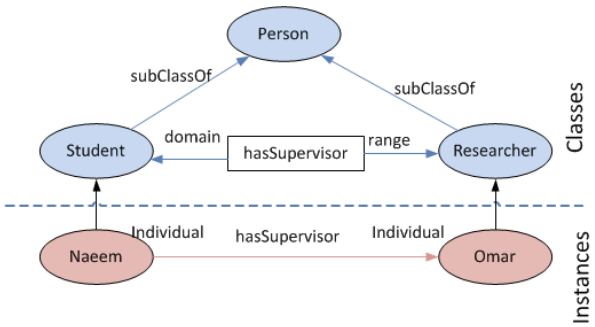


Figure 1.3: Pictorial representation of the person ontology with instance data

There are several advantages of knowledge representation in the form of ontologies. Wang et al. (2004) identified the following advantages:

1. Knowledge Sharing: The use of ontologies for context specification enables software agents and services to have a common set of concepts in a context while interacting with one another.

2. Logic-based reasoning: Based on ontologies, software agents and services can exploit various existing logic-based reasoning mechanisms to deduce new information from existing information.

3. Knowledge Reuse: By reusing well-defined Web ontologies of different domains (e.g. hotel ontology and countries ontology), new ontologies can be composed without starting from scratch.

In the next-subsection, the different languages which can be used to represent ontologies are discussed.

1.1.1.1 Ontology languages

Web Ontology Language (OWL) is a W3C proposed standard for representing knowledge on the Semantic Web and it provides constructs for cardinality restrictions, boolean expressions and restrictions on properties (Dean and Schreiber, 2004). It is based on Description Logic (DL) and has three variants with different levels of expressiveness for reasoning i.e. OWL Lite, OWL DL and OWL Full.

The Web Ontology Language [1] 2, informally OWL 2, is an ontology language for the Semantic Web that became a W3C Recommendation on Oct 27 2009. OWL 2 is compatible with the OWL standard of 2004 which it supersedes. As in OWL, the main syntactic form of OWL 2 ontologies is based on an RDF serialization, although various alternative syntactic forms are available too. OWL 2 is also available in three variants i.e. OWL 2 EL, OWL 2 QL and OWL 2 RL. Figure 1.4 shows the specification of person ontology with RDF/OWL in turtle format.

Similarly, efforts have been made towards building Semantic Web services using service description standards based on ontologies i.e. Web Ontology Language for Services (OWL S) , WSMO (Martin et al., 2007).

1.1.1.2 Ontological reasoning

Reasoning is a cognitive process by which a conclusion is reached. Using ontological reasoning Semantic Web applications reason on information and derive new information that is not expressed in the ontology explicitly. Ontology languages such as RDFS, OWL and OWL 2 are based on DL, therefore for reasoning, they can exploit the considerable existing body of DL reasoning engines such as FaCT++ (Tsarkov and Horrocks, 2006) and Pellet (Parsia and Sirin, 2007).

[1] http://www.w3.org/2007/OWL/draft/ED-owl2-profiles-20090420/

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix : <http://www.semanticweb.org/ontologies/2012/9/Person.owl#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://www.semanticweb.org/ontologies/2012/9/Person.owl> .

<http://www.semanticweb.org/ontologies/2012/9/Person.owl> rdf:type owl:Ontology .

:hasSupervisor rdf:type owl:ObjectProperty ;
               rdfs:range :Researcher ;
               rdfs:domain :Student .
:Person rdf:type owl:Class .
:Researcher rdf:type owl:Class ;
            rdfs:subClassOf :Person .
:Student rdf:type owl:Class ;
         rdfs:subClassOf :Person .
###  http://www.semanticweb.org/ontologies/2012/9/Person.owl#Naeem
:Naeem rdf:type :Student ,
               owl:NamedIndividual ;
       :hasSupervisor :Omar .
###  http://www.semanticweb.org/ontologies/2012/9/Person.owl#Omar
:Omar rdf:type :Researcher ,
               owl:NamedIndividual .
```

Figure 1.4: Turtle representation of person ontology developed in OWL

DL reasoning helps the knowledge experts to design and maintain high quality ontologies. It drives the inference from existing ontological concepts and properties and detect whether the derived concepts and properties bring any inconsistency or contradiction in ontology. Therefore, a high quality ontology fulfils certain logical requirements in order to remain consistent. The important logical requirements are as follows (Antoniou and Van Harmelen, 2004):

1. Class membership: if x is an instance of a class C, and C is a sub-class of D, then we can infer that x is an instance of D.

2. Equivalence of classes: If class A is equivalent to class B, and class B is equivalent to class C, then we can infer that class A is equivalent to class C.

3. Class consistency: If x is an instance of a class A and A is a subclass of B $\sqcap$ C, A is subclass of D, and B and D are disjoint. Then we have an inconsistency in an ontology because A should be empty. It is an important logical requirement for ontological reasoning as inconsistent ontologies may lead to erroneous conclusions.

4. Instance checking: If instance x satisfies a certain property-value pair of a class A (property-value pairs that are declared sufficient for membership in a class A), then x must be instance of class A.

In the next section, an introduction to the logic layer of the Semantic Web stack is provided and its current status is discussed in detail.

1.1.2 The logic layer

As the ontology layer of the Semantic Web has reached maturity (i.e. standards such as RDF, RDFs, OWL, OWL 2), the next step is to work on the logic layer to develop advance reasoning capabilities on semantic enriched data for new knowledge extraction and efficient decision making.

Adding the logic layer in the Semantic Web means making use of rules to make inferences. Rules are used to express computational or business logic, express policies or contracts in information systems which don't have an explicit control flow and are suitable for execution in dynamic situations for business collaboration. Rule-based systems have been extensively used in several applications and domains, such as databases, e-commerce, personalization, games, businesses (B2B, B2C) and academia. In e-Business, they can be used to represent sellers offering products and services (Grosof et al., 2009). Figure 1.5 demonstrates a simple example where, by using the concepts defined in the person ontology, the application on the logic layer with the DL reasoner defines a rule i.e. '*Person, who is a student and has a supervisor appears in the examination*'. The reasoning process results in the exploitation of the information defined in the person ontology and results in new information i.e *apperinexam(naeem)*.

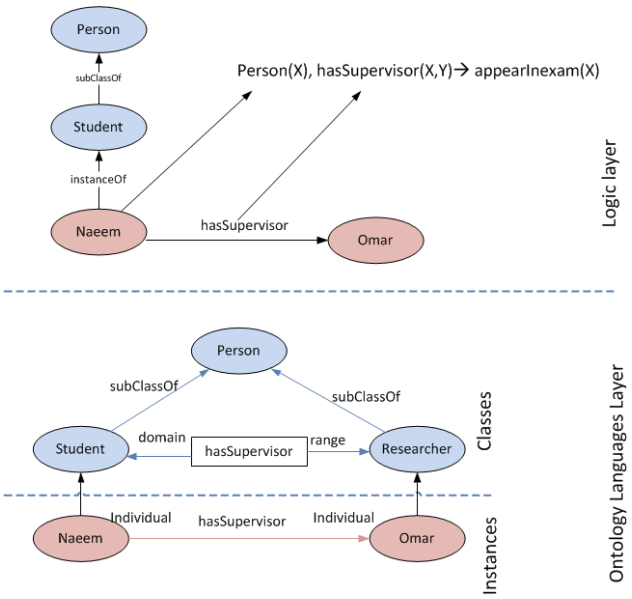


Figure 1.5: Logic layer exploiting ontological knowledge

Rules can be classified into different categories as explained in the next sub-section.

1.1.2.1 Rules classification and knowledge acquisition

Boley et al. (2007) grouped the rules on the Semantic Web into the following categories:

- Deductive rules: Deductive rules are the statements of how to derive information from other information by using logical inference. The execution of deductive rules results in making implicit information explicit. To explain with an example, the following rule
  IF
  movie ?M was produced before 1930
  THEN
  ?M is a black and white movie
  infers that if a movie was produced before 1930, then it is a black and white movie as there were no colour movies at that time. Deductive rules are also known as derivation rules in the business rules community, constructive rules by logicians, and views in the database community.

- Normative rules: Normative rules are those that pose some constraints on the data or on the business logic to ensure their consistency in the ontology or knowledge base. To explain with an example, the following rule
  IF
  ?C is Customer
  THEN
  ?C has unique identification number
  infers that if someone is a customer, then he must have a unique identification number.

- Reactive rules: Reactive rules are those rules which, when executed, update the ontology or knowledge base upon which they are being acted. In reactive rules, we verify the satisfaction of conditions and also execute the action whenever message arrival or timer event triggers the rule. The reactive rules are further grouped into the following two categories:

  * Event-Condition-Action (ECA) rule: ECA rules are rules of the form *ON Event IF Condition DO Action*, where Action should be executed if the Event occurs, provided that the Condition holds. A simple example of ECA rule is as follows:
  ON request from customer ?C to book a movie
  IF
  customer ?C is blacklisted
  DO
  deny ?C's request

  * Production rules : Production rules are rules of the form IF Condition DO Action, where Condition queries the working memory containing the data

on which the rules operate. Action should be executed whenever a change to the underlying database makes the condition true. A simple example of an ECA rule is as follows:
IF
customer ?C is loyal
Then
give Discount to ?C

Each type of rule discussed have different requirements for implementation. Reactive rules require more complex language for representation and reasoning compared to the realization of deductive and normative rules.

To represent the different kinds of rules on the Semantic Web, different rule-based languages have been proposed. The simplest of them is N3Logic, a logic proposed by Tim Berners-Lee, that allows rules to be expressed on the logic layer in the Semantic Web. It extends RDF with syntax for nested graphs and quantified variables with predicates for implication and accessing resources on the Web. It also includes functions for computation such as cryptographic, string, math etc. The main goal of N3Logic is to be a minimal extension to the RDF data model so that the same language can be used for logic and data representation (Berners-lee et al., 2008).

Similarly, the Rule Markup Language (RuleML) [2] is an international effort to standardize the inference rules on the Semantic Web for the seamless publishing and sharing of rule bases. The objective is on rule interoperation between industry standards. RuleML builds a hierarchy of rule sub-languages upon XML, RDF and OWL, e.g., Semantic Web Rule Language (SWRL). SWRL is intended to be the rule language of the Semantic Web. It includes a high-level abstract syntax for Horn-like rules and the rules are expressed in terms of OWL concepts (classes, properties, individuals). Rules are saved as a part of ontology.

There are two ways in which rules can be used for knowledge acquisition on the Semantic Web as depicted in Figure 1.6. A one-way knowledge flow exists from an ontology module to a rule-based reasoning module, where an ontology module's instances are imported as basic facts and filtered with conditions in the rules. This passive knowledge query uses only deductive rules, whereas if a rule engine derives implicit new facts and updates those facts back to an ontology module, then this is a reverse knowledge flow from a rule module to an ontology module. This reverse knowledge flow requires normative and reactive rules (Dix et al., 2009). It is important to note here that knowledge acquisition by either active or passive knowledge is realized in the presence of a certain reasoning methodology.

[2] ruleml.org

In the next section, I discuss the current reasoning methodology being used by Semantic Web applications.

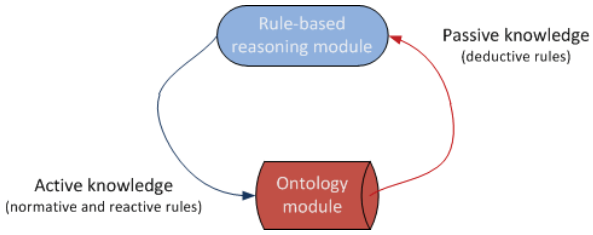


Figure 1.6: Two way knowledge acquisition on the Semantic Web

1.1.2.2 Reasoning methodology

Reasoning is the core by which Semantic Web applications reach a conclusion. It is applied in various areas such as product recommendations, auctions, identification of requirements, vendor selection, negotiation, agent communication and information integration (Deng and Wibowo, 2008; Cheung and Cheong, 2007; Shim et al., 2002; Assche et al., 1988; Wen et al., 2008). It is performed at the logic layer of the Semantic Web. Current rule-based languages such as N3Logic, SWRL, OWL-RL etc. and ontology languages such as RDF, RDFs, OWL and OWL 2 are based on DL (Baader et al., 2005) which provides syntax and semantics to model concepts, roles and individuals, and their relationships.

DL is a subset of predicate logic (Van Emden and Kowalski, 1976) and thus inherits its limitation i.e. it adopts a standard logical model of open-world assumption (OWA) where a statement can't be assumed true on the basis of failure to prove it. In other words, OWA states that there can be true facts that are not contained in the knowledge base. This can be elucidated with the help of an example. By taking into account the ontology defined in Figure 1.2, if we want to know the truthfulness of the statement that *'Naeem' is a citizen of 'Pakistan'*, a logic based on a closed-world assumption (CWA) will return a *'No'* because a closed-world assumption implies that everything we don't know or information which is not present in the model is considered to be false. On the other hand, an OWA states that everything we don't know is undefined. The DL and the inferences performed in Semantic Web applications over it follow OWA, such reasoning being called monotonic reasoning. The following is an example of monotonic reasoning:

- Premise: All students are Person.
- Premise: Naeem is a student.

- Conclusion: Therefore Naeem is a Person.

Representing in a logical notation, considering $\mathcal{T}$, $\mathcal{F}$ and $\mathcal{G}$ representing some statements, then monotonic reasoning can be expressed formally as follows:

$$\mathcal{T} \models \mathcal{F} \rightarrow \mathcal{T} \sqcup \mathcal{G} \models \mathcal{F} \quad (1.1)$$

It is evident from equation 1.1 that, in monotonic reasoning, if we enlarge the set of axioms, we cannot retract any existing assertions or axioms. To explain with an example, consider a knowledge base containing the following information at one point in time:

- Premise: All students are Person.
- Premise: Naeem is a student.
- Conclusion: Therefore Naeem is a Person.

Later on, some new information comes into the knowledge base as follows:

- Premise: Naeem has graduated from Curtin.

After the addition of new information in the knowledge base, if I query the knowledge base for *'Naeem is a student?'*, it will return true. Similarly, if I query the knowledge base for *'Naeem has graduated from Curtin?',* it will return true as well.

It is evident from the example that the addition of new information does not result in the retraction of previous information. Nute (1994) argued that such reasoning does not add to the knowledge base (e.g. *'Naeem is not a student'*), it merely rearranges existing knowledge. In monotonic reasoning, a knowledge base cannot represent and reason on contradictory information. In this case, the premise '*Naeem is a student*' is already present in the knowledge base and if a system asserts a new premise i.e. *'Naeem is not a student as he has graduated from Curtin'*, it will result in an error. This problem can be overcome by using non-monotonic reasoning where the knowledge base can represent and reason in the presence of contradictory information. In the next sub-section, the current research efforts on the logic layer are explained and the support for non-monotonic reasoning is discussed.

1.1.2.3 Logic layer and support for non-monotonic reasoning

The initially proposed single stack architecture (SSA) of the Semantic Web by Tim Berners-Lee assumed that the Semantic Web stack is composed of a main language and every new development should be built on top of existing layers (Berners-Lee, 2000; Lee, 2003). In response to criticisms that this proposal was unrealistic and unsustainable, Berners-Lee then proposed an alternative multi-stack architecture (MSA) to overcome the limitation of SSA (Lee, 2005, 2006). The MSA, as depicted in Figure 1.7, is more realistic in the long run and in such a framework, rules lie next to the ontology layer which results in the following advantages:

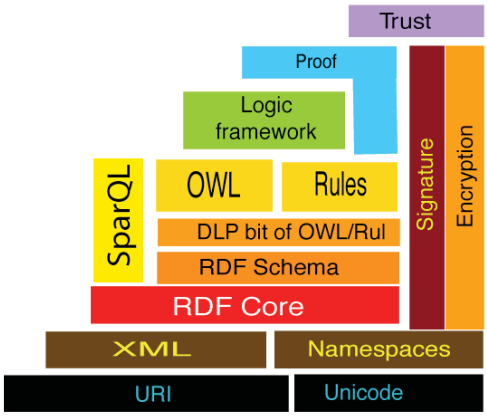


Figure 1.7: Updated Semantic Web stack (reproduced from Horrocks et al. (2005))

- they can serve as an extension of, or alternative to, DL-based ontology languages
- they can be used to develop a declarative system using ontological information
- combining DL with rules will make possible the execution of expressive queries on instances
- rules can also be useful in defining integrity constraints over individuals of an ontology e.g. axioms *Person has SSN* and *Person(george)* are satisfiable in OWL even if we don't define an SSN for George (Meditskos and Bassiliades, 2009). Additionally, it is impossible to assert that persons who study and live in the same city are '*home students*' in OWL, while this can be done easily using rules:

$$studies(X,Y), lives(X,Z), loc(Y,U), loc(Z,U) \longrightarrow homeStudent(X). \quad (1.2)$$

Significant debate is being generated on the suitability of Logic Programming in the domain of the Semantic Web (Grosof et al., 2003). Logic Programming is a predominant paradigm for expressing knowledge with rules, and making inferences and answering queries. It provides both a declarative reading (a programming paradigm that expresses the logic of a computation without describing its control flow) and an operational reading of rules (with implementations). Its semantics underlie a large part of four families of rule systems i.e. SQL relationship databases, OPS5 heritage production rules, Prolog, and Even-Condition-Action rules and are being used as a proposal for rules in the context of the Semantic Web.

Many efforts have focused on the mapping, intersection or combination of DL and logic programs (LP) in order to overcome the shortcomings that emerged during the development of practical OWL applications (Patel-Schneider and Horrocks, 2007). In order to overcome the limitations of reasoning on OWL, Grosof et al. (2003) proposed Description Logic Programs (DLP) which lie at intersection of LP and DL (as shown in Figure 1.8) instead of using Full First Order Logic (FOL) to address OWL issues.

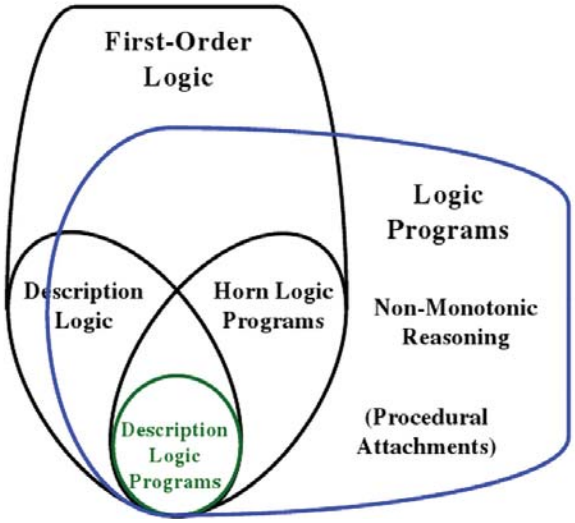


Figure 1.8: Expressive overlaps among knowledge representation languages (Grosof et al., 2003)

FOL can express (positive) disjunctives which are inexpressible in LP, but it does not provide support for expressing negation-as-failure (representation of incomplete information) and procedural attachments (the association of action performing procedural invocation with the drawing of conclusion about particular predicate). Negation-as-failure is a way to represent incomplete information in the body of a rule and is present by the word '*not*'. To explain this, consider the following rule:

$not\ Train \longrightarrow DonotCross$

This states that if it is unknown whether a train is approaching or not, then do not cross the railway lines. In such cases, the absence of information i.e. whether a train is coming or not, does not restrict the application to reason and reach a decision. The importance of such reasoning has been raised by researchers that has resulted in the introduction of the '*Two Towers*' Semantic Web stack as depicted in Figure 1.9.

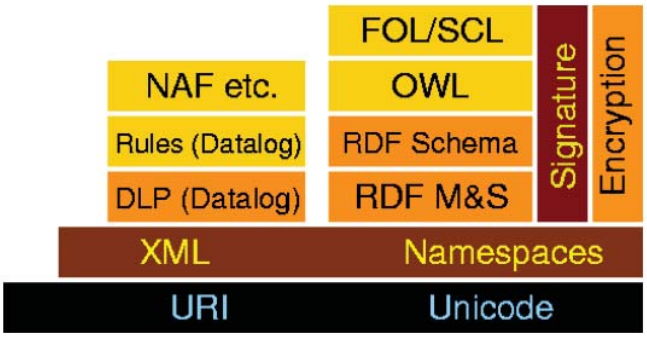


Figure 1.9: Semantic Web layer cake with negation-as-failure (reproduced from Horrocks et al. (2005))

On the other hand, DLP does not provide features to support the non-monotonic behaviour of the system. Non-monotonic behaviour, in contrast to monotonic behaviour, follows CWA and is performed when the underlying information is incomplete and/or contradictory. It is evident from the discussion that the current Semantic Web development technologies do not support non-monotonic reasoning and follow an assumption that *"the underlying information for decision making is consistent and the addition of new information doesn't result in a contradiction with existing information"*. In other words, they assume that

i) no conflicts will arise during the process of decision-making, and

ii) the introduction of new information will not result in achieving a different output.

But by using such assumptions, the current Semantic Web applications ignore information that might be incomplete and/or contradictory but may be important in providing better insights in the decision making process. This can be elucidated with the help of an example of an online purchase of a book. Before making the decision to buy a book, a buyer evaluates a list of available books and compares the titles on the basis of the information provided by various Web site users in the form of arguments and counter-arguments in respect of each book. After reading the information and evaluating the arguments and counter-arguments, the buyer convinces himself either to buy a specific book or not to make a purchase. Buyers can also submit their rationale (arguments and counter-arguments) for the choice made. There may be

incomplete and/or contradictory reviews provided by website users which need to be solved to discover the correct insight from it. This is especially important in Semantic Web applications where the aim is for applications to act autonomously on behalf of users. In such cases, their ability to deal with either incomplete and/or contradictory information is crucial to facilitate the decision making process and achieve Business Intelligence.

In next section, the importance of Business Intelligence in Semantic Web applications is highlighted. The challenges faced by Semantic Web applications in achieving Business Intelligence due to the absence of non-monotonic reasoning is also discussed.

1.2 Challenges in Semantic Web applications for Business Intelligence in an enterprise

Over the past few decades, advancements in Internet, World Wide Web (WWW) and Artificial Intelligence (AI) technologies have engendered a resurgence of interest in the use of software intelligence for business applications, known as Business Intelligence (BI). While the term BI is relatively new, computer-based business intelligence systems go back, in one form or another, for close to forty years (Power and Sharda, 2009). BI as a term has been used interchangeably with decision support systems (DSS), executive information systems, and management information systems (Thomsen, 2003). Formally, Business Intelligence (BI) is the use of high-level software intelligence to produce actionable information that is delivered at the right time, and is immediately accessible, easily comprehendible and exportable to other softwares to assist the business decision-making process (Negash and Gray, 2003). It involves finding, gathering, aggregating, and analysing information from different heterogeneous sources for decision making.

The Semantic Web provides the tools and technologies to realize BI in an enterprise (Saggion et al., 2007). Of the different layers of the Semantic Web stack, the ontology layer helps to semantically annotate the information coming from different heterogeneous sources and makes it understandable by Semantic Web applications. The logic layer helps the Semantic Web applications to specify rules using logic-based languages in order to perform advance reasoning on semantically annotated information and generate knowledge from it.

Although the Semantic Web provides languages for the development of Semantic Web applications for BI in order to facilitate decision making in an enterprise, as

mentioned in the last section, there are cases where the current Semantic Web applications for BI are facing some challenging situations. Some of these important areas are:

1. Representation and reasoning over structured information that may be incomplete and/or contradictory and exists within the enterprise and/or in other enterprises

   The development of the Semantic Web has helped information systems to overcome the limitations of semantic heterogeneity and Web-based DSS is now an active area of research in BI, impacting significantly on the way information is exchanged and businesses are conducted. However, to remain competitive, companies rely on BI to continuously monitor and analyse the operating environment (both internal and external) for them, in order to identify the potential risks, and to devise competitive business strategies.

   To explain with an example, consider a scenario where an enterprise '*abc*' asks its departments to forward their recommendations for selecting a relocation service *'xyz'*. To make recommendations, the departments have to take into account the supplier's business policies, their service reputation and how the enterprise requirements can be fulfilled by them. A Web-based DSS that can represent information and reason about it can help the enterprise to make the recommendation i.e. *"whether or not to select the relocation service xyz"*. However, the current Web-based DSS applications are not able to represent and reason over such information that is present within an enterprise and/or in other enterprises, which could be incomplete and/or contradictory and provide no decision support to decision makers. For example, consider the following information:

   - *if company(xyz) and placeOrder(abc,xyz) then giveDiscount(abc)*
   - *if shopper(abc), not make an advancePayment(xyz) then ¬ giveDiscount(abc)*

   where symbol '¬' is used to represent contradictory information. This will be explained further in the next section.

   The first rule specified by enterprise *'abc'* states that if we place an order, then we expect a discount from *'xyz'*. However, the *'xyz'* policy states that if a shopper places an order and he does not pay in advance, then he will not get any discount. If such information exist within an enterprise or in other enterprises

then the existing reasoning engines on the logic layer are not able to capture that information and utilize it in reaching a conclusion. The situation becomes more complicated when information comes from website users who have used the service and provided diverse feedback on the WWW. Taking this information into account can guide the reasoning process to a different output.

2. Integration of information/results generated by different Semantic Web applications in an enterprise to support intelligent decision making

   The availability of integrated, high quality information is a pre-requisite for a decision support system (DSS) to aid the enterprise level decision-making process. The introduction of the Semantic Web ensures the seamless integration of information derived from diverse sources and transforms the DSS into an adoptable and flexible Semantic Web-DSS (Web-DSS). But, as discussed in the first scenario, the current Semantic Web lacks the capability to represent, reason and integrate incomplete and/or contradictory information. This, in turn, renders an enterprise incapable of *knowledge integration*; that is, the integration of knowledge about a subject that could be incomplete, contradictory and distributed among different Web-based DSS within an enterprise or in other enterprises.

   This can be elucidated with the help of an example where the higher level management of an enterprise asks for recommendations from different departments about selecting a relocation service provider. The integration of different recommendations forwarded by each department into a single recommendation is known as *knowledge integration*. It is important to note that the recommendations made by each department could be incomplete and/or contradictory and thus it is a great challenge for the Web-based DSS to represent reason and integrate those diverse recommendations in order to assist higher level management in making a final decision about the selection of a service provider '*xyz*'.

3. Representation, reasoning and integration of unstructured information that may be incomplete and/or contradictory and exists within the enterprise and/or in other enterprise

   In recent past years there is tsunami of data that has been generated and unstructured information accounts for around 80% [3] of the information in it. This information ranges from customer reviews, users buying preferences for

[3] http://www.aiim.org/Research-and-Publications/Research/White-Papers/Data-is-Unstructured-Information

new product, business policies of an enterprise or collaborating enterprises etc, which when considered by applications can provide better insights in the decision making process according to their needs. However, it is also possible that such information may be in different formats and potentially incomplete and/or contradictory within themselves or with information coming from heterogenous sources. Such scenario can be explained by the example of unstructured business policies in an enterprise.

As it is known, business policies are of paramount importance in the working of an enterprise. Operational business processes that are derived from business policies consists of business processes and business rules that define how an enterprise carries out its operations. However, it has been observed that over a period of time, operational business processes may not comply with enterprise business policies. A lack of systematic methodologies to check for such non-compliance results in the dependence of enterprises on ad-hoc, time-consuming process mapping techniques. Although previous work in the literature considers the discovery of business processes from business policies (Wang et al., 2009), their defeasible nature is not considered, where conflicts may arise in business policies due to the following factors:

(a) elicitation of business policies by different viewpoints by different department, and

(b) merging the business policies of two different enterprises that are seeking a possible merger to address new market challenges (Rajsiri et al., 2010).

To explain the defeasible nature of business policies, consider a very common example concerning pricing policy, discussed in the literature (Antoniou and Arief, 2002; Grosof et al., 2002). A typical scenario in a pricing policy is whether or not to give a discount to individuals based on their purchasing history. The business rules to achieve this functionality could be constructed as follows:

R1 (company a) 5% discount if a buyer is a loyal customer

R2 (company b) 10% discount if a buyer has a history of large spending

R3 (company a) No discount if a buyer has a late-payment history

Suppose a buyer ‘Jon’ is a loyal customer. As a result, the business rule ‘R1’ applies to him and he receives a 5% discount. However, it is later learned that ‘Jon’ also has a late-payment history. In such a case, the decision made earlier on the basis of business rule ‘R1’ may have to be retracted in view of the new information because of business rule ‘R3’, even though Jon is a loyal

customer. In such situations, a policy decision made earlier may become in conflict in the presence of contradictory information. To address such issues, there is need for a framework that analyses business policies, provides different conflicts resolution strategies for the decision maker and after their resolution generate a graphical representation of the process (in the form of business process map). The generated business process map will provide a complete picture for the business manager in identifying and making recommendations to resolve the non-compliance of operational business processes with the business policies.

It can be seen from above mentioned challenges that the notion of information that is present at one point of time can be changed by the introduction of new information which may be either incomplete and/or in conflict with the information on hand. In such situations, a decision made earlier needs to be reconsidered and reasoned again in the presence of new incomplete that may be incomplete and/or contradictory. For reasoning over such incomplete and/or contradictory information, a system needs to perform non-monotonic reasoning. Such reasoning is an important feature that needs to be included in Semantic Web applications. In current Semantic Web applications, a decision once made can't be retracted. As a result, non-monotonic reasoning can't be realized in current Semantic Web application for BI.

The current challenges being faced by the Semantic Web applications discussed above have been tackled, one way or other, in the area of Artificial Intelligence (AI). In the next section, a reasoning methodology from the area of AI for incomplete and contradictory information representation and non-monotonic reasoning is discussed.

1.3 Defeasible reasoning

The term 'defeasible reasoning' was coined as a concept in the philosophy of law to mean 'convincing' although not rigorous reasoning. Defeasible reasoning is a rule-based approach to perform reasoning on uncertain information where a rule supporting a conclusion may be negated or invalidated with the emergence of new information, as evident in the following example:

A *Tweety flies because it is a bird*

B *Tweety does not fly because it is a penguin*

C *Tweety flies because it is a magic penguin*

A concludes that Tweety flies because it's a bird, however, later information from *B* negates the previous conclusion and states that Tweety cannot fly because it's a

penguin. C negates the conclusion of B and supports A's conclusion by providing justification that Tweety flies because it's a magic penguin.

Defeasible reasoning is a simple and efficient implementation of rule-based non-monotonic reasoning. It can represent facts, rules, and priorities among rules. It provides enhanced representational capabilities with low computational complexity as compared to mainstream approaches for non-monotonic reasoning (Antoniou and Bikakis, 2007). Antoniou et al. (2007) summarizes the important features of defeasible reasoning as follows:

1. It is a rule-based approach without disjunction.
2. Classical negation, represented by the symbol '$\neg$', is used in the head to represent contradictory information and '*not*' is used in body of a rule to represent incomplete information.
3. Rules may support contradictory conclusions.
4. Reasoning is skeptical in the sense that contradictory rules do not fire. Thus, consistency is preserved.
5. Priorities on rules may be used to resolve conflicts among rules.

Formally, a defeasible theory D is a triple (F, R, $>$) where 'F' is a set of literals (called facts), 'R' a finite set of rules and '$>$' a superiority relation on R. The set of rules are categorised into the following two categories:

1. Strict rules: Strict rules are rules whose conclusion can't be retracted, denoted by '$\rightarrow$.' An example of a strict rule is '*Professors are faculty members*' and is written formally as *professor(X)* $\rightarrow$ *faculty(X)*. Strict rules are intended to define relationships that are definitional in nature, for example ontological knowledge.
2. Defeasible rules: Defeasible rules are rules whose conclusion can be retracted in the presence of new information. It is denoted by '$\Rightarrow$'. An example of a defeasible rule is '*Professors are typically tenured*' and is written formally as professor(X) $\Rightarrow$ tenured(X). The main point is that the information '*Professors are tenured*' is not sufficient evidence to conclude that a particular professor is tenured. Defeasible rules are intended to define information which is not absolutely true and may be overridden by new information.

A superiority relation on defeasible rules is represented by the symbol i.e. ' $>$'. When r1 $>$ r2, then r1 is called superior to r2 and r1 will be executed and its conclusion will

be added in the knowledge base. This expresses that r1 may override r2. For example, given the rules

$r : professor(X) => tenured(X)$

$r^{'}$:$visiting(X) => \neg tenured(X)$

which contradict one another, no conclusive decision can be made about information regarding whether a visiting professor is tenured. But if a superiority relation $>$ with $r^{'} >$ r is introduced, then it can be concluded that he/she cannot be tenured. Antoniou and Wagner (2003) highlighted the importance of defeasible reasoning in Semantic Web applications and outline its important in areas of Modeling Business Rules and Policies, recommender and Brokering systems, and declarative negotiation strategies.

Although defeasible reasoning seems to be a good option to address the issues of non-monotonic reasoning in Semantic Web applications, the superiority relation on defeasible rules are hard-coded individual preferences and if conflicts arise at run time, defeasible reasoning doesn't provide any solution. This can be explained this with an example by assuming that a virtual team (VT) for the Olympic Games comprises the the Olympic International Committee (OIC), the Organising Committees for the Olympic Games (OCOG) and the host city (HC). The objective of the virtual team is to make important decisions about sports activities. These three committees have their own particular goals and expectations which impact on the overall organisation of the sports events. Further assume that the current task of VT members is to decide "*whether or not a scheduled match will be played in rainy conditions*". To accomplish this task, each member of the VT provides his/her views in the form of rules about the stated task in a defeasible reasoning system as follows:

OIC *if ground(perth), not rain(monday)* $\Rightarrow \neg$ *groundReady(perth)*

OCOG *if ground(perth), drainage(perth,good), rain(monday)* $\Rightarrow$ *groundReady(perth)*

HC *if ground(perth), conditionOfLight(perth,bad)* $\Rightarrow \neg$ *groundReady(perth)*

As previously mentioned, in a defeasible reasoning system, a member can define a superiority relation between two contradictory rules only if both of them are defined by him. As there are conflicts in the rule base between the rules defined by different members of the VT i.e. the rule defined by '*HC*' and '*OIC*' is in conflict with '*OCOG*', the defeasible reasoning system cannot resolve conflicts through reasoning and fails to assist the VT in the decision-making process. To find a solution for this problem, argumentation which is a human's way of handling conflicts during debates and discussions provides a good option. In the next section, an introduction to argumentation is provided.

1.4 Argumentation

Argumentation is a rich interdisciplinary area of research, traditionally spread across philosophy, communication studies, linguistics and psychology. In our daily life, argumentation, however, often has negative connotations, suggesting quarrelsomeness and unpleasantness. However, this is not true in all cases. In a classical sense, argumentation is the study of effective reasoning to reach to a conclusion which is the key way humans deal with incomplete and/or contradictory information by taking into account the exchange and evaluation of arguments and counter-arguments relevant to a certain issue (Zarefsky, 2009). Decisions from argumentative reasoning are backed by an explanation generated during the choices made.

Argumentation is inherently a process rather than an instant picture and the building blocks of argumentation are arguments and relationship between those arguments (Loui, 1998). According to Walton (2009); Palau and Moens (2009), an argument is a set of statements (propositions) made up of three parts, a conclusion, a set of premises, and an inference from premises to conclusion. During the process of argumentation, relationships among the arguments link them with each other in a certain pattern to support the ultimate conclusion. Such linking patterns are called "Argumentation Schemes" which provide a way to perform reasoning over the set of premises and conclusion. These argumentation schemes have emerged from informal logics (Walton, 1989). Schemes help categorize the way arguments are built and aim to fill the gap between logic-based applications and human reasoning by providing schemes capturing stereotypical patterns of human reasoning e.g. arguments from expert opinion schemes (Letia and Groza, 2008; Rahwan et al., 2007a).

Toulmin (Freeley and Steinberg, 2008; Toulmin, 2003) proposed a model to enhance the understanding of the structure of practical reasoning that occurs in any argument. He categorized premises which give arguments a richer structure, and one which corresponds more closely to the way in which arguments are presented. Figure 1.10 presents the elements of Toulmin's model of argument structure.

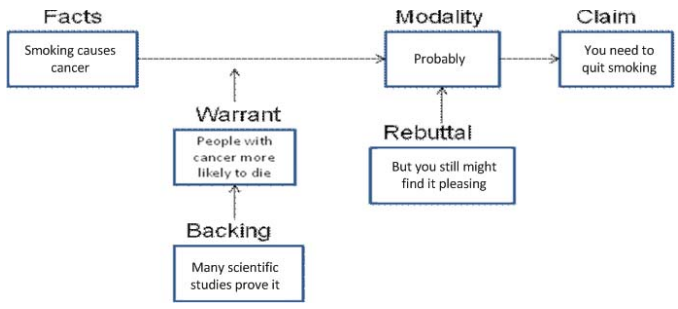


Figure 1.10: Toulmin's model of argument structure

Toulmin used modal qualification to express the concept of the degrees of cogency. The degrees of cogency are certainty, probability, plausibility or possibility as shown in Figure 1.11. According to Baroni et al. (1998), such classification can help in classifying the various ways an argument can be analysed. Perelman (1969) tried to find a description of the techniques of argumentation used to obtain the approval of others for their opinions, calling it "new rhetoric". Both Toulmin and Perelman tried to present an alternative to formal logic that is better suited to analyzing every day communication. Eemeren and Grootendorst (2004) studied argumentation as a means of resolving differences of opinion by considering argumentation as a discourse activity. They proposed the pragma-dialectical theory which views argumentation as ideally being part of a critical discussion which progresses through four discussion stages to resolve a difference of opinion: the confrontation stage, opening stage, argumentation stage and concluding stage. Argument diagramming is often claimed to be a powerful method to analyse and evaluate arguments. Different tools have been used for the diagramming of arguments e.g. Araucaria is a freely available, open source software package that allows the text of an argument to be loaded from an Argument Markup Language (AML) file, and provides numerous tools for marking up this text and producing Standard, Toulmin, Wigmore diagrams (Reed et al., 2007).

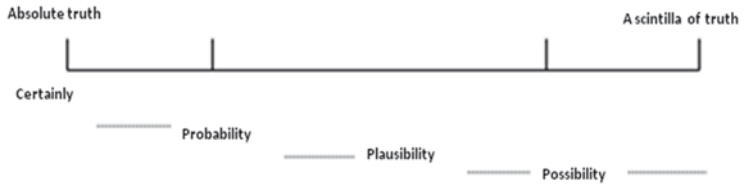


Figure 1.11: Degrees of cogency

Over the last couple of years, argumentation has gained a lot of attention from the artificial intelligence research community which led to the investigation of argumentation and its application in various domains. From its theoretical foundations, argumentation can be integrated into a number of real world applications such as planning, MAS, legal reasoning, knowledge engineering, analysis of news reports, clustering, argumentation support systems, mediation systems and computer-supported collaborated argumentation (Chesnevar et al., 2006b; Rahwan, 2005). The ASPIC project (Argumentation Services Platform with Integrated Components) involves the development of components implementing theoretical models for argumentation-based inferences, decision making and dialogue in Multiagent applications. Also efforts such as the Argument Interchange Format (AIF) for the development of standard shared notation to represent and exchange argumentation knowledge among agents are being made (Chesnevar et al., 2006a).

The large number of interactions between Web users on the WWW needs to be captured in a certain semantic structure in order to make it explore-able by others and to automate the process of argument build-up and analysis. Argument blogging is an attempt to provide an environment on the web where Web user can harvest the current resources on the web by structuring them into argumentative dialogues and storing the resultant dialogue into a database for further analysis and reuse (Wells et al., 2009). Web 2.0 can be used as a powerful paradigm for designing augmentation tools for solving challenges on a global scale in collaboration. Shum (2008) provides a comprehensive view of Web 2.0 features that are the driving force for the realization of argumentation on the Web. The value of argumentation formalisms have been harvested in the fields of philosophy, AI and the WWW however its potential has also been unlocked for Semantic Web applications. In the next section, I discuss and identify the current research gaps for the realization of argumentation support in Semantic Web applications.

1.5 Argumentation support in Semantic Web applications: Research gaps

Current research on the logic layer of the Semantic Web uses rule-based languages to design and develop Semantic Web applications. However, the current developments have two major limitations as follows:

1. Different rule-based languages have been proposed for knowledge representation and reasoning in Semantic Web applications, however, each language has its own syntax and semantics leading to inter-operability issues among different reasoning engines. Therefore, the rules specified in a Semantic Web application can't be used in other ones. Similarly, the output of a reasoning engine can't be shared with other reasoning engines.

2. Like traditional applications, advance Semantic Web applications perform reasoning under certain assumptions that *'the underlying information for decision making is consistent and addition of new information doesn't result in contradictions with the existing information'*. In other words, they assume that:

   i) no conflicts arise during the process of decision-making,

   ii) and the introduction of new information will not result in achieving different outputs.

As a result of this assumption, they don't cater for other information that could be incomplete and/or contradictory, but may represent the correct facts, and if considered, may lead to a different result. In these scenarios, where contradictory information appears within enterprise boundaries, they either eradicate it or do not include this information in the decision-making processes.

The application of defeasible reasoning is seen as an important attempt to address the problem discussed above i.e. to represent and reason over incomplete and/or contradictory information. However, as pointed out in Section 1.3, in defeasible reasoning, the priorities are predefined and hard-coded in the application and assume no more possible conflicts will arise during the decision-making process. In contrast, the applications discussed in Section 1.2 are subjected to incomplete and/or contradictory information where conflicts arise at run time and this is not satisfiable with current approaches. As a result, the current enterprises cannot exploit the information on the WWW outside of their boundaries to assist the decision-making process. This calls for the design and development of intelligent Semantic Web applications that can transform incomplete and/or contradictory information into useful knowledge to assist the decision maker in the decision making process.

In an attempt to find inspiration for reasoning over incomplete and/or contradictory information where conflicts may arise at run time, the literature on argumentation in Philosophy and its exploitation in the field of Artificial Intelligence was studied in detail where identifying and resolving conflicts takes place during the process of argumentation itself. This presents the opportunity to realise similar benefits by equipping defeasible reasoning with the ability to conduct argumentative reasoning over contradictory interests to produce a conclusion.

It is evident from the above discussion that current Semantic Web development technologies defined at the logic layer of the Semantic Web do not provide any solution to represent reason and integrate information that could be incomplete and/or contradictory. Therefore, enterprises need a logic-based framework that can take into account incomplete and/or contradictory information on the WWW and transform it into useful knowledge that, in turn, assists the members of an enterprise in their decision making process to achieve BI.

1.6 Research objectives of the thesis

The objective of this research is to propose, develop and validate a Generic Framework for carrying out Argumentative reasoning in Semantic Web Applications (GF@SWA). In order to address the primary objective, the research objective can be broken down

into the following sub-objectives:

1. To propose a rule-based declarative language for incomplete and/or contradictory information representation and reasoning in Semantic Web applications. Such information representation enables Web users to provide their information i.e. specifications or preferences, that can be taken into account by Semantic Web applications, considered in the reasoning process and produce customized results for the decision maker.

2. To propose a methodology for an argumentation-driven reasoning engine to reason over incomplete and/or contradictory information by taking into account different conflict resolution algorithms to resolve conflicts between arguments. Additionally, to propose a methodology to display a justifiable explanation of conflict resolution and reasoning results to non-technical decision makers.

3. To propose a mechanism to integrate the information being produced by an argumentation-driven reasoning engine in the form of a reasoning chain and represent its graphical representation to the decision maker for a better understanding of the results. Additionally, to propose a mechanism to export reasoning chains to other software systems in order to integrate the reasoning chains produced by different information systems into a coherent reasoning chain. Such knowledge integration will provide a complete picture about information spanning across different Semantic Web applications.

4. Exploitation of GF@SWA in different Semantic Web applications as follows:

   (a) Design and develop a Web-based Intelligent DSS for representation and reasoning over incomplete and/or contradictory information to assist the decision maker in decision making process.

   (b) Design and develop a Web-based Intelligent DSS for enterprise knowledge integration.

   (c) Design and develop a Web-based Intelligent DSS for process map discovery from business policies.

5. Evaluate and validate the proposed framework and Semantic Web applications with the help of case studies and implementation.

This thesis addresses the research objectives outlined above, resulting in addition of significant knowledge to the existing body of literature.

1.7 Scope of the thesis

The aim of this thesis is to design and develop a logic-based framework to support argumentation in Semantic Web applications (GF@SWA). Defeasible logic programming (DeLP) is used for incomplete and/or contradictory information representation, reasoning and integration to support intelligent decision making. DeLP is extended in the following aspects:

1. syntax and semantics for data-driven or forward-chain reasoning;
2. syntax and semantics for goal-driven reasoning to resolve conflicts among arguments using different argumentation-driven conflict resolution strategies;
3. syntax and semantics for information and knowledge integration.

The utility and applicability of GR@SWA has been explained with the help of different Semantic Web applications for BI. Even though concepts are drawn from the philosophical view of argumentation and logic programming, this thesis does not claim to make a contribution to those areas.

1.8 Significance of the thesis

The significance of this thesis can be discussed under two broad sections: scientific significance and social significance.

1.8.1 Scientific significance

Currently, there is no argumentative reasoning engine for carrying out automated reasoning in the Semantic Web context. This research will contribute significantly to the existing body of knowledge for building a framework which adheres to Web standards that can perform argumentative reasoning as a standalone component in Semantic Web applications. On the basis of the literature review, it has been identified that non-monotonic reasoning and argumentation will have a significant impact on business applications (Kontopoulos et al., 2008) e.g.

1. Reasoning with incomplete and/or contradictory information
   Business rules often have to deal with incomplete information because other players may not be able (e.g. due to communication problems) or willing (e.g. because of privacy or security concerns) to provide complete information.

Additionally, the elicitation of business rules from different viewpoints may result in contradictory information during the decision-making process. This is a typical case for applying the logic-based framework to support argumentative reasoning to assist the decision-making process.

2. Reasoning in prediction systems
   Prediction systems specify their knowledge in the form of rules that can reason on some information and help the decision maker to identify suitable goods and services. The choices suggested by prediction systems need to be backed by explanations. This thesis will equip the prediction systems with argumentation support to generate explanations in the form of arguments and counter-argument against each suggested choice.

3. Generic framework for information processing, integration and exchange

   In an open computing environment, such as the WWW or an enterprise intranet, various decision support systems are expected to work together to support information exchange, processing, and integration. Currently, there is no generic framework that can process, integrate and exchange incomplete and/or contradictory information. This thesis contributes a significant body of knowledge for the development of a generic framework that can be exploited by Semantic Web applications for information information processing, integration and exchange on the WWW.

4. Reasoning chains for non-technical decision makers
   There is no framework that provides a visual representation of the reasoning process to non-technical decision makers, therefore this research paves the way to building a more interactive system for non-technical decision makers.

1.8.2 Social significance

A huge number of electronic business transactions are carried out on a daily basis in e-commerce applications. This research will enable or support such business entities to carry out decision making in situations where there is incomplete and/or contradictory information.

1. Reasoning over customer feedback in e-Commerce applications
   Due to the availability of semantic tools for the semantic enrichment of data, a number of attempts have been made to transform e-commerce data in the form of OWL/RDF. A typical e-commerce site contains product ontology, Web user's ontology (FOAF) and feedback ontology (SIOC). Reasoning on such semantically

linked data with the help of an argumentative reasoning engine will reveal a number of relationships between products and customer feedback. Such reasoning will be very helpful in improving products and the quality of service.

2. Ontology engineering, alignment and merging
   When ontologies and rules are developed by different authors and/or sources are merged, inconsistencies and contradictions arise naturally and argumentative reasoning could be used as an important mechanism to resolve these conflicts.

3. Trust establishment in e-Commerce applications
   Trust is the key element in commerce, both in traditional commerce and e-Commerce. If a customer (or Web user) wants to buy a device from a website, he has to negotiate with the website systems to automatically establish trust with the goal of successfully completing the transaction. This negotiation is based on the policies and the credentials each system has. The customer and website policies describe who they trust and for what purposes. Argumentative reasoning will provide a suitable solution to address the requirement of such e-Commerce transactions on the web.

1.9 Thesis plan

This thesis is structured into nine chapters as follows:

- Chapter 2 provides a critical survey of relevant existing research. In particular, the existing frameworks for argumentation in the fields of Philosophy, Artificial intelligence, the WWW and the Semantic Web are discussed and critiqued.
- In Chapter 3, the problem definition and research objectives are presented.
- In Chapter 4, the conceptual framework for incomplete and/or contradictory information representation, reasoning and integration is outlined.
- In Chapter 5, a conceptual framework for Argumentation-enabled Web-based intelligent DSS for incomplete and/or contradictory structured information representation, reasoning and integration (Web@IDSS) is developed.
- In Chapter 6, a conceptual framework for enterprise knowledge integration through Argumentation-enabled Web-based intelligent DSS (Web@KIDSS) is developed.

- In Chapter 7, a conceptual framework for Semantic Web applications to consider unstructured information that may be incomplete and/or contradictory is developed. To explain the working of the conceptual framework, a case study that takes into account the business polices of an enterprise for the generation of a business process map is considered. Argumentation-enabled Web-based Intelligent DSS that uses knowledge representation approach with argumentative reasoning for process map discover from unstructured business policies (KR@PMD) is developed.
- In Chapter 8, the evaluation and validation of the proposed framework is provided.
- In Chapter 9, the conclusion to the thesis is given and future research directions are provided.

The structure of the thesis is summarised in Figure 1.12. Chapters 5, 6 and 7 are all elaborations on the conceptual framework presented in Chapter 4.

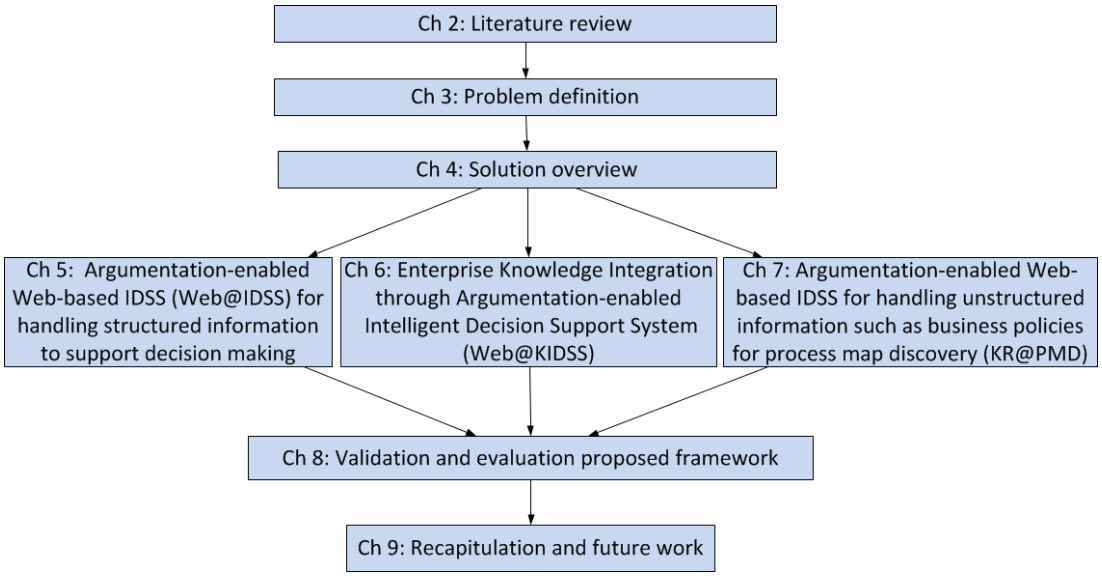


Figure 1.12: Outline of chapters

1.10 Conclusion

This chapter introduced the Semantic Web and discussed the ontology languages layer and logic layer in detail. The limitations of current development technologies on the logic layer which results in certain challenges for Semantic Web applications were outlined. The importance of defeasible reasoning and argumentation techniques as

suitable candidates to address the challenges faced by Semantic Web applications in the area of BI was detailed. Additionally, the objectives of undertaking this research were discussed, followed by a description of the scope and significance of this thesis in enabling argumentation support in Semantic Web Applications. Finally, the thesis plan was presented.

Chapter 2 - Literature Review

2.1 Introduction

In this chapter, a comprehensive review of the literature, focusing on two important aspects of the research problem, is presented. The first aspect (discussed in Section 2.3), focuses on the study of argumentation models, frameworks and applications in different areas of research. The objective of this study is to identify key elements of argumentation, its strengths and weakness and exploit them to address the challenges faced by Semantic Web applications. The second aspect (discussed in Section 2.7), focuses on the study and categorization of existing approaches for reasoning in Semantic Web applications. In Section 2.8, a critical evaluation of the existing literature is given and seven critical research issues that need attention are identified in order to provide a framework for argumentation support in Semantic Web applications.

2.2 Basic definitions

In this section, some important definitions are outlined in order to prepare the reader for a better understanding of the concepts discussed in this chapter.

2.2.1 Argumentation

Argumentation is defined as "a verbal and social activity of reason aimed at increasing (or decreasing) the acceptability of a controversial standpoint for the listener or reader, by putting forward a constellation of propositions intended to justify (or refute) the standpoint before a rational judge" . It is the field of study in which rhetoric, logic and dialectic meet Rahwan et al. (2007b).

2.2.2 Argumentation systems

The applications governed by the rules of argumentation are known as argumentation systems (Munoz and Botia, 2008) . Different argumentation models and frameworks have been used in applications to address issues in different domains of research, and all of them have important notions, such follows:

- the definition of argument;
- the notion of conflict between arguments;
- the notion of defeat;
- an argumentation semantics that selects acceptable (justified) arguments (possibly including an underlying logical language and a notion of logical consequence).

2.2.3 Argument, Rebuttal, Undercut and Acceptable arguments

Argumentation is inherently a process rather than an instant picture and the building blocks of argumentation are arguments and the relationships between those arguments. According to the definitions in the literature by (Walton, 2009; Palau and Moens, 2009; Besnard and Hunter, 2008), an argument is a set of statements made up of a minimum of three parts:

- a conclusion, also known as a claim, is a proposition which could be either true or false. These claims are used to drive other claims;
- a set of premises used to support the conclusion;
- inference or reasoning steps from premises to conclusion.

The support of an argument provides the reason (justification) for the claim of the argument. An argument can be supported by other arguments known as its sub-arguments. Counter-arguments or rebuttals are also arguments that attack an argument with a contradictory claim. Counter-arguments, in turn, may be defeated and the process may continue, resulting in the construction of argumentation lines (Garcia and Simari, 2004). An undercutting argument is an argument with a claim that contradicts some of the assumptions/inference of another argument.

For arguments to be acceptable, they must be weighed, compared and evaluated to identify the set of warrants and a conclusion which convinces all decision makers. An

acceptable set of arguments is coherent and strong enough to defend itself against any attacking argument.

2.2.4 Argumentation scheme

During the process of argumentation, relationships among arguments link them with one another in a certain pattern to support the ultimate conclusion. Such linking patterns are called Argumentation Schemes (Walton, 2005). A leading example of an argumentation scheme is that which represents the argument from expert opinion (Walton, 1997). Argument from expert opinion can be a reasonable argument if it meets the conditions displayed in the following argument form, where A is a proposition, E is an expert, and D is a domain of knowledge: E is an expert in domain D. E asserts that A is known to be true. A is within D. Therefore, A may plausibly be taken to be true.

2.2.5 Argumentation life cycle

According to Eemeren and Grootendorst (2004); Walton (2009), four tasks under the umbrella of argumentation are identification, analysis, evaluation and invention. The task of detection involves the construction of an argument and attaching it to an argumentation scheme, if possible. It involves the detection of a difference of opinion. In the analysis phase, the participant tries to find implicit premises and conclusions and tries to make them explicit to better evaluate the argument. Arguments missing some premises or, in some instances, a conclusion, are termed Enthymeme. In the evaluation phase, the strength of an argument is determined, i.e. either strong or weak, in accordance with the general criteria applicable to that argument. The last phase is invention, in which we try to construct new arguments that can be used to prove a specific conclusion.

2.2.6 Types of arguments

According to Walton (2006), three major types of argument are as follows:

- In a deductive argument (e.g. mathematical proof in propositional logic), if the premises are true, then the conclusion must be true. The reasoning process based on deductive arguments is known as deductive reasoning.
- An inductive argument involves a kind of generalization from the empirical evidence gathered. Inductive arguments sometimes use statistical techniques to

establish the strength (or confidence) of the supported claim. The reasoning based on inductive arguments is known as inductive reasoning.

- In a presumptive argument, the conclusions are said to be plausible given the premises. Plausibility is different from probability. While probability is determined by reasoning from statistical evidence, plausibility states that the conclusion holds by default provided no adequate evidence supports the contrary view. Arguments can be depicted graphically using argument diagramming techniques (Reed and Rowe, 2007).

2.2.7 Patterns of arguments

During the argumentation process, arguments can be arranged in three ways or patterns, called complex argumentation patterns, as discussed by Eemeren et al. (2002); Zarefsky (2009) and Reed et al. (2007).

- Subordinative argumentation: In this pattern of argumentation, arguments are arranged in a serial structure and depend on one another in a specific order to carry the resolution.
- Coordinative / linked argumentation: Arguments are arranged in a convergent structure. Each argument is independent of the others and the entire group of arguments must be carried out to carry the resolution.
- Multiple/ Parallel / Convergent argumentation: Each argument is independent of the others and each is sufficient to carry the resolution.

2.2.8 Monological and dialogical argumentation

According to Rotstein et al. (2010); Besnard and Hunter (2008), argumentation is monological if a single agent or entity has collated the knowledge to construct arguments for and against a particular conclusion. If a set of entities or agents interacts to construct arguments for and against a particular claim, then such argumentation is called dialogical argumentation. Newspaper articles, political speech, review articles, or problem analysis by an individual seeking to draw a conclusion are examples of monological argumentation, whereas, lawyers arguing in court, trader negotiations and debates on an issue are examples of dialogical argumentation .

2.2.9 Static and dynamic argumentation framework

The argumentation framework is considered to be dynamic if the knowledge-base from which the arguments are derived is dynamic, i.e. it can be changed during the argumentation process either with external changes or via guided changes. In a static argumentation framework, by contrast, a single set of evidences is used in the argumentation process, i.e. the knowledge-base does not change during the process of argumentation. As a result, only one instance of argumentation framework would exist (Rotstein et al., 2010).

2.3 Argumentation-based models, frameworks and applications

In Chapter 1, a general introduction to argumentation was given. In this section, it is elaborated in greater detail. The current literature on argumentation models, frameworks and applications can be divided into two broad categories:

1. Philosophical argumentation
   Models, frameworks and applications emphasising enrichment of the internal structure of an argument as in the work done by Toulmin (2003) are considered a philosophical model of argumentation,

2. Logic-based argumentation
   Frameworks and applications built on a logic-based argumentation framework are grouped under the umbrella of logical models of argumentation. The current frameworks and applications that exploit argumentation models for reasoning on the WWW are studied and compared.

2.4 Philosophical models of argumentation

"I see what your premises are, says the philosopher, and I see your conclusion. But I just don't see how you get there. I don't see the argument". These statements distinguish the notion of argument in philosophy from the technical notion of argument in logic by placing greater emphasis on the internal reasoning structure that leads the premises to a conclusion (Parsons, 1996). The history of argumentation in philosophy can be traced back to the beginnings of rhetoric in ancient Greece. Rhetoric is the

art of using language to communicate effectively. Citizens learned techniques to argue in court so that they could defend themselves. Aristotle carried out a systematic treatment of argumentation and rhetoric. Until the 1950s, argumentation was based on rhetoric and logic, but in 1958, Toulmin provided a logical structure of arguments and explained how the process works, using it as a tool to analyze various kinds of philosophically-problematic reasoning. Perelman (1969) tried to find a description of the techniques of argumentation used to obtain the approval of others for their opinions and called it 'new rhetoric'. Both Toulmin and Perelman tried to present an alternative to formal logic that was better suited to analyzing every day communication. Eemeren and Grootendorst (2004) studied argumentation as a means of resolving differences of opinion by considering argumentation as a discourse activity. They proposed pragma-dialectical theory which views argumentation as ideally being part of a critical discussion which progresses through four discussion stages to resolve a difference of opinion: the confrontation stage, opening stage, argumentation stage and concluding stage.
In this chapter, the existing literature on argumentation, based on philosophical concepts, is grouped into one of the following two categories:

1. Theoretical models of argumentation.

2. Argumentation frameworks and applications.

In the following section, each category is discussed in detail.

2.4.1 Theoretical models of argumentation

2.4.1.1 Toulmin's model and its extensions

Toulmin, a British philosopher, pointed out that formal logic relies on the rigorous testing of arguments based on mathematical rules carried out to declare them either valid or invalid, which is of very little practical value (Toulmin, 2003; Freeley and Steinberg, 2008). He proposed a model to better understand the structure of practical reasoning that occurs in any argument. He believed that reasoning is much more closely associated with the activity of testing and shifting existing ideas through the process of justification, rather than using inference to discover new ideas. He categorized premises in such a way that an argument was provided with a richer structure, one which corresponds more closely to the way in which arguments are presented.

He distinguished six parts in argument structure and presented these in a diagrammatic representation, as depicted in Figure 2.1. The elements are:

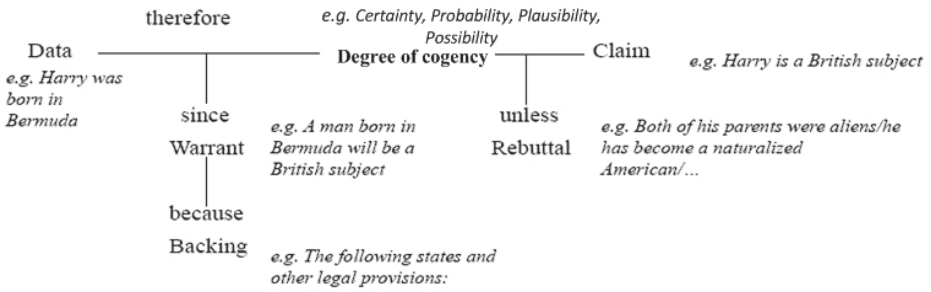


Figure 2.1: An illustration of Toulmin's model of argument structure (Toulmin, 2003)

1. Claims: Every argument makes assertions based on data. The assertion of an argument is the claim of the argument.

2. Grounds: Data and hard facts, plus the reasoning behind the claim to establish the foundation of the claim.

3. Warrants: Evidence and reasoning to justify the move from grounds to claim. Warrants are not self-validating.

4. Backing: The backing (or support) for an argument gives additional support to the warrant by answering different questions.

5. Modal qualification or degree of cogency: Qualifying the claim to express the degree of cogency or modal specification. This is the extent to which the argument is both sound and intellectually compelling. Toulmin used modal qualification to express the concept of degree of cogency. The degrees of cogency are certainty, probability, plausibility or possibility.

6. Rebuttals or Counter-arguments: Any rebuttal is an argument in itself, and thus may include a claim, warrant, backing and so on. It also can have a rebuttal.

According to Baroni et al. (1998), Toulmin's conceptual model of argumentation can help to classify the various ways an argument can be analyzed. According to Toulmin, three possible strategies are:

1. If the initial data of the opponent is wrong, all the conclusions derived from that data will be undermined.

2. If there is a flaw in the line of reasoning that relates data to the conclusion, i.e. warrant, this might mean questioning the knowledge used in the current context or questioning the inference rules.

3. If inconsistencies can be detected in the opponent's background knowledge, challenge the backing.

Table 2.1 summarizes the different extensions made to Toulmin's model of argument representation. Each of these extensions is made keeping in view the purpose to be fulfilled in a specific domain, as illustrated in the table.

2.4.1.2 Argumentation schemes proposed by Walton and Reed

Argumentation schemes provide a way to perform reasoning over a set of premises and a conclusion. These argumentation schemes have emerged from informal logic and help to categorize the way arguments are built, aiming to fill the gap between logic-based application and human reasoning by providing schemes which capture stereotypical patterns of human reasoning, e.g., arguments from an expert opinion scheme. Formally, an argumentation scheme is composed of a set of premises A_i, a conclusion C, and a set of critical questions CQ_i with the aim of defeating the derivation of the consequences (Rahwan et al., 2007a; Letia and Groza, 2008).
The aim of an argument in presumptive or plausible reasoning is to shift the burden of proof in a dialogue. Blair (1999) describes and discusses approximately thirty such argumentation schemes. For each scheme, he provides a description, a formulation, a set of associated critical questions, at least one and often several cases which are actual or invented examples of the scheme in use, and a discussion of the scheme, in which he typically draws attention to its salient properties, relates it to other schemes, discusses the fallacies associated with it, comments on its presumptive force, and mentions typical contexts of its use. Fallacies are the violations of rules of critical discussion that hinder the resolution of opinion. Blair listed six characteristics of fallacy as follows: (1) dialectical (2) pragmatic (3) commitment-based (4) presumptive (5) pluralistic and (6) functional. These characteristics will help in the identification, classification and evaluation of fallacies. Subsequently, Walton (2005) tried to address the justification of a certain scheme.

Table 2.1: Extension to Toulmin's model of argument structure

	Extension	Purpose	Domain
Bench-Capon (1989)	Addition of a pre-supposition component	Introduction of assumptions to support the claim	Legal expert system
Branting (1993)	Warrant extension	Construction of warrants hierarchy showing different levels of abstraction	Legal expert system
Freeman (1993)	Warrant extension	Classifications of warrant to enrich the justification structure between assertion and data	Legal expert system
Newman and Marshall (1992)	Extended entire structure	Mapping of argumentation schemes, argument structure to argumentative discourse	Legal expert system
Clark (1991)	New approach to knowledge representation and problem-solving based on Toulmin's model	Compare different opinions for risk assessment	Geological risk assessment
Zeleznikow and Stranieri (1995)	New approach to justified reasoning in rule-based systems and neural networks	Predict the outcome of property disputes in the domain of Australian family law.	Legal expert system

Reed and Walton (2003) also showed that argumentation schemes help users to identify and evaluate common types of argumentation in daily discourse, but the ways in which argumentation schemes drive a dialogue onwards, through a combination of critical questioning and relevance maintenance, is largely unaddressed. Therefore, the authors explored the relationship between the argument-as-process and argument-as-product representations, using, as a focus, the roles that argumentation schemes play in the two approaches. Arguments found in text are considered to be products because they are already there, and when the argument is used to fill the unstated premises or conclusions, the task is seen as argument-as-process. To understand this notion, suppose that Bob and Helen are having a critical discussion on tipping, and that Helen is against tipping. She thinks that tipping is a bad practice that ought to be discontinued. Suppose in this context, Helen puts forward the following argument: *Dr. Phil says that tipping lowers self-esteem.*
Dr. Phil is an expert psychologist, so the argument is, at least implicitly, an appeal to expert opinion. It is also, evidently, an instance of argument from consequences. Helen is telling her opponent, Bob, that lowering self-esteem is a bad consequence of an action. Her argument is based on the assumption that since this bad outcome is a consequence of tipping, tipping itself is a bad thing. Thus, Helen's argument is an enthymeme, that is, it is a chain of argumentation that can be reconstructed as follows:
The self-esteem argument:

- Dr. Phil says that tipping lowers self-esteem. Dr. Phil is an expert in psychology, a field that has knowledge about self-esteem.
- Tipping lowers self-esteem.
- Lowering self-esteem is a bad thing.
- Anything that leads to bad consequences is itself bad as a practice.
- Tipping is a bad practice.

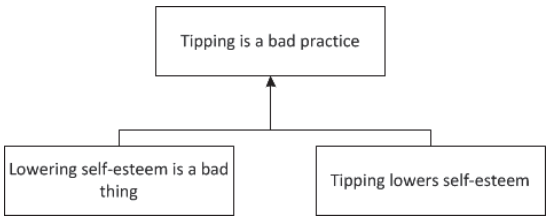


Figure 2.2: Illustration of the self-esteem argument

How can one know this? How can one fill in the unstated premises and link them with other premises and conclusions in a chain of argumentation that represents Helen's

line of argument? One tool which is needed is the argumentation scheme. Figure 2.2 illustrates the self-esteem argument in which the argumentation scheme is used to reach a conclusion.

2.4.2 Argumentation frameworks and applications

2.4.2.1 Zeno argumentation framework

The Zeno argumentation framework (Gordon and Karacapilidis, 1997) is a formal model of argumentation based on the informal models of Toulmin's and Rittel's Issue-Based Information Systems (IBIS). The Zeno model contains the argumentation elements: issue, position, pro-argument, contra-argument, preference, decision, and comment, as illustrated in Figure 2.3. A message in the Zeno discussion forum (mediation system) may contain more than one such argumentation element, if the author expresses complex information in a single contribution. Most contributions in a forum will arise as replies to existing arguments, so that an argumentation tree develops. Zeno uses five standards of proof:

1. Scintilla of Evidence: the choice has some pros.
2. Preponderance of Evidence: the pros outweigh the cons given the preference constraints.
3. No Better Alternative: no choice is preferred on the basis of the preference constraints.
4. Best Choice : one choice is preferred to every alternative choice on the basis of the preference constraints.
5. Beyond Reasonable Doubt: no con reason against a particular choice, and no pro reason for an alternative.

2.4.2.2 Carneades argumentation Framework

The Carneades argumentation framework (Gordon and Walton, 2006; Gordon et al., 2007) is a formal, mathematical model of argument evaluation which applies proof standards to determine the defensibility of arguments and the acceptability of statements on an issue-by-issue basis. It carries features from both the Zeno framework and argumentation schemes. The framework use three kinds of premises (ordinary premises, presumptions and exceptions) and information about the dialectical status

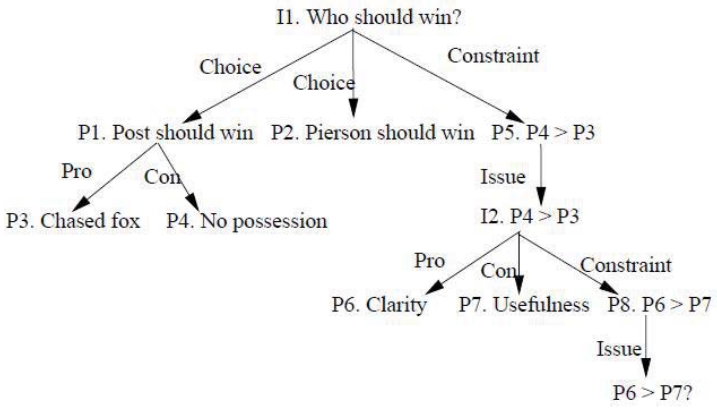


Figure 2.3: Zeno argumentation model

of statements (undisputed, at issue, accepted or rejected) to model critical questions in such a way as to allow the burden of proof to be allocated to the proponent or the respondent, as appropriate. The proof standards of Carneades are:

1. Scintilla of Evidence: supported by at least one defensible pro argument.

2. Preponderance of Evidence: the strongest defensible pro argument outweighs the strongest defensible con argument, if there is one.

3. Dialectical Validity: supported by at least one defensible pro argument, and none of the con arguments are defensible.

4. Beyond Reasonable Doubt: supported by at least one defensible pro argument; all of the pro arguments are defensible and none of the con arguments are defensible.

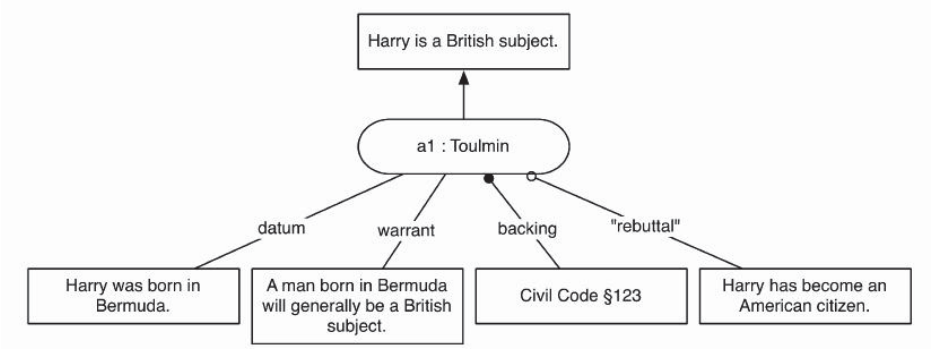


Figure 2.4: Carneades argumentation model

Figure 2.4 is a reconstruction of Toulmin's standard example about British citizenship in the Carneades framework. The 'rebuttal' is modeled as an exception and the backing as an assumption. Both the datum and warrant are ordinary premises.

Alternatively, backing could be modeled as the premise of an additional argument pro the warrant, by generalizing the concept of an argumentation scheme to cover patterns with multiple arguments. Carneades [1] provides tools that support a variety of argumentation tasks, including:

1. argument mapping and visualization;
2. argument evaluation, applying proof standards and respecting the distribution of the burden of proof;
3. argument construction from OWL ontologies and defeasible rules;
4. argument interchange in XML, using the Legal Knowledge Interchange Format (LKIF).

2.4.2.3 Sense-Making tool : Araucaria

Argument diagramming is often claimed to be a powerful method for analysing and evaluating arguments (Reed and Rowe, 2007). Ongoing work is being conducted on building software for the analysis of arguments, resulting in the development of software for specific groups of users with particular needs, leading to a plethora of such tools. As a result, there is a need for a tool that can support different theoretical approaches to analyze arguments.

The Araucaria [2] tool aims to do this. The Araucaria system (Reed and Rowe, 2004) has been used to mark up and diagram textual arguments, supporting analysts' work in reconstruction and identification. Araucaria is a freely available, open source software package which allows the text of an argument to be loaded from an Argument Markup Language (AML) file, and provides numerous tools for marking up this text and producing Standard, Toulmin, and Wigmore diagrams. Araucaria supports different styles of argumentation, as well as translation features from one style to another. It is currently being used in the construction of an online repository of arguments drawn from newspaper editorials, parliamentary reports and judicial summaries from around the world. Online Visualization of Argument (OVA) [3] is the web-based version of Araucaria.

It is evident from discussion on the philosophical models of argumentation that it plays a pivotal role as reasoning methodology in field of Philosophy. In the next subsection , the logic-based models of argumentation and their exploitation in the field of Artificial Intelligence is discussed.

[1] http://carneades.berlios.de/
[2] http://araucaria.computing.dundee.ac.uk/doku.php
[3] http://www.arg.dundee.ac.uk/?page_id=143

2.5 Logic-based models of argumentation and applications

Traditional models of reasoning were monotonic and unable to cope with incomplete, uncertain and dynamic information. These reasoning models are built on first-order predicate logic or a subset of the same and perform reasoning under certain assumptions such as:

1. the given problem is fully specified (the solution to the problem lies in the specified information);
2. the specifications are consistent;
3. new facts are also consistent with the already specified specifications;
4. new facts do not lead to the retraction of previous conclusions.

If $\mathcal{T}$, $\mathcal{F}$ and $\mathcal{G}$ represent some statements, then, monotonic reasoning can be expressed formally as follows:

$$\mathcal{T} \models \mathcal{F} \rightarrow \mathcal{T} \sqcup \mathcal{G} \models \mathcal{F}. \tag{2.1}$$

It is evident from equation 1 that if a set of axioms in monotonic reasoning is enlarged, existing assertions or axioms cannot be retracted. Such reasoning does not add to our knowledge base and merely rearranges it (Nute, 1994). This is a basic property which makes sense for mathematical knowledge but is not desirable for knowledge representation, in general.

In the 1970s, argumentation was considered to be another way to formalise defeasible reasoning or non-monotonic reasoning because of its close resemblance to human patterns of reasoning, indicating that argumentation is the way in which a person takes a standpoint and defends this standpoint. It is much more related to day-to-day argumentation than the reasoning of logicians, who tend to concentrate on the way in which conclusions are derived from premises (Eemeren et al., 1996). Some examples are as follows:

1. A well-known example from Artificial intelligence is as follows:

   Argument : Tweety flies because Tweety is a bird Counter-argument : Tweety is different therefore it does not fly

2. In epistemology, the standard example is

   Argument : This looks red, therefore it is red. Counter-argument: But the ambient light is red, therefore it is not red

In recent years, argumentation has gained considerable attention from the artificial intelligence research community which has led to the investigation of argumentation and its applications in various domains. From its theoretical foundations, argumentation can be integrated into a number of real world applications, such as planning, MAS, legal reasoning, knowledge engineering, the analysis of news reports, clustering, argumentation support systems, mediation systems and computer-supported collaborated argumentation (Chesnevar et al., 2006b).
In the field of AI, researchers are not particularly interested in the internal structure of an argument. In contrast, they consider an argument to be a single entity and hence are much more interested in modeling and evaluating the relationships between arguments to reach a conclusion. In this section, I broadly divide the current literature into two categories as follows:

1. Argumentation frameworks.
2. Argumentation systems or applications.

2.5.1 Argumentation frameworks

Broadly speaking, I can divide argumentation frameworks into the following five categories:

1. Abstract argumentation framework.
2. Bipolar argumentation framework.
3. Preference-based argumentation framework.
4. Value-based argumentation framework.
5. Assumption-based argumentation framework.

2.5.1.1 Abstract argumentation framework

Dung (1995) proposed a very influential semantic foundation for an argumentative framework based on the notion of the acceptability of arguments. He defined the characteristics of the argumentative framework according to the relationship between arguments and between sets of arguments. He defined an argumentation framework, emerging from logic programming, as a pair $AF =< \mathcal{A}, attack >$ where $\mathcal{A}$ is the set of arguments and *attack* is the binary relation on AR, representing the conflict between them. If $(A, B) \in attack$ then argument A attacks and defeats argument

B . The notion of *defence* is defined from the notion of defeat by: an argument A_i defends A_j against B iff there exist $((B, A_j) \wedge (A_i, B)) \in attack$. He defined certain properties of the argumentation framework which help to categorize arguments into different extensions, such as preferred, stable and ground extensions. These properties are:

1. Conflict Free : Given an AF $F = (\mathcal{A}, attacks)$. A set $S \subseteq \mathcal{A}$ is conflict-free in F, if, for each $a, b \in S$, $(a, b) \notin attacks$.

2. Admissible Set : Given an AF $F = (\mathcal{A}, attacks)$. A set S$\subseteq \mathcal{A}$ is admissible in F, if

   (a) S is conflict-free in F

   (b) $a \in \mathcal{A}$ is defended by S in F, if for each $b \in A$ with $(b, a) \in attacks$, there exists a $c \in S$, such that $(c, b) \in attacks$.

The framework abstracts the details of the underlying language, argument structure, origin and nature of arguments and argumentation rules. The presented semantics, therefore, are clearer and more precise and as a result, the relationship between the arguments can be analyzed in isolation from other relationships (e.g. implications). Additionally, this framework encompasses a large variety of specific formalisms, such as non-monotonic reasoning and game theory; as a result, it can be regarded as a powerful tool for comparing different systems. Although his work elaborated in detail the semantics of the argumentation network, Dung took an argument as an atomic entity, and his notion of attack is also weak, because it considered all arguments to be of the same strength. If an argumentation framework contains no even cycles, the dispute is resolvable and this resolution can be achieved in a time linear to the number of attacks. The framework also assumes that a complete set of arguments is given together with the set of conflicts between arguments, and focuses on the definition of the status of an argument. Argumentation semantics define the properties required for a set of arguments to be acceptable. A set of arguments exhibiting these properties is called an extension of the argumentation framework, for example:

- **Admissible semantics** A set $E \subseteq A$ is admissible if and only if E is conflict-free and E defends all its elements.

- **Preferred semantics** A set $E \subseteq A$ is a preferred extension if and only if E is maximal for set inclusion among the admissible sets.

- **Stable semantics** A set $E \subseteq A$ is a stable extension if and only if E is conflict-free and every a $\in$ A , E is attacked by an element of E.

- **Grounded-semantics** The grounded extension of <A,R>is the smallest subset of A with respect to set inclusion among the subsets of A which are admissible and coincide with the set of arguments acceptable w.r.t. itself.

Table 2.2 presents the syntax used to represent arguments, the set of arguments and the relationships between the arguments. Table 3 gives a comparison of abstract argumentation frameworks on the basis of the notion of attack, argument acceptability criteria, extension and miscellaneous features. The notion of attack is defined as a tuple of the following form:

$$\{(argument\ OR\ set\ of\ arguments), (argument\ OR\ set\ of\ counter-argument)$$

$$, nature\ of\ attack, set\ of\ constraints\}$$

Table 2.2: Symbols with their respective description

Symbol	Description
A, B ,C	Individual Arguments
$\mathcal{S}$,$\mathcal{Y}$	Set of arguments
$\longrightarrow$	Direct attack or Direct support
$\not\mapsto$	Indirect attack or Indirect support
$\circlearrowleft$	Recursive attack
$\rightrightarrows$	Set of attack (sequence of attack)
$_C \succ_B$	Left argument has higher priority than argument on right
$\approx$	No preference
$\mathcal{C}$	Set of constraints (Propositional formula)

As proposed by Dung (1995), in Table 2.3, the notion of attack is represented as (A, B,→), where A is an argument and B is its counter-argument and there is a direct attack between A and B. Similarly, ($\mathcal{S}, \mathcal{Y}$,→) represents a direct attack between the set of arguments $\mathcal{S}$ and the set of arguments $\mathcal{Y}$. Similarly, ($\mathcal{S}$, (A,B) | A, $\circlearrowleft$) represents an attack between the set of arguments $\mathcal{S}$ and an argument A and indicates that there is also a recursive attack between argument A and its counter-argument B . It is evident from Table 2.3 that the researcher built this argumentation framework on top of Dung's framework by adding different flavours of attack, whereas the acceptability criteria and extensions are quite consistent. Each of these frameworks will be discussed briefly below.

Table 2.3: Comparison of abstract argumentation frameworks

	Notion of Attack	Acceptability criteria	Feature	Extensions [4]
Dung (1995)	(A, B,$\rightarrow$) $\in$ *attack*	Conflict free and admissible set	All arguments are of same strength. Disputes are not resolvable with even cycles	Preferred, Stable, Grounded.
Bochman (2003)	($\mathcal{S}, \mathcal{Y}$,$\rightarrow$) $\in$ *attack*	Consistent and admissible set	Collective argumentation to represent semantics of disjunctive logic programs.	Preferred, Stable, Grounded extension.
Nielsen and Parsons (2007)	($\mathcal{S}$, A, $\rightarrow$) $\in$ *attack*	Acceptable and admissible set	Synergy among arguments: Strong attack	Preferred, Stable, Grounded, Complete.
Coste-Marquis et al. (2005)	(A,B,$\nrightarrow$) $\in$ *attack*	Conflict free and admissible set	Provide prudent semantics i.e. two arguments having indirect attack will not belong to same extension	Preferred, Stable, Ground, Complete.
Coste-Marquis et al. (2006)	(A,B, $\rightarrow$, $\mathcal{C}$) $\in$ *attack*	Conflict free and admissible set	Constraint argumentation, Constraints over arguments	Preferred p-extensions, Stable p-extension, Preferred c-extension, Stable -c extension
Baroni et al. (2009)	($\mathcal{S}$, (A,B) | A,$\circlearrowleft$) $\in$ *attack* where (A, B)$\in$ *attack*	Defeat, Conflict free	Recursive attacks with out restrictions	Preferred

[4]For definition and description of each extension, readers are referred to corresponding article of the authors.

Bochman (2003) extended Dung's work by the direct representation of global conflicts between sets of arguments, whereas Nielsen and Parsons (2007) introduced the notion of joint attacks in which a set of arguments can attack other arguments. Katie Atkinson (2008) analyzed the two computational models of argumentation, i.e. the Abstract Argumentation Framework (AAF) and argumentation schemes. The AAF is the best framework to use when completely identified sets of arguments are available and a binary relationship exists between them. Very often, however, such a set is not available, in which case argumentation schemes can help to identify the ways in which arguments can be attacked or defended and assist in the evaluation of arguments with respect to a certain context. On the resolution of contextual issues, arguments can be abstracted to an argumentation framework and evaluation can be carried out with respect to logical relations between arguments. The author proposed an abstract argumentation scheme framework that represents the components of argumentation schemes in an argumentation framework. As a result, the structure of schemes is used to guide the dialogue and provide contextual elements of evaluation, whilst retaining the desirable properties of abstract frameworks to enable evaluation with respect to the logical relations between arguments.

Coste-Marquis et al. (2005) extended Dung's framework with prudent semantics to better handle controversial arguments. Under prudent semantics, no two arguments belong to the same extension if one of them indirectly attacks the other. Coste-Marquis et al. (2006) also extended Dung's framework to take into account several additional constraints on the admissible sets of arguments, expressed as a propositional formula over the set of arguments, called the constrained argumentation framework. All the frameworks discussed above are static argumentation frameworks. Cayrol et al. (2008) overcame the limitations of Dung's framework by introducing the dynamic argumentation framework and studied the impact of the addition of a new argument which interacts with one previous argument on a set of framework extensions.

2.5.1.2 Bipolar argumentation frameworks

Most argumentation systems define only one type of relationship between arguments i.e. an attack/defeat relationship. However, different studies reveal that another type of relationship may exist between arguments: the "support" relationship. Such an argumentation framework is called a bipolar argumentation framework (BAF) and is defined as $\langle \mathcal{A}, attacks_{def}, attacks_{sup} \rangle$ where $\mathcal{A}$ is a set of arguments, a binary relation $attack_{def}$ on $\mathcal{A}$ is called a defeat relation and another binary relation $attacks_{sup}$on $\mathcal{A}$ is called a support relation.

Amgoud et al. (2008) provided a comprehensive survey on the use of bipolarity in argumentation frameworks and elaborated its importance in argumentation processes in real world applications. Cayrol and Lagasquie-Schiex (2009, 2010) discussed bipolarity at the interaction level in the argumentation process. They defined the meta-argumentation framework and introduced the concept of coalition in BAF, based on the coherence of the admissible set. The arguments in coalition cannot be used separately in the attack process. Oren et al. (2007) describe the evidential bipolar argumentation framework that supports argument schemes, burden of proof, and accrual of argument. Table 2.4 provides a comparison of different bipolar argumentation systems. Most of these bipolar argumentation frameworks consider different types of attacks such as direct and indirect. Some BIFs consider joint attacks between arguments, whereas others also consider attacks on an argument by a set of arguments. The BAF provides a more enriched structure for argument representation, such as coalitions.

2.5.1.3 Preference-based argumentation frameworks

In argumentation frameworks, one argument may be preferred over another when it is more specific or has a higher probability or certainty. Such an argumentation framework is called a preference-based argumentation framework. It is defined as a triplet $\langle \mathcal{A}, attacks, \succeq \rangle$ where $\mathcal{A}$ is set of arguments, *attacks* is the binary attack relation defined on $\mathcal{A}X\mathcal{A}$ and $\succeq$ is a (total or partial) pre-order (preference relation) defined on $\mathcal{A}\mathcal{X}\mathcal{A}$.

Table 2.5 provides a comparison of preference-based argumentation systems. Modgil (2009) extended Dung's theory of argumentation to integrate meta-level argumentation about preferences between arguments to add more semantics to attack relationships between arguments. The result of an attack of one argument on another argument depends on the existence of a preference argument, stating the preference of the attacking argument on the attacked argument. The preferences between arguments are not predefined; instead, arguments claim them.

Table 2.4: Comparison of bipolar argumentation frameworks

	Notion of Attack	Acceptability criteria	Extra notes	Extensions [5]
Amgoud et al. (2008)	(A,B, $\rightarrow \mid \not\rightarrow$) $\in attack_{support}$, (A,C, $\rightarrow \mid \not\rightarrow$)$\in attack_{defeat}$	Conflict free	Acceptable argument belongs to ground extension. Rejected argument is attacked by acceptable argument	Acceptable arguments Rejected arguments Abeyance arguments
Cayrol and Lagasquie-Schiex (2009)	(A, B, $\rightarrow \mid \rightrightarrows$) $\in attack_{set-support}$, (A,C, $\rightarrow \mid \rightrightarrows$)$\in attack_{set-defeat}$	[6] Conflict free (Safe)	Coherence of admissible set	Admissible extension, Preferable extensions
Cayrol and Lagasquie-Schiex (2010)	(A, B, $\rightarrow \mid \rightrightarrows$) $\in attack_{set-support}$, (A,C, $\rightarrow \mid \rightrightarrows$)$\in attack_{set-defeat}$	Coalition	Metaargumentation using coalition concept	Coalition-preferred, Coalition-stable
Oren et al. (2007)	($\mathcal{S}$,A, $\rightarrow$)$\in attack_{evidence-support}$ ($\mathcal{S}$,A, $\rightarrow$)$\in attack_{evidence-Defeat}$	$^{+}$Conflict free, admissible set	Evidential Argumentation Systems. Framework support argument schemes, burden of proof, accrual of argument	Preferred extensions, stable extensions, grounded extensions

[5]For definition and description of each extension, readers are referred to corresponding article of the authors.

[6]Conflict free : This notion means that for set to be conflict free, argumentation system consider more conflicts than conflicts considered in notion of conflict-free by Dung.

Table 2.5: Comparison of preference-based argumentation frameworks

	Notion of Attack	Acceptability criteria	Extra notes	Extensions[7]
Modgil (2009)	(P, (B,C), $\rightarrow$) $\in$ attack	Conflict free	Argument are used to represent preferences. Meta-level argumentation-based reasoning about preferences (explicit argument used to define preference)	Admissible, Preferred, Stable and Ground extensions
Amgoud and Cayrol (2002)	((A, B), $_A \succ_B$) $\in attack$	Individual acceptability, joint acceptability	Partial or complete pre-ordering of argument to define preference	Acceptable arguments Rejected arguments Abeyance arguments
Martinez et al. (2006)	((A, B), $_A \succ_B$) $\in attack_{Properdefeatar}$,((A, B), $_A \succ_B$) and ((B, A), $_B \succ_A$) $\in attack_{Blockingdefeatar}$	Defeat-free or suppressed	Defined two types of defeat in pre-AF. Proper defeater (symmetric blocking defeater(symmetric)	Accepted arguments Non-accepted arguments rejected arguments
Martinez et al. (2008)	((A,B)$\wedge$(B,C), $_A \succ_B$) $\in attack_{Strongdefender}$, ((A,B)$\wedge$(B,C), $_B \succ_A$) $\in attack_{Weakdefender}$, ((A,B)$\wedge$(B,C), $_A \approx_B$) $\in attack_{Normalefender}$, ((A,B)$\wedge$(B,C), $NoPref$) $\in attack_{UnqualifiedDefender}$,	Admissible set	Attacks are ordered by their force	Studied admissible sets according to quality of defenders

[7]For definition and description of each extension, readers are referred to corresponding article of the authors.

Amgoud and Cayrol (2002) address the acceptability of arguments in preference-based argumentation frameworks, proposing a proof theory for this preference-based argumentation framework. The proof theory verifies whether a given argument A is acceptable or not. The proof theory is presented as a dialogue tree between two players, PRO and OPP. Martinez et al. (2006) extended the notion of defeat in an argumentation framework. Depending on the outcome of the preference relation, an argument may be a proper defeater or a blocking defeater of another argument. Martinez et al. (2008) equipped the argumentation framework with a set of abstract attack relations of varied strength, such as strong defender, weak defender, normal defender and unqualified defenders.

2.5.1.4 Value-base argumentation framework

In preference-based argumentation frameworks, it is not always possible to absolutely define the preference for an argument over its counter-argument, especially in practical reasoning, such as in law, politics and ethics. To address problems in such domains, a value-based argumentation framework has been proposed. A value-based argumentation framework (VAF) is a 5-tuple: $VAF =< \mathcal{A}, attacks, \mathcal{V}, val, valpref >$ where $\mathcal{A}$ is the set of arguments, $attacks$ is the binary attack relation defined on $\mathcal{A}\mathcal{X}\mathcal{A}$, $\mathcal{V}$ is a non-empty set of values, val is a function which maps from elements of $\mathcal{A}$ to elements of $\mathcal{V}$, and $valpref$ is a preference relation on $\mathcal{V}\ X\ \mathcal{V}$.

Table 2.6 provides a comparison of different preference-based argumentation systems. Bench-Capon (2003) proposed a value-based argumentation framework to quantify the strength of arguments and discussed the possibility of persuasion in the face of uncertainty and disagreement. He argued that persuasion is pivotal in argumentation, and that the strength of an argument depends on the social values it advances; the success of one argument over another depends on the strength of the values advanced by the argument concerned. Haenni (2009) presented a formal theory of probabilistic argumentation to handle uncertain premises for which respective probabilities are known. Probability is used to measure the credibility (weight) of possible arguments and counter-arguments; thereafter, the overall probabilistic judgment of the uncertain proposition in question is carried out to reach a certain conclusion.

Table 2.6: Comparison of value-based argumentation frameworks

	Notion of Attack	Acceptability criteria	Extra notes	Extensions [8]
Bench-Capon (2003)	((A,B), $valpref$(val(A),val(B)) $\in$ attack,	Conflict free, admissible for audience	Sceptically acceptable arguments are a good source for persuasion	Objective acceptance, Subjective acceptance, Indefensible
Haenni (2009)	$\Phi \mid s \models \perp$ then s is in conflict $\Phi \mid s$ represent conditional KB	Conflict free, admissible, Probabilistic criteria	Reasoning process consists of qualitative and quantitative parts.	No extensions

[8]For definition and description of each extension, readers are referred to corresponding article of the authors.

2.5.1.5 Assumption-based argumentation framework

Assumption-based argumentation addresses the issues of how to find arguments, identify attacks, and exploit premises shared by different arguments. Formally, an assumption-based argumentation is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \mathcal{M} \rangle$ where

- $\langle \mathcal{L}, \mathcal{R} \rangle$ is a deductive system, with a language $\mathcal{L}$ and a set of inference rules $\mathcal{R}$
- $\mathcal{A} \subseteq \mathcal{L}$is a (non-empty) set, whose elements are referred to as *assumptions*
- $\mathcal{M}$ is a total mapping from $\mathcal{A}$ into $\mathcal{L}$, where $\neg\alpha$ is the contrary of α .

In an assumption-based argumentation framework (ABF), arguments are deductions supported by assumptions. Bondarenko et al. (1993) presented an ABF in which a sentence is a non-monotonic consequence of a theory if it can be derived monotonically from a theory extended by means of acceptable assumptions. The notion of acceptability for such assumptions is formulated in terms of their ability to successfully "counter-attack" any "attacking" set of assumptions. The authors investigated applications of the proposed framework to logic programming, abductive logic programming, logic programs extended with classical" negation, default logic, autoepistemic logic and non-monotonic modal logic. Dung et al. (2009a) provided a review of ABF and stated that ABF makes use of under-cutting as the only way in which one argument attacks another argument.

Table 2.7 presents a comparison of two assumption-based argumentation frameworks.

Table 2.7: Comparison of assumption-based argumentation frameworks

	Notion of Attack	Acceptability criteria	Extra notes	Extensions [9]
Bondarenko et al. (1993)	If (A,$a_{assumption}$)$\vdash\alpha$ and $\{\alpha,b_{assumption}\}\vdash\perp$ then $(\alpha,b_{assumption}) \in attack$	Logical , admissible set	Provides new semantics for negation as failure in logic programming	Weakly admissible, Weakly preferred extensions
Dung et al. (2009a)	$(\mathcal{S},\mathcal{Y})\in attack$ if ($\alpha \in \mathcal{S},\beta\in \mathcal{Y})\in attack_{undercut}$	Acceptable dispute trees	Provides review of different Assu-AF . Dispute trees	Approximation (Dispute tree computation)

[9]For definition and description of each extension, readers are referred to corresponding article of the authors.

2.5.2 Argumentation Systems

2.5.2.1 Abstract argumentation system

Vreeswijk (1997) defined an abstract argumentation system that is capable of dealing with a number of problems of defeasible reasoning. He defined an abstract argumentation system as a triple$(\mathcal{L}, R, \preceq)$ composed of a language $\mathcal{L}$ that has the capability of negation in the head of a rule to present contradiction, a set of inference rules R, and $\preceq$, a relationship between arguments. He called this argumentation system a collection of defeasible proofs, or arguments of varying conclusive force. Although he assumed a predefined order among the rules, he also pointed out that conclusive force is not determined solely by syntactical structure; rather, further information is needed from the semantics of the discourse domain to establish whether one argument is stronger than another. He identified two types of rules, strict rules and defeasible rules. Rules can be chained together to form arguments.

2.5.2.2 Defeasible Logic Programming (DeLP) server

The DeLP server, proposed by Garcia and Simari (2004), is based on Defeasible Logic Programming (DeLP), which is a general-purpose defeasible argumentation formalism based on logic programming, and is intended to model inconsistent and potentially contradictory knowledge. A defeasible logic program has the form $\psi= (\Pi, \Delta)$, where Π and Δ stand for strict knowledge and defeasible knowledge, respectively. The set Π involves strict rules of the form $P \leftarrow Q_1 \ldots Q_n$ and facts (strict rules with empty body), and is assumed to be non-contradictory (i.e., no complementary literals P and $\sim P$ can be inferred, where $\sim P$ denotes the contrary of P). The set D involves defeasible rules of the form $P \leftarrowtail Q_1 ..... Q_n$ which stand for $\imath Q_1 \ldots Q_n$ provide a tentative reason to "believe P". Rules in DeLP are defined in terms of literals. A literal is an atom A or the strict negation ($\sim A$) of an atom. Default negation (denoted not A) is also allowed in the body of defeasible rules. Deriving literals in DeLP results in the construction of arguments. Let h be a literal, and $\psi= (\Pi, \Delta)$ a DeLP program. The$< A, h >$ is an argument structure for h, if A is a set of defeasible rules of D, such that:

1. there exists a defeasible derivation for h from $P \cup A$;
2. the set $P \cup A$ is non-contradictory, and
3. A is minimal: there is no proper subset A of A such that A satisfies conditions (1) and (2). In short, an argument structure $< A, h >$, or simply an argument A for h, is a minimal non-contradictory set of defeasible rules, obtained from a

defeasible derivation for a given literal h. The literal h will also be called the conclusion, supported by A. Note that strict rules are not part of an argument structure.

A derivation for the lateral can be defeasible or strict. Let $\psi = (\Pi, \Delta)$ be a DeLP and L a ground literal. A defeasible derivation of L from P, denoted $P \rightsquigarrow L$, consists of a finite sequence $L_1, L_2, \ldots, L_n = L$ of ground literals, and each literal Li is in the sequence because:

(a) L_i is a fact in P, or

(b) there exists a rule R_i in ψ (strict or defeasible) with head L_i and body $B_1, B_2, \ldots, B_k$ and every literal of the body is an element L_j of the sequence appearing before $L_i (j < i)$.

The derivation for literal h will be a strict derivation denoted by P $\rightarrow$L, if either h is a fact or all the rules used for obtaining the sequence $L_1, L_2, \ldots, L_n = L$are strict rules. Strict derivation does not require defeasible rules.

In the DeLP program, the P cannot be contradictory, whereas the $\psi = (\Pi, \Delta)$ may be contradictory. Let $\psi = (\Pi, \Delta)$ be a DeLP and the two literals h and h_1 disagree, if and only if the set $P \cup h, h_1$ is contradictory. When contradictory goals can be derived defeasibly, argumentation formalism is used to decide between them. DeLP, being declarative, does not capture interactions between pieces of knowledge and the burden of defeasible inference falls on the language processor. However, priorities could be used as an alternative approach. In DeLP, the construction of argument structures is non-monotonic: that is, adding facts or strict rules to the program may cause some argument structures to be invalidated because they become contradictory.

In DeLP, answers (yes, no, undecided, or unknown) to queries are supported by arguments. However, an argument may be defeated by another argument. Let us take an example where an argument $< A_1, h_1 >$ counter-argues, rebuts, or attacks $< A_2, h_2 >$ literal h, if and only if there exists a sub-argument $< A, h > of < A_2, h_2 >$ such that h and h_1 disagree. To compare arguments, two criteria are available:

1. Generalize Specificity: This favors two aspects of arguments as follows: a. Prefer an argument with greater information content, or b. Prefer an argument with less use of rules (more precise, more concise).

2. Argument comparison using rules priorities.

DeLP uses argumentation formalism to treat contradictory information by identifying contradictory information in the knowledge base and applying a dialectical process to decide which information prevails. Some formalisms define explicit priorities among

rules and use these priorities to decide between competing conclusions. The use of these priorities is usually embedded in the derivation mechanism and competing rules are compared individually during the derivation process. In such formalisms, the derivation notion is bound to one single comparison criterion. In DeLP, to decide between competing conclusions, the arguments that support the conclusions are compared. Thus, the comparison criterion is independent of the derivation process, and could be replaced in a modular way.

2.5.2.3 Defeasible reasoning-based argumentation engines

Bryant and Krause (2008) provided a very comprehensive survey of existing practical implementations of both defeasible and argumentation-based reasoning engines in the literature and emphasized the need for well-designed empirical evaluation and well-formed complexity analysis to justify the practical applicability of reasoning engines. Nathan [10] proposed an early implementation of a defeasible reasoner using specificity criteria to resolve conflict between generated arguments. To determine the support for a conclusion, a warrant procedure based on a series of incremental steps is used to classify an argument as "in" or "out" in a series of levels. This bottom-up approach to reasoning determines that an argument is warranted when it is "in" in at all remaining levels.

2.5.2.4 OSCAR

OSCAR proposed by Pollock (2000) is an agent-based architecture implemented in the LISP programming language for rational agents, i.e. it is equipped with practical and epistemic cognition. Practical cognition is about what to do and OSCAR directs the agent's interaction with the world, whereas epistemic cognition is about what to believe; most of the work in rational cognition is performed by epistemic cognition. The reasoning consists of the construction of arguments (nodes) to support the conclusion, linked to one another through atomic reasons (dependencies) and forming an inference graph. Two kinds of defeaters are identified in OSCAR. The first is the rebutting defeater which attacks the conclusion of inference, and the second is the undercutting defeater which attacks the connection between the premises and the conclusion. The resolution of conflict is carried out by reasoning schemas to compute the defeat status and the degree of justification, given the set of arguments constructed. The defeasible reasoner of OSCAR is allowed to draw conclusions tentatively, and as a result, an argument may be justified in one stage of reasoning and unjustified later, without any

[10] http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/reasonng/defeasbl/nathan/0.html, Last accessed (22/10/2012)

additional input. However, an argument is warranted when the reasoner reaches a stage where, for any new stage of reasoning, the argument remains undefeated. This is useful when dealing with limited resources, providing three possible statuses to a subset of arguments: defeated, undefeated, and justified.

2.5.2.5 IACAS

Vreeswijk (1995) presents another argumentation system working on defeasible argumentation, designed to carry out interactive argumentations on computers and allowing a person to start with a dispute, a given number of facts, rules and cases. The fact that it is interactive, is capable of finding the right number of arguments to reach a conclusion, and is capable of analysing the epistemic status of propositions, sets it apart from other argumentation systems. The building blocks of the argumentation process are propositions, and strict and defeasible rules. Strict rules represent deductive argumentation steps and defeasible rules represent plausible argumentation steps. Arguments are displayed in a tree-like structure with a conclusion on the left-hand side and their premises on the right-hand side. It has a command line interface and allows the evaluation of the status of an argument to be certain, beyond reasonable doubt, some presumption in favor, balanced or undetermined. IACAS implementation shows that defeasible arguments are essential for carrying out formal argumentation.

2.5.2.6 Critical and recommender systems (C & R)

Chesnevar et al. (2006b) identified that the current C & R systems are incapable of dealing with the defeasible nature of information. These systems are based on machine learning and information retrieval algorithms. With no inference capabilities, decisions rely on heuristics. Systems based on these quantitative approaches have been criticized for their inability to generate easily understandable and logically clear results; therefore, much of the implicit information remains uncovered. In this paper, the authors present a novel approach for the integration of user-supported systems such as critics and recommender systems with a defeasible argumentation framework to enhance the practical reasoning capabilities of such systems. Formalisms such as description logics can be integrated to achieve this objective but they lack the capability to deal with the defeasible nature of user preferences. DeLP has therefore proven to constitute the simplest yet most expressive language for encoding rule-based knowledge with incomplete and potentially inconsistent information. The user preference criteria are modeled as facts, strict rules and defeasible rules which, in turn, with the addition of background information, are used by the argumentation framework to prioritize

suggestions, thus enhancing the final results provided to users. Gomez et al. (2005) tried to integrate ontology theory, defeasible argumentation and belief revision to define ontology algebra, and suggested how different aspects of ontology integration can be defined in terms of defeasible argumentation and belief revision. OWL ontology is simply a collection of information comprising classes and properties, an approach which is associated with the DeLP program for representing knowledge in which facts and strict rules are distinguished. More formally, Ad-Ontology is a DeLP program P = (KP , KG, Δ) where KP stands for particular knowledge (facts about individuals), KG stands for general knowledge (strict rules about relations held among individuals), and Δ stands for defeasible knowledge (defeasible rules).

2.5.2.7 Miscellaneous applications

Rahwan and Larson (2008) identified that little work exists on understanding the strategic aspects of argumentation among self-interested agents and introduced an argumentation mechanism design (ArgMD) which enables the design and analysis of argumentation for self-interested agents. This work lies at the intersection of game theory and formal argumentation theory. In this mechanism, the agent must decide which arguments to reveal simultaneously and the mechanism calculates the set of accepted arguments based on acceptability criteria.

Rahwan et al. (2004) also identified the role of argumentation in the agent's negotiation. They identify the elements of environment (e.g. communication language, domain language, negotiation protocol) that host the agents and proposed a conceptual framework which outlines the core features required by agents for argumentation-based negotiation. Such negotiation will enable agents to operate in a dynamic, uncertain and unpredictable environment.

Dung et al. (2009b) also proposed a framework and described the extensive application of argument-based decision making and negotiation to a real-world scenario in which an investor agent and an estate manager agent negotiate to lease land for a computer-assembly factory. Agents are equipped with beliefs, goals, preferences, and argument-based decision-making mechanisms, and take uncertainties into account. Argumentation techniques are used in multi-agent systems to specify autonomous agent's reasoning, which involves forming and revising beliefs and actions according to inconsistent, uncertain and contradictory information. Such techniques have been used to facilitate multi-agent interactions which involve the dialogue process between software agents who have contradictory views about certain domains of discourse.

2.6 Comparison between philosophical and logic-based argumentation frameworks and applications

Many similarities as well as differences exist between philosophical and logic-based argumentation frameworks and applications.

Table 2.8 provides a comparative study of both types of argumentation frameworks and applications based on a number of distinct features. Of the various features, the most important are the representation of an argument structure and reasoning methodology. Philosophical models consider an enriched and complex argument structure which comprises a set of elements that facilitate the subjective assessment of an argument by the participants in decision making. The acceptability of arguments is again subjective in nature, e.g. strong, moderate, weak etc., and is dependent on human computation. Whereas logic-based argumentation models and applications consider argument to be a very simple structure and try to define explicit semantics so that a reasoning engine can evaluate the arguments, the evaluation of arguments in logic-based argumentation models and applications is very simple, i.e., it is either true or false. As a result, it is very easy to compute an acceptable set of arguments.

Argument structure and reasoning methodology are therefore the two features that are of paramount importance for the design and development of new software applications. A comparative study of both eases the decision about which argumentation paradigm to use for the development of new software applications.

Table 2.8: Comparison of logic-based argumentation frameworks/applications with philosophical models of argumentation/applications

Feature	Logic-based argumentation frameworks /applications	Philosophical models of argumentation
Argument structure	Most of the argumentation framework take arguments as atomic entities and focused on the interaction among arguments. In logic-based application, the arguments comprise of a set of premises and a conclusion.	Focus on enriched representation of the internal structure of an argument and interaction among elements of an argument such as facts, claim, warrant, backing, modality etc.
Argument type	Mostly comprise of deductive arguments	Mostly comprise of inductive and presumptive arguments
Justification for a claim	Rely on rule of inference	Rely on enriched structure of an argument
Acceptability of arguments	automated	manual or semi-automated
Arguments evaluation	Evaluation of arguments relies on the rigorous testing of arguments based on mathematical rules which declare them true or false.	Evaluation of arguments is done by humans who weighted or weighted them as either weak or strong.
Argumentation type	Mostly follow monological argumentation	Mostly follow dialogical argumentation
Reasoning	The reasoning is performed by a software component i.e.reasoning engine.	The reasoning task is either fully human dependent or semi-automated.
Initialization	Logic-based argumentation frameworks start with a set of existing arguments and the interaction among them. The application follows dynamic argumentation.	Tools built on a philosophical model of argumentation facilitate the participants/human to construct arguments.

In Section 2.7, the current reasoning approaches on the Semantic Web are categorized.

2.7 Categorization of reasoning approaches on the Semantic Web

As pointed out in Chapter 1, the ontology languages layer of the Semantic Web have reached a level of maturity and now efforts are being focused on the development of the logic layer of the Semantic Web. The logic layer provides a foundation for Semantic Web applications to perform advance reasoning techniques for automated information extraction, reasoning and integration to facilitate the decision-making process.

Broadly speaking, the current reasoning approaches on the Semantic Web can be divided into the following two categories:

1. Monotonic reasoning

   A reasoning is known as monotonic reasoning if during the reasoning process, once a conclusion is asserted, it can't be retracted later on in the presence of new information. Monotonic reasoning follows Open-World Assumptions (OWA) where everything I don't know or information which is not present in the model is considered undefined.

   The current monotonic reasoning-based approaches on the Semantic Web can be classified into the following three sub-categories:

   a) Ontology-driven reasoning: Approaches that make use of ontologies for knowledge representation and reasoning. Section 2.7.1.1 elaborates on ontology-driven reasoning in detail.

   b) Semantic Web rule-based reasoning: Approaches that make use of the Semantic Web rule-based languages to represent and reason over information present on the Semantic Web . These approaches are presented in Section 2.7.1.2

   c) Fuzzy logic-based approaches : Approaches that make use of fuzzy logic to represent and reason over information present on the Semantic Web. These approaches are presented in Section 2.7.1.3

2. Non-monotonic reasoning

   A reasoning is known as non-monotonic reasoning if, during the reasoning process, once a conclusion is asserted, it can be retracted later in the presence of new information. Non-monotonic reasoning follows Close-World Assumptions (CWA)

where everything I don't know or information which is not present in the model is considered false. The current non-monotonic reasoning-based approaches can be classified into the following two sub-categories:

a) Defeasible logic-based approaches: Approaches that make use of defeasible logic-based rule languages to represent and reason over information present on the Web. These approaches are discussed in Section 2.7.2.1

b) Argumentation-based approaches: Argumentation-based approaches make use of argumentation techniques to represent and reason over information present on the Web. These approaches are presented in Section 2.7.2.2

In the following Section 2.7.1, the different sub-categories of monotonic reasoning are discussed in detail, followed by non-monotonic reasoning sub-categories in Section 2.7.2.

2.7.1 Sub-categories of monotonic reasoning

2.7.1.1 Ontology-driven reasoning

Ontologies (Fensel, 2003) are the core of the Semantic Web and provide formal and explicit specification of a certain domain. They use a combination of classes, and their relationships or properties, instances and axioms are defined in some formal language. The W3C has proposed two ontology languages for representing knowledge on the Semantic Web. The first one is RDFS, based on XML and logic programming, which is a lightweight ontology language. The second language is OWL, which is based upon description logic and provides constructs for cardinality restrictions, Boolean expressions and restrictions on properties. OWL ontologies come in three species: Lite, DL, and Full, ordered in increasing expressivity.

In addition to serving the purpose of representation, an ontology also enables logical inference on facts through axiomatization. Hence, ontologies on the Web should provide constructs for effective binding with logical inference primitives and options to support a variety of expressiveness and computational complexity requirements. Table 2.9 depicts a set of axioms defined for OWL-Lite and these are exploited by DL reasoning engines, such as FaCT++ (Tsarkov and Horrocks, 2006) and Pellet (Parsia and Sirin, 2007) to achieve the inference-ability objective. A number of architecture and web applications have been built by modeling domain knowledge in the form of ontologies, using the DL reasoner as an inference engine, such as a reasoning agent for the Semantic Web (Oguz et al., 2008), OSGi-based infrastructure to manage context-aware services (Gu et al., 2004) etc.

Table 2.9: OWL ontology reasoning semantics

Property	Semantics
TransitiveProperty	(?P rdf:type owl:TransitiveProperty) ∩(?A ?P ?B) ∩(?B ?P ?C) ⇒(?A ?P ?C)
subClassOf	(?a rdfs:subClassOf ?b) ∩ (?b rdfs:subClassOf ?c) ⇒ (?a rdfs:subClassOf ?c)
subPropertyOf	(?a rdfs:subPropertyOf ?b) ∩(?b rdfs:subPropertyOf ?c) ⇒ (?a rdfs:subPropertyOf ?c)
disjointWith	(?C owl:disjointWith ?D) ∩ (?X rdf:type ?C) ∩(?Y rdf:type ?D) ⇒(?X owl:differentFrom ?Y)
inverseOf	(?P owl:inverseOf ?Q) ∩ (?X ?P ?Y) ⇒ (?Y ?Q ?X)

2.7.1.2 Semantic Web rule-based driven reasoning

Proposals for the integration of rule languages and ontology languages can be classified by the degree of integration (Antoniou et al., 2005). Firstly, the hybrid approach is one where there is strict separation between the rule predicates and ontology predicates and reasoning is done by interfacing the existing rule reasoner with the ontology reasoned; whereas, with the homogeneous approach, both rules and ontologies are embedded in the same logical language $\mathcal{L}$ without making a prior distinction between the rule predicates and ontology predicates, and the reasoning single reasoner can be used for reasoning purposes.

The following two steps are involved:

- Compilation of rules as a Rete network.
- Matching phase i.e. data-driven reasoning by passing the facts present in the working memory through the Rete network.

The Rete (Forgy, 1982) algorithm involves two steps. The first is the compilation of rules in the form of a network called a Rete network. The second is the matching phase, in which the rule engine matches the conditions of the rules in the knowledge base against the facts in the working memory. As a result of this match, a single rule fires. Firing the rule instance will add a new fact to the working memory. The matching phase starts again and only the new inferred facts filter through the compiled rules network and result in the firing of another rule; so the process continues. The process will stop when no more rules match the new inferred facts.

The importance of Semantic-based Web-based decision support systems (DSS) in business applications has been identified by a number of researchers over a period of time (Vahidov and Kersten, 2004; Silverman et al., 2001; Toni, 2007). Kartha and Novstrup (2009) proposed a combination of ontologies and decision rules for building a decision support application for time sensitive targeting. They represented knowledge with the help of rules known as 'decision rules' which: (a) include primitives from multiple ontologies and primitives that are defined by algorithms that are outside the rule framework; (b) are time-dependent; and (c) incorporate default assumptions. They developed what is known as the Sentinel system, which is general enough to support a wide variety of DSS tasks.

Ceccaroni et al. (2004) present an environmental decision support system (called OntoWEDSS) for waste water treatment to improve the diagnosis of faults in a treatment plant, which provides support for complex problem-solving and facilitates knowledge modeling and reuse. The system is based on the integration of case-based and rule-based reasoning with an ontology, i.e. Waste-Water Ontology (WaWO) for the representation of the domain and for reasoning. Nicolicin-Georgescu et al. (2010) present an approach to managing data warehouse cache allocations via DSS by using autonomic computing and Semantic Web technologies. They presented heuristics for autonomic computing adoption, using ontologies for DSS system modeling and ontology-based rules for heuristic implementation.

Similarly, Salam (2007) presents a supplier performance contract monitoring and execution of DSS, using OWL-DL [11] for knowledge representation SWRL [12] to express rules on top of OWL-DL ontologies. Cheung and Cheong (2007) address the challenges of market operations using a rule-based approach in mission-critical decisions and Garcia-Crespo et al. (2011) propose a semantic model for knowledge representation in e-business. Yang et al. (2009) proposed a Semantic Web-DSS and provide semantics for defining static and dynamic semantics representation based on ontologies and quantitative decision making comprising three steps: publishing decision requirements, bidding, and role-based collaboration among decision peers (each Semantic Web-DSS is a peer) to negotiate decision models.

2.7.1.3 Fuzzy logic-based reasoning

A number of researchers used fuzzy logic-based quantitative approaches for reasoning to address the issues of group decision-making. Subsorn et al. (2008) proposed a Web-based group decision support system framework to deal with imprecise

[11] http://www.w3.org/TR/owl-guide/
[12] http://www.w3.org/Submission/SWRL/

decision-making problems. The framework is based on a fuzzy analytic hierarchy process for group decision-making. The framework enables group members to develop satisfactory group solutions and allows group leaders to form final/acceptable, satisfactory group solutions. Ma et al. (2010) proposed 'Decider', a fuzzy multi-criteria group decision-making (MCGDM) process model that aims to support preference-based decisions over the available alternatives that are characterized by multiple criteria in a group. The model can handle information expressed in linguistic terms, Boolean values, as well as numeric values to assess and rank a set of alternatives within a group of decision makers.

Noor-E-Alam et al. (2010) also addressed the issue of multi-criteria decision-making (MCDM) involving multiple experts and pointed out that the participation of many experts makes the conflict aggregation process difficult. They developed a DSS based on types of fuzzy-based conflict aggregation algorithms, namely, a possibility measure and averaging conflict aggregation. Yue (2011) addressed the issue of multiple attribute decision-making (MADM) and developed an algorithm for determining the weights of decision makers within a group decision environment, in which the information regarding each individual decision is expressed by a matrix in interval numbers. He also defined positive ideal and negative ideal solutions of group opinion, the separation measures and the relative closeness from the positive ideal solution.

Cabrerizo et al. (2010) used fuzzy logic to address the issue of consensus building among experts when information is incomplete. They developed a consensus model to address group decision-making problems with incomplete unbalanced fuzzy linguistic information. The working of the model is supported by consistency and consensus measures, and with the help of a feedback mechanism, personalized advice is provided to the experts for modification to their unbalanced fuzzy linguistic preference relations. Similarly, efforts are being made to represent the results of the decision-making process to the end user in an easily comprehendible form, such as that of Li et al. (2001) who proposed a visualized information retrieval engine based on fuzzy control.

The aforementioned fuzzy logic-based approaches, being quantitative, have been criticized for their inability to generate easy-to-understand and logically clear results for justification purposes. These approaches follow monotonic behaviour whereby once a conclusion has been drawn, it cannot be retracted. Additionally, they lack inference reasoning capability over contradictory information; for BI, we need such inference mechanisms.

2.7.1.4 Description Logic Programs (DLP)

Logic Programming is a predominant paradigm for expressing knowledge with rules, and for making inferences and answering queries. It provides both a declarative reading (a programming paradigm that expresses the logic of a computation without describing its control flow) and an operational reading of rules (with implementations). Its semantics largely underpin four families of rule systems, i.e. SQL relationship databases, OPS5 heritage production rules, Prolog, and Even-Condition-Action rules, and it is used as the proposal for rules in the context of the Semantic Web.

Many efforts have focused on mapping, intersection, or a combination of description logics (DLs) and logic programs (LP) to overcome the shortcomings that emerged during the development of practical OWL applications (Patel-Schneider and Horrocks, 2007). To overcome the limitations of reasoning on OWL, Grosof et al. (2003) proposed Description Logic Programs (DLP) which lie at the intersection of LP and DLs (as shown in Figure 2.5), instead of using Full First Order Logic (FOL) to address OWL issues. FOL can express positive disjunctives which are inexpressible in LP, whereas it does not provide support for expressing negation-as-failure (representing incomplete information) and procedural attachments (e.g. the association of an action performing procedural invocation with the drawing of a conclusion about a particular predicate). On the other hand, Logic programs do not provide these features to support the non-monotonic behaviour of the system.

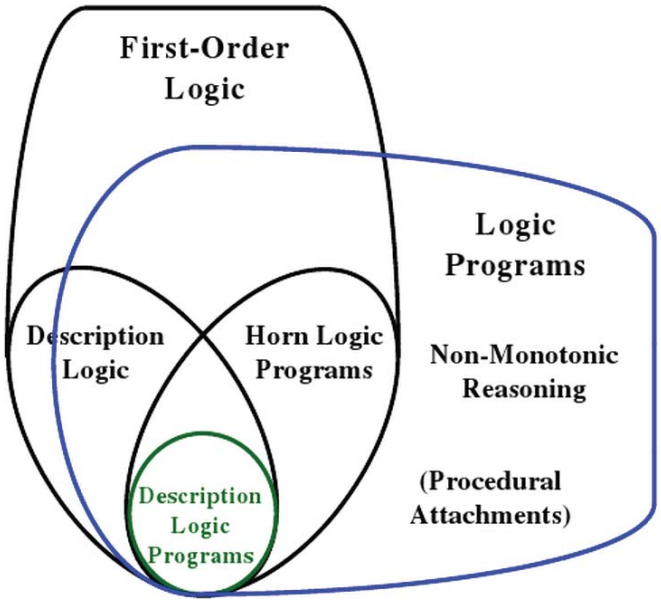


Figure 2.5: Expressive overlaps among knowledge representation languages (Grosof et al., 2003)

2.7.2 Sub-categories of non-monotonic reasoning

2.7.2.1 Defeasible logic-based reasoning

Nute (1988) highlighted the importance of defeasible reasoning in decision support systems and developed a logic for defeasible reasoning by extending Prolog. The new logic comprises facts and presumption, absolute rules and defeasible rules, and introduced another kind of weak rule known as a 'defeater'. Causey (1994) developed 'EVID', a system for interactive defeasible reasoning and Johnston and Governatori (2003) developed an algorithm that integrates defeasible logic into a decision support system by automatically deriving its knowledge from databases of precedents.

Dr Prolog (Antoniou and Bikakis, 2007) is a prolog-based implementation for carrying out defeasible reasoning on the Web. It provides declarative system support rules, facts, ontologies, RuleML, and both monotonic and non-monotonic rules. It takes into consideration both open world and closed world assumptions and provides features for reasoning with inconsistencies. The system provides a number of variants such as ambiguity blocking, ambiguity propagation and contradictory literals. Defeasible theories are imported in defeasible logic or RuleML syntax and translated into logic programs with the help of a logic translator. The Reasoning Engine compiles the logic programs and the meta-program which corresponds to the DL version that the user selects (ambiguity blocking/ propagating), and evaluates the answers to the user's queries. They extended RuleML DTDs to represent defeasible theories in XML format. Dr Brokering (Antoniou et al., 2007) is a Dr-Prolog-based software agent implementation to address the problem of brokering and matchmaking; i.e. how a requester's requirements and preferences can be matched against a set of offerings collected by a broker.

Dr-Device (Kontopoulos et al., 2011; Bassiliades et al., 2004) is a CLISP-based defeasible reasoning implementation provided with a VDR-Device reasoning system, RDF loader/translator and rule loader/translator component. The VDR-Device is an integrated development environment equipped with a graphical front end used for deploying defeasible rules on top of RDF schema ontologies. The rule base is initially submitted to the rule loader which transforms the rules into CLISP-like syntax through an XSLT stylesheet. The resulting program is forwarded to the rule translator where defeasible logic rules are compiled into a set of CLISP production rules. In parallel, the RDF downloader downloads the RDF documents and translates them into CLISP objects according to the RDF-to-Object scheme. The reasoning system performs inference on transited RDF metadata using defeasible rules and generates the objects that constitute the result of the initial rule program. The RDF extractor exports the

resulting objects in the form of RDF/XML to the user. Dr-Device is implemented in Jess and integrates well with RuleML and RDF. Unlike Prolog, Dr-Device supports only one variant: ambiguity blocking. At present, it does not support OWL ontologies. In addition, Dr-Prolog uses Logic Programs with well-founded semantics, which is formally equivalent to the formal model. In contrast, Dr-Device uses the logic meta-program as a guiding principle, but there is no formal proof of the correctness of the implementation. On the other hand, Dr-Device has the relative advantage of easier integration with mainstream software technologies. SweetJess (Grosof et al., 2002) is another defeasible reasoning system based on Jess and closely resembles courteous logic programs. It integrates well with RuleML but it can only perform reasoning in DML + OIL ontologies and not on RDF data as Dr-Device and Dr-Prolog does. However, it allows for procedural attachment and it implements only one reasoning variant. Moreover, it imposes a number of restrictions on the programs so that it can map on Jess. Table 2.10 presents a comparison of defeasible reasoning-based information systems.

2.7.2.2 Argumentation-based approaches

The WWW, being distributed and ubiquitous, provides a universal platform for Internet users to interact with each other. Previously, there was one-way traffic of content contributors on the WWW. The content provides information which was mainly based on their thinking, observations and knowledge and the readers were not able to reply to the author's arguments if a difference of opinion existed.

Web 2.0 has revolutionized the WWW and provides a platform for the readers and converts them from reader to content developer. This development led to the realization of argumentation among users of the WWW. Blogs are one of the best examples of this. Semantic Web technology adds more flavour to Web contents by enriching the content with certain semantics to make the content processable by machines and automate interaction and support the decision-making process. On the basis of the level of functionality, the current applications of argumentation on the WWW are divided into the following three categories:

1. Web-based argument-assistance systems.
2. Semantic Web-based argumentation support frameworks and applications.
3. Semantic Web-based argumentation support applications with shared ontology (AIF).

In the following sections, each of these categories is discussed in detail.

Table 2.10: Comparison of defeasible logic based Web IDSS applications

	Dr-Prolog	Dr-Device	Situated Courteous logic
Language	Prolog	JESS	JESS
Logic	Defeasible logic	Defeasible logic	Situated Courteous logic
Semantic data	RDFS/OWL	RDF	DAML+OIL
Rules representation	RuleML	RuleML	RuleML
Incomplete knowledge representation	Yes	Yes	Yes
Conflict representation	Yes	Yes	Yes
Data-driven reasoning	No	Yes	Yes
Goal-driven reasoning	Yes	No	No
Conflict resolution	User defined priorities at compile time	User defined priorities at compile time	User defined priorities at compile time
Explanation	Textual	Textual	Textual
AIF reification	No	No	No

2.7.3 Web-based Argument-assistance systems

Web 2.0 is a powerful paradigm for designing argumentation tools to solve challenges in collaboration on a global scale. However, there is a huge gap between Web 2.0 technologies and argumentation formalisms. Argumentation formalism focuses on a particular kind of semantic structure for organizing elements in such a way that computation and inference can be performed to reach a conclusion, whereas Web 2.0 moves the emphasis away from argumentation formalism features, such as no predefined information organization schemes, and is more focused on self-organization and community-driven indexation of elements, e.g. folksonomies that can be rendered as clouds (Shum, 2008).

To bridge this gap, an argument assistance application offers a step forward; for instance, argument assistance systems overcome the limitations of threaded discussion forums by making a clear distinction between unsupported premises and supported premises known as claims. To evaluate the existing applications, I define a scale for argument evaluation and argument acceptability as depicted in Table 2.11.

Table 2.11: Scale for evaluation and acceptability of arguments

Type	Scale	Description
Scale for Argument Evaluation	Fully Human dependent (FHD)	System depends upon participants for arguments evaluation
	Partially Human dependent(PHD)	System provides support to participants by linking current information to other resources or external links that can provide some sort of justification or arguments network for better visualization to facilitate argument evaluation.
	Not Human dependent (NHD)	System has built-in semantics for arguments evaluation e.g some logical criteria.
Scale for Acceptability of arguments	Fully Human dependent (FHD)	System depends upon participants to compute/ consider acceptable arguments.
	Partially Human dependent (PHD)	System provides support to participants in terms of external useful links or arguments; graph-like structure to facilitate participants to tag acceptable arguments.
	Not Human dependent (NHD)	System has built-in semantics for arguments evaluation e.g. voting mechanism

Table 2.12 shows a comparative analysis of different web-based argument assistance applications, from which the following important observations can be made:

1. Most argumentation assistance applications are based on dialogical argumentation with the exception of Debatabase [13], which follows monological argumentation.

2. Argument structures vary from very simple structures, such as premises and conclusions, to very complex argument structures, such as the argument structures in ConvinceMe[14], Debatepoint[15] and Truthmapping.[16]

3. Argumentation is mostly used for application assistance in persuasion and debate.

4. The evaluation of arguments is either fully or partially dependent on humans. None of the systems have the semantics to automate the process of argument evaluation.

5. The acceptability of an argument is either fully human-dependent or is not at all dependent on humans. In the latter case, different mechanisms are used, such as voting.

6. Content contributors are not as prolific as they are on the social networks.

[13]http://www.idebate.org/debatabase/topic-index.php
[14]http://www.convinceme.net/
[15]http://debatepoint.org/
[16]http://www.truthmapping.com/

Table 2.12: Comparison of Web 2.0 based argument assistance systems

Framework /application	Purpose	Argument Structure	Evaluation of arguments	Acceptability of arguments	Content contributors	Argumentation type
Lead line (Farnham et al., 2000)	Persuasion	Premise, Conclusion	FHD	FHD	A Small group	Dialogical
ConvinceMe.net	Persuasion	Premises, Pro, Cons, convinced, rebuttal, evidence, comments	PHD	NHD	Medium (on avg. 5-20 contributors)	Dialogical
Truthmapping	Persuasion	Premises, conclusion, agree, disagree, withdrawn	PHD	NHD	Medium (on avg. 5-20 contributors)	Dialogical
Debatabase (Experts)	Learning	Premise, Pro,Cons, Conclusion	FHD	FHD	A person	Monological
IBISMod	Problem solving	Issues, positions and arguments	FHD	PHD	Medium (on avg. 5-20 contributors)	Dialogical
DebetePedia	Debate	Premise, Pro,Cons	PHD	FHD	Medium (on avg. 5-30 contributors)	Dialogical
Debatepoint	Debate	Premise, attack, support, moderate, irrelevant, comments	FHD	NHD	Large(30-above)	Dialogical
Deme	Debate	Premises, opinions, comments	PHD	PHD	Medium (on avg. 5-20 contributors)	Dialogical
Discourse DB	Predicting trends about issues	Premise, pro, cons, mixed	PHD	FHD	Large (30 -above)	Dialogical
Standpoint	Clustering of people with same belief	Premise, Attack, Support, Comments	PHD	NHD	Large (30-above)	Dialogical

2.7.4 Semantic Web-based argumentation support frameworks and applications

The Semantic Web is an extension of the WWW on which information is annotated with meta-data or ontologies to make it processable by machines. The Semantic Web plays an important role in automating the computing of user interaction. Realizing the importance of argumentation, Sprado and Gottfried (2009) defined an argumentation-based framework for a decision support system in the context of spatio-temporal systems. The DS framework is based on two paradigms, argumentation and description logics. Argumentation is applied to identify and analyze consistent sets of arguments, whereas description logics help to define terminological knowledge to categorize the arguments at a semantic level. In this framework, arguments refer to a conceptual description of a given state of affairs (Concept-Based Argumentation) and use the preferences among them to resolve conflicts at a conceptual level. Similarly, CoAKTinG (Bachler et al., 2004) provides tools to assist scientific collaboration by integrating intelligent meeting spaces, ontologically annotated media streams from online meetings, decision rationales and group memory capture, meeting facilitation, issue handling, planning and coordination support, constraint satisfaction, and instant messaging or presence.

The HCONE Kotis (2010) argumentation ontology supports the capture of the structure of an entire argumentation dialogue as it evolves among collaborating parties within a period. It allows the tracking and identification of the rationale behind atomic changes and/or ontology versions. CoPe_it! also provides a mechanism to evaluate the strength of a position, and so represents another interesting development. Positions or alternatives are posted after the completion of an appropriate form. Each time a user posts a discourse item, CoPe_it! re-evaluates the whole discussion and indicates a solution.

Table 2.13 provides a comparative analysis of different argumentation-based Semantic Web applications, summarized as follows:

1. Apart from debate, they are used to predict trends and cluster information.
2. Applications follow dialogical argumentation.
3. Current applications are not fully autonomous because they are partly dependent on humans for their functionality.

Table 2.13: Comparison of semantic based argumentation support applications

Framework /application	Ontology	Purpose	Argument Structure	Evaluation of argument	Acceptability of arguments
HCONE (Kotis, 2010)	Argumentation ontology	Handling conflitcs in shared ontology development	issues, positions, and arguments (for, against)	FHD	PHD
Cope_it (Tzagarakis et al., 2009)	Incremental formalization	Learning	items, favor, against	PHD	NHD
Cohere (Shum, 2008)	Cohere Data model	Ideas-linking	Ideas, Question, Answer, Pro, Con	PHD	PHD
Cicero (Dellschaft et al., 2008)	DILIGENT argumentation ontology	Handling conflitcs in shared ontology development	Issues, positions, and arguments (for, against)	FHD	PHD
CoAKTinG (Bachler et al., 2004)	E-Learning	Issues, Ideas, Arguments and Decisions	Dialogue Mapping	PHD	PHD

2.7.5 Semantic Web-based argumentation support applications with a shared ontology (AIF)

Currently, a large number of interactions occurring on the WWW need to be captured in certain semantic structures to make it possible for them to be explored by others (to back up their argument's support or rebuttal), and to automate the process of argument build-up and analysis. Argument interchange format (AIF) is one step towards providing a standard ontology for capturing such interactions (Rahwan et al., 2007b; Chesnevar et al., 2006a; Iyad Rahwan, 2009). Table 2.14 depicts different argumentation applications that share a common ontology.

Table 2.14: Comparison of semantic web-based argumentation support system with shared Ontology

Framework /application	Purpose	Argument Structure	Evaluation of argument	Acceptability of arguments	Argumentation type
ArgDF [17]	Representing and visualizing arguments	Argument structure is defined by argumentation scheme	PHD	NHD	Dialogical
Argument blogging (Wells et al., 2009)	Harvesting textual resources from the web and structuring them into distributed argumentative dialogue	Premise, support, refute, attack inference	PHD	PS-HD	Dialogical
Araucaria (Reed and Rowe, 2004)	Representing and visualizing arguments	Support different diagrams with rich structure	FHD	FHD	Dialogical
SIOC Argumentation Module(Uldis Bojars et al., 2008)	Modeling the structure of the discussions on social websites	Statement, issue, idea, elaboration, decision, position	PHD	PHD	Dialogical

[17]http://www.argdf.org/

2.8 Critical evaluation of the existing approaches to support monological argumentation in Semantic Web applications

In this section, a critical evaluation of the existing approaches in the literature for information representation and reasoning in the Semantic Web is presented in order to build an integrated view and identify the key issues that need to be addressed to have a complete methodology that provides monological argumentation-driven automated reasoning support in Semantic Web applications. The provision of such a methodology will enable enterprises to consider information that is potentially incomplete and/or contradictory which exists either within the enterprise or in other enterprises, represent and perform automated reasoning over it to identify and resolve any conflicts which may arise, followed by the integration and representation of the reasoning results to assist decision makers in the enterprise-wide decision-making process.

As seen from the discussion on reasoning approaches in the literature, decision makers are extremely dependent on software applications to assist them in the process of decision making (Carlsson and Turban, 2002; Shim et al., 2002). They need intelligent applications that can transform information (which may be incomplete and/or contradictory) into useful knowledge as well as providing qualitative insights, so that a human style of reasoning can be expected in software applications. To address this, researchers in the field of Artificial Intelligence (AI) have long been striving to realize human-like decision-making power in software applications. The vision of Semantic Web applications also derives its concepts from AI. However, as discussed in Section 2.7.1.1, the logic-based languages that lie at the logic layer of the Semantic Web are deductive in nature and perform monotonic reasoning i.e. reasoning under assumptions that *the underlying information for decision making is consistent and the addition of new information doesn't result in contradictions with existing information*(Antoniou and Van Harmelen, 2004; Horrocks et al., 2005) . In other words, they assume that

i) no conflicts will arise during the process of decision-making, and

ii) new information will not result in a different output.

To overcome the limitations of the Semantic Web discussed above, defeasible reasoning based approaches have been proposed in the literature which enable Semantic Web applications to perform non-monotonic reasoning over incomplete and/or contradictory information (Antoniou and Bikakis, 2007; Kontopoulos et al., 2011; Bassiliades et al., 2004). As pointed out in Sections 1.3 and 2.7.2.1, even though

defeasible reasoning seems to be a good option to address the issues of non-monotonic reasoning in Semantic Web applications, however, the superiority relation on defeasible rules are hard-coded preferences specified by a single user before performing reasoning, and if a conflict between the rules arises during reasoning, then the existing defeasible reasoning-based approaches don't provide a solution to address them. As a result of this, Semantic Web applications built using reasoning approaches are inflexible in responding to dynamic situations and they lack the ability to make judgments in such situations, unlike humans who may be able to make decisions even in situations where the information may be incomplete and/or contradictory.

To address this problem, the concept of '*argumentation*' has been studied in the literature on AI. Argumentation is much more closely related to a human style of reasoning that takes into account the concepts from the study of arguments to support opinions, claims, and proposals, and ultimately to lead to justifiable decisions and conclusions (Prakken and Vreeswijk, 2002; Obeid, 1992) . Toulmin (2003) was the first to provide the logical structure of an argument and his work has been extended by a number of researchers to enrich the argument structure and address a variety of reasoning problems in the philosophy of law and other disciplines. The formal foundations of argumentation have been well explored in the academic literature (as discussed in Sections 2.4 and 2.5). However, their major drawback is that most software applications that are based on logic-based argumentation formalisms are built separately, and are proprietary in nature. As a result of this, the code is not available for enhancement and general use and cannot be applied directly to address the issues of non-monotonic reasoning in Semantic Web applications.

Approaches have been proposed in the literature that apply the concepts of argumentation in the Semantic Web. It can been seen from the discussion in Section 2.7.2.2 that argumentation-based reasoning approaches have proven to be very useful in empowering Semantic Web applications. It enables Semantic Web applications to take into account potentially incomplete and/or contradictory information and through argumentative reasoning, bring these to an agreeable conclusion, if possible. However, it is evident from the discussion in Sections 2.7.3, 2.7.4 and 2.7.5 that most argumentation-based Web applications are dialogical in nature where the reasoning mechanism is driven by the decision makers involved in the discussion. As a result of this, argumentation-based Semantic Web-applications are missing a very important and reusable component, that is, a reasoning engine capable of performing monological argumentation over underlying information that may be incomplete and/or contradictory. This is considered to be an integral part of Semantic Web applications for product recommendation, auctions, identification of requirements, vendor selection, negotiation, agent communication and information integration (Deng and Wibowo,

2008; Cheung and Cheong, 2007; Shim et al., 2002; Assche et al., 1988; Wen et al., 2008; Dong et al., 2011; Xue et al., 2012). So, due to the lack of reusable components (i.e. monological argumentation driven reasoning engine), most of these existing Semantic Web applications follow philosophical argumentation-based frameworks where reasoning is performed by humans to cogitate and evaluate arguments and to take action.

So, to have a reasoning engine that performs monological argumentation in Semantic Web applications, the current approaches discussed in the literature do not provide any solution. Hence, the main inadequacy of the existing approaches, from the literature discussed above, in having an argumentation-based approach for reasoning in Semantic Web applications which addresses all the aspects required for taking into account potentially incomplete and/or contradictory information either within an enterprise or in other enterprises can be summarized as:

1. Incapability of logic-based languages to represent information that is potentially incomplete and/contradictory coming from different sources either within an enterprise or in other enterprises.

2. Absence of a monological argumentation-driven reasoning engine (i.e. hybrid reasoning engine) to identify and resolve conflict in the underlying information.

3. No methodology for information and knowledge integration and their graphical representation to assist the decision maker in enterprise-wide decision making.

In the following sub-sections, each of these issues is discussed in detail.

2.8.1 Incapability of logic-based languages to represent information that is potentially incomplete and/contradictory coming from different sources

The reasoning approaches proposed in the literature, such as ontology-driven reasoning, Semantic Web rule-based reasoning and DLP discussed in Sections 2.7.1.1, 2.7.1.2 and 2.7.1.4 respectively are based on description logic (DL) which is a subset of predicate logic and therefore it inherits the limitations of predicate logic i.e. it only performs monotonic reasoning under certain assumptions as follows:

1. The given problem can be fully addressed with the available information (i.e. the solution to the problem lies within the available information).

2. The information or specification of rules required for decision-making is consistent. In other words, it is assumed that no contradictory information will emerge during the decision-making process.

3. If new information is added to the application, it will be consistent with the already available information or specifications.

4. New information does not lead to a retraction of previous conclusions.

which limits their capability to represent and reason by taking into account the information present on the Semantic Web that could be potentially incomplete and/or contradictory. To overcome the abovementioned problem, defeasible logic-based implementation has been proposed in the literature that provides a formalism to represent incomplete and/or contradictory information from a single user/source. In this approach, a decision maker can define his preferences over the contradictory rules at design time and these preferences are used to resolve conflicts during the process of automated reasoning. However, these approaches do not provide a solution for information representation when incomplete and/or contradictory information comes from different sources and when there is more than one user involved in the decision-making process. Hence, from the above discussion, it can be inferred that existing Semantic Web stack languages present at the logic-layer of the Semantic Web are incapable of representing incomplete and/or contradictory information that may exist within the enterprise or in different enterprises and make it available for reasoning purposes. Semantic Web applications built using these languages fail to represent information where contradictory information may come from different users/sources. However, such an approach is needed to capture all the information and the decision makers' opinions during the decision-making process.

In Chapter 3, the problem associated with the representation of information which is potentially incomplete and/or contradictory is identified and defined, and in Chapter 4, a solution is proposed to address the problem defined in the existing literature.

2.8.2 Absence of an monological argumentation-driven reasoning engine to identify and resolve conflicts present in information coming from different sources

The issue i.e. the absence of an argumentation-driven reasoning engine to identify and resolve conflicts present in information coming from different sources, can be subdivided into the following sub-issues:

1. Rete network and its limitations.

2. Lack of hybrid reasoning in Semantic Web reasoning engines.

3. Lack of different argumentation-driven conflict resolution strategies.

In the next sub-sections, each of these issues is discussed in detail.

2.8.2.1 Rete network and its limitations

Semantic Web application reasoning engines use the Rete network for the compilation of rules and work in close coordination with the working memory. However, the compilation of rules that may represent incomplete and/or contradictory information is not possible in the existing Rete network due to the following limitations:

1. The general Rete network works only for predicate logic-based rule languages that follow monotonic reasoning. Therefore, it is not capable of representing potentially incomplete and/or contradictory information as Rete nodes.

2. A Rete network only executes one rule in a single match-execute cycle. If two rules are activated, only the rule with the higher order preference for execution defined by an individual (owner of the contradictory rules) will be executed. When underlying information is potentially incomplete, to capture it may require the execution of both contradictory rules each of which may represent a different view point. However, the current Rete network fails to address this objective.

Hence, there is need to extend the Rete network for the representation of incomplete and/or contradictory information as Rete nodes and enable the two contradictory production rules to fire and instances of both production rules i.e. arguments, to be added to the argument set. In Chapter 3, the problem associated with the Rete network is formally identified and defined, and in Chapter 4, a solution for the problem defined in the existing literature is proposed as well as extensions to the Rete network in order to compile business rules (representing incomplete and/or contradictory information) in the form of a Rete network.

2.8.2.2 Lack of hybrid reasoning in Semantic Web reasoning engines

Attempts have been made in the literature to perform reasoning over incomplete and/or contradictory information in order to realize non-monotonic reasoning in Semantic Web applications such as Dr-Prolog (Antoniou and Bikakis, 2007), Dr-Device Kontopoulos et al. (2011); Bassiliades et al. (2004) and Situated Courteous logic (Grosof et al., 2002). These defeasible logic-based applications use either data-driven reasoning or goal-driven reasoning. Data-driven reasoning is used to move from current facts

to a conclusion, whereas goal-driven reasoning is backward chain reasoning used to move from a conclusion to the facts. But in the case of Semantic Web applications, both types of reasoning are needed: data-driven reasoning for the construction of arguments from underlying information and goal-driven reasoning to identify and resolve conflicts that exit between arguments. However, none of these attempts provide a solution that has both data-driven and goal-driven reasoning to reason over incomplete and/or contradictory information. Another requirement for the reasoning engine is that it should have the capability to resolve conflicts using different criteria, either automatically or being guided by the members of the decision-making process in order to achieve their goals. Defeasible logic-based attempts in Semantic Web applications provide only goal-driven reasoning with an objective to identify the facts that support the conclusion (Antoniou and Bikakis, 2007). Their methodology doesn't provide any support to reasoning in an environment where conflicts may arise at run time such as in group decision making. When conflicts arise between the rules, these formalisms represent and handle only individual preferences in the form of priorities. These priorities are usually embedded in the derivation mechanism and competing rules are compared individually during the derivation process. Therefore, the derivation notion is bound to one single comparison criterion (defined by a single user) and fails to take into account the multiple factors that are important for making an informed decision.

To address the abovementioned drawbacks of defeasible reasoning-based Semantic Web applications, argumentation-based reasoning approaches in the existing literature have been discussed that take into account incomplete and/or contradictory information and reach an agreeable solution if possible. However, it is evident from the discussion in Section 2.7.2.2 that the Semantic Web and Web 2.0 are influenced by the philosophical view of argumentation, in which considerable emphasis is given to building arguments by human participation. Less importance has been given to monological argumentation, i.e. the construction of automated arguments and automated conflict resolution, and the acceptability of arguments by a reasoning engine to reach a conclusion. Therefore, such Semantic Web applications do not provide a solution for automated reasoning over underlying information. Therefore, there is need for a system to be equipped with monological argumentation with an automated built-in mechanism for argument construction and thereafter, through a reasoning process, identify and resolve conflicts and recommend a decision. However, such an approach has not been proposed in the existing literature. Hence, based on the above discussion, it can be inferred that existing Semantic Web-based approaches fail to provide a solution for reasoning over information that may be potentially incomplete and/or contradictory either within the enterprise or in other enterprises. In Chapter 3,

the problem associated with hybrid reasoning is formally identified and defined, and in Chapter 4, a solution to the problem defined in the existing literature is proposed as well as a hybrid reasoning methodology for argument construction and conflict resolution.

2.8.2.3 Lack of different argumentation-driven conflict resolution strategies

The need for a hybrid reasoning engine discussed in the previous section generates a set of arguments which may conflict with each other. An argument may attack its counter-argument and defeat it on the basis of certain criteria such as the strength or the weight of the argument i.e. the argument with more strength will defeat its counter-argument. The criteria to establish defeat between an argument and its counter-argument are also context dependent. As pointed out in Section 2.5.1, different argumentation frameworks have been proposed which use different defeat criteria. In the working environment of an enterprise, different kinds of Semantic Web applications operate and each may have different reasoning contexts as discussed in Section 1.2. In order to enable these applications to reason over information and resolve conflicts, different conflict resolution strategies are required so that each application can use its own conflict resolution strategy for the establishment of priority between an argument and its counter-argument. Hence, there is need for different argumentation-driven conflict resolution strategies, each using different criteria to establish defeat between an argument and its counter-argument.

In Chapter 3, the problem associated with conflict resolution between arguments is identified and defined, and in Chapter 4, a solution for the problem defined in the existing literature is proposed as well as different argumentation-driven conflict resolution strategies to address the need for different applications in an enterprise.

2.8.3 No methodology for knowledge integration or the graphical representation of the reasoning process and results to assist in enterprise-wide decision making

Semantic Web applications within enterprises today publish their information on the Web, either on their intranet or on the World Wide Web, which triggers the need to integrate the knowledge produced by the information systems of different enterprises to obtain a better picture of enterprise-wide decision making. However, current Semantic Web applications do not provide an enterprise-level knowledge integration methodology, especially when the results on a subject are potentially incomplete and inconsistent across the information systems of different enterprises. Most Semantic Web-based

reasoning engines differ from each other in the following aspects:

1. each has different knowledge-based representation;
2. each has different reasoning semantics;
3. each has a different output format.

This results in enterprises being unable to share, reason and integrate information coming from different Semantic Web applications either within the enterprise or in different enterprises. Additionally, the decision maker in an enterprise always need in depth visibility of the reasoning process in order to take into account the rationale behind the conclusion and make appropriate decisions. The monotonic reasoning systems discussed in Section 2.7.1, and the non-monotonic reasoning-based system discussed in Section 2.7.2 provide no visibility or information about the reasoning process or how results are reached. Similarly, they provide no graphical representation of the reasoning process in the form of a reasoning chain which can help the decision maker trace the path from the evidence to the final conclusion and easily identify the basis on which the decision was reached.

Hence, it is evident from the discussion above that there is need for a methodology that provides a solution for knowledge integration which depicts the reasoning process in a graphical representation format in order to provide a better analysis environment for the decision maker so that appropriate decisions can be made. In Chapter 3, the problem associated with knowledge integration is formally identified and defined, and in Chapter 4, a solution for the problem defined in the existing literature is proposed, as well as a methodology for knowledge integration and its graphical representation to assist the decision maker in enterprise-wide decision making.

2.9 Conclusion

In this chapter, a survey of the existing literature on argumentation and its adoption in the fields of Philosophy and AI was presented. Also, a critical analysis of existing reasoning approaches deployed on the Semantic Web was given which categorised them as either monotonic or non-monotonic reasoning. It is evident from a critical evaluation of existing reasoning approaches that monotonic reasoning has a number of limitations that inhibit their ability to reason over information that could be potentially incomplete and contradictory. Non-monotonic reasoning, especially defeasible reasoning, is a good option but it works under certain constraints which curtail its adoption in Semantic Web applications for business intelligence.

Chapter 3 - Problem Definition

3.1 Introduction

In Chapter 2, a review of the literature on argumentation, philosophical and logic-based models, argumentation frameworks and implementations was presented. The current reasoning approaches being employed in Semantic Web applications were also discussed and categorised. Several advancements that have been made in terms of reasoning on information on the Semantic Web were outlined. Of the various Semantic Web applications, defeasible reasoning-based systems are capable of addressing the issues of representation and reasoning about information that could be incomplete and/or contradictory. However, as pointed out in Chapter 2, defeasible reasoning-based systems work under certain constraints such as a definition of individual preferences and hard-coding them in Semantic Web application at compile time. After the critical evaluation of existing reasoning approaches on the Semantic Web, the different research gaps that need to be addressed in order to develop a framework for the representation of incomplete and/or contradictory information, reasoning and integration in Semantic Web applications were outlined.

In this chapter, the problem to be addressed in this thesis is formally defined, and the different research issues are identified and transformed into research objectives, using science and engineering methodologies to address the research objectives. In the next section, the key concepts which are used this point forward in this thesis are outlined.

3.2 Key concepts

This section presents the definition of important terms used throughout this thesis.

Enterprise

A business entity that makes use of software applications to gain better insights from the existing information and guide their business activities.

Decision maker

A person who carries out the decision making process by using software applications in an enterprise.

Information

The term 'information' refers to information in the following format:

- the production rules governing the inference mechanism, and
- the facts over which the inference mechanism is being applied.

Reasoning

Reasoning is a cognitive process of looking at reasons for beliefs, conclusions and actions.

Web-based Intelligent DSS

A Semantic Web-based application that captures information and performs reasoning by making use of high level software intelligence to provide decision support to the decision maker in the decision making process.

Defeasible logic programming (DeLP)

This is a formalism that combines the results of Logic Programming and Defeasible Argumentation. DeLP provides the possibility of representing information in the form of rules such as strict and defeasible rules in a declarative manner.

Knowledge base

The collection of production rules represented using DeLP are saved in the rule base and the facts/evidence represented using DeLP are saved in the working memory. The working memory and rule base are collectively known as the knowledge base.

Hybrid reasoning engine

This performs hybrid reasoning over information saved in the knowledge base. Hybrid reasoning comprises two types of reasoning: data-driven reasoning for argument construction; and goal-driven reasoning for conflict identification between arguments and their resolution.

Reasoning chain

This is the output of a hybrid reasoning engine integrated in the form of a chain. It links the facts to the conclusions drawn. Its graphical representation helps the decision maker to better understand the reasoning results. The process of generating the reasoning chain is called information integration.

Integration scheme

This is an argumentation scheme consisting of the decision maker's criteria to determine if the reasoning chain adheres to the prerequisite requirements for the decision-making process.

Valuation of reasoning chain

This involves the execution of the integration scheme on a reasoning chain. During this process, all the premises and critical questions originating from the integration scheme are executed on the reasoning chain under consideration. The resulting reasoning chain which either passes or fails the test is called the valued reasoning chain.

Integrated recommendations space

The integration of the valued reasoning chains that passed the test through hybrid reasoning results in the generation of an integrated recommendation space. The process of generating an integrated recommendations space is called knowledge integration.

Rule Markup Language (RuleML)

This is a standard format to ensure compatibility of rules among different Semantic Web applications. It supports the representation of different rule types and its syntax has been extended to express defeasible rules and superiority relations (Bassiliades et al., 2004; Pham et al., 2008).

Argument Interchange Format (AIF)

This is an international effort to develop a representational mechanism for exchanging argument resources between research groups, tools, and domains using a semantically rich language (Chesnevar et al., 2006a; Iyad Rahwan, 2009; Rahwan et al., 2007b). The AIF was developed as a commonly agreed upon core ontology. The AIF ontology specifies the basic concepts used to express arguments and the relationship between arguments.

3.3 Problem definition

As mentioned in earlier chapters, the invention of the Internet has transformed the working environments of enterprises and now, the majority of business activities, one way or the other, are performed via the Internet. A number of software applications have been developed which assist enterprises to transform the information in their database/knowledge base into useful knowledge that assists them in their decision-making process. With the advent of Web 2.0, a vast amount of information is being produced over the WWW which is of interest to enterprises in their decision-making processes (such as customer reviews about the products and services of an enterprise, published business policies of collaborating enterprises, discussion on the required features of new products and services) and the introduction of the Semantic Web helps their systems to understand and consider such information during the decision-making process. By doing so, the decision makers can gain better insight into the available information, leading to either improvement in their working environments, products and services or to a successful collaboration with their partners.

As pointed out in Section 1.2, information on the WWW outside of an enterprise's boundaries can be incomplete and/or contradictory. So, to understand and consider such information during the decision-making process, there is a need to design and develop intelligent Semantic Web applications that can transform the incomplete and/or contradictory information on the WWW into useful knowledge along with qualitative insights, so that decision makers can expect a human style of reasoning in software applications (Carlsson and Turban, 2002; Shim et al., 2002). However, as pointed out in Sections 1.5 and 2.8, the current Semantic Web development technologies defined at the logic layer to infer knowledge do not provide any support to represent, reason and integrate incomplete and/or contradictory information. ***As a result, current enterprises cannot exploit the information on the WWW outside of their boundaries for decision-making processes. This triggers the need for enterprises to have a logic-based framework that can take into account***

incomplete and/or contradictory information on the WWW and transform it into useful knowledge that in turn, assists their decision-making process to achieve BI.

To realize such a logic-based framework for informed decision making, the primary objective is to represent the underlying information in a declarative format to drive the reasoning process in Semantic Web applications. As highlighted in Section 1.1, attempts have been made by different researchers to annotate the information on the WWW with user-defined ontologies and then perform reasoning on top of it. As discussed in Section 2.7.1.1, ontology-based reasoning approaches are an outcome of research efforts in this regard. However, due to their limited expressiveness and reasoning capabilities, they have been extended with different kinds of rule-based languages such as N3Logic and SWRL defined at the logic layer of the Semantic Web as discussed in Section 2.7.1.2 . Though these rule-based languages are capable of introducing more advanced reasoning capabilities in Semantic Web applications to address complex problems, as pointed out in Section 2.8, they do not provide any support to represent incomplete and/or contradictory information. Some attempts have been made in the literature to make use of defeasible reasoning to represent incomplete and/or contradictory information as discussed in Section 2.7.2.1 . Though this provides a solution for considering incomplete and/or contradictory information coming from a single source, during information representation, they assume that if a contradiction exists between any two rules, then a priority over them should be specified manually for their successful execution. Furthermore, they are not capable of information representation when it comes from different sources exists within an enterprise or in other enterprises. In order to address such shortcomings, ***there is need for a rule-based language that extends existing defeasible reasoning-based rule languages and can represent incomplete and/or contradictory information exists within an enterprise or in other enterprises to assist the decision maker in the decision making process.*** From this point onwards, such a rule-based language is referred to as the system's rule-based language.

Once the issue of the representation of information has been solved, then the next challenging task is the sharing of rules among different Semantic Web applications, both internal and external to an enterprise. As pointed out in Section 2.8, each Semantic Web-based reasoning engine has its own syntax and semantics for rule representation, leading to rule inter-operability issues among different applications. This makes the sharing of the rules a challenging task across different applications that need to communicate with each other in order to process a task. In Section 1.1.2.1, the importance of RuleML to ensure compatibility of rules among different Semantic Web applications is highlighted. ***Therefore, there is a need for a translation***

mechanism that can translate the rules defined in a standard format to the system's rule-based language format.

Once the information is present in the form of rules, then the next important step is to transform the facts that drive the reasoning process into a format suitable for the execution of rules defined in the system's rule-based language. As pointed out in Section 1.1.1.1, the standards for sharing facts on the Semantic Web are in OWL/RDF format. However, facts on the WWW can be in textual form (unstructured information) or structured data in traditional SQL relational databases. These facts needs to be transformed into a standard format so that they can be used by Semantic Web applications. For facts in an unstructured format, there are different tools and technologies available that either semantically annotate unstructured text e.g. the Knowledge and Information Management (KIM) platform, and provide data in OWL/RDF file format. To transform structured information into a standard format, the D2RQ platform enables applications to access an RDF-view on structured data i.e. a non-RDF database such as SQL databases, through a rule engine API's over the Web via the SPARQL Protocol and as Linked Data. Once the data is available in OWL/RDF, it can be used in different Semantic Web application. However, such data follows an XML format which cannot be used directly by the rules defined in a systems's rule-based language. This data needs to be translated into a format suitable to execute rules defined in a system's rule-based language. Therefore, there is a need for ***a translation mechanism to translate facts defined in OWL/RDF format into a format that can be exploited by the system's rule-based language***.

Once information representation is achieved, the framework needs to have the capability to perform reasoning on it. As pointed out in Section 2.8.2 , Semantic Web-based reasoning engines follow the monotonic reasoning approach and are not capable to reason over information that may be incomplete and/or contradictory. Of the three approaches discussed in Section 2.7.2.1 that consider defeasible reasoning-based reasoning in Semantic Web applications, there is a single attempt based on Situated Courteous logic (Grosof et al., 2002) that carries out data-driven reasoning. However, during the reasoning process, if a conflict arises, the reasoning engine removes the information with the lower priority from the knowledge base in order to keep the remaining information consistent. Such a loss of information during the reasoning process does not provide deep insights into the choices made during the decision-making process. Additionally, it uses a Rete network for the compilation of rules and works in close coordination with the working memory. This results in the system having greater efficiency. As pointed out in Section 2.8.2.1, using the Rete network for data-driven reasoning on incomplete and/or contradictory information brings the following challenges:

1. The current Rete network works only for predicate logic-based rule languages that follow monotonic reasoning. Therefore, it has to be extended to represent incomplete and/or contradictory information as Rete nodes.

2. A Rete network only executes one rule in a single match-execute cycle. If two rules are activated, then only the rule that has a higher preference order (specified at compile time) will be executed. However, reasoning on incomplete and/or contradictory information may result in the activation of more than one rule, which may represent different viewpoints in relation to the issue at hand. These can be called arguments and the construction of arguments by the reasoning engine is a basic characteristic of the argumentation process to drive the reasoning process to a certain conclusion. As highlighted in Section 2.8.2 , current reasoning engines are not capable of performing such reasoning.

Therefore, ***the reasoning engine needs to be able to perform data-driven reasoning capable of arguments construction from information saved in the knowledge base and maintain them as an arguments set. The arguments in an arguments set may contradict with each other.***

A reasoning engine should also have the capability to resolve conflicts using different criteria, either automatically or being guided by the members of the decision making group to achieve their goals. As discussed in Section 2.7.2.1, the use of defeasible reasoning-based attempts in Semantic Web applications enables goal-driven reasoning, however, their objective is to identify the evidence that supports the conclusion. Their reasoning methodology doesn't provide any support for reasoning in an environment where conflicts may arise at run time, such as in group decision making. Additionally, when conflicts arise between rules, these formalisms represent and handle only individual preferences in the form of priorities. These priorities are usually embedded in the derivation mechanism and competing rules are compared individually during the derivation process. Therefore, the derivation notion is bound to one single comparison criterion and fails to take into account the multiple factors that are important for making an informed decision. The most suitable solution is provided by argumentation formalisms that have been recognized as a potential contender for capturing the discussion among Web users on the WWW. A number of Web applications have been built that enable users to represent their arguments about an issue in discourse and to become involved in an argumentative discussion. However, as pointed out in Section 2.8, due to the lack of reusable components from AI research, most of these applications follow philosophical argumentation-based frameworks where reasoning is performed by humans to cogitate and evaluate arguments and to take action. Less importance has been given to building automated arguments and furthermore, automated conflict

resolution and the acceptability of arguments to reach a conclusion. ***Therefore, the reasoning engine needs to have goal-driven reasoning that is capable of performing argumentation-driven conflict resolution. From this point onward, such a reasoning engine is referred to as a hybrid reasoning engine as it supports both kinds of reasoning, namely data-driven reasoning and goal-driven reasoning***.

It is important to note that during the argumentation process for conflict resolution, an argument may attack its counter-argument and defeat it on the basis of certain criteria such as the strength or the weight of the arguments. The argument with more strength will defeat its counter-argument. The criteria to establish defeat between an argument and its counter-argument are also context dependent. As pointed out in Section 2.6, different argumentation frameworks have been proposed which use different defeat criteria. In the working environment of an enterprise, different kinds of Semantic Web applications operate and each may have different reasoning contexts. In order to enable these applications to reason over information and resolve conflicts, different conflict resolution strategies are required so that each application can use its own conflict resolution strategy for the establishment of priority between an argument and its counter-argument. ***Therefore, argumentation-driven conflict resolution needs to be extended with different conflict resolution strategies, each using different criteria to establish defeat between an argument and its counter-argument.***

One of the important features required of reasoning engines is an ability to answer queries, with an explanation of how a particular conclusion was reached. The queries can be simple queries such as whether information is true/false or they can be complex. As pointed out in Section 2.7.2.1, the approach taken by Grosof et al. (2002) removes information from the knowledge base that causes a contradictory situation which results in a loss of information that may provide an explanation of the results. Though defeasible logic-based implementation retains contradictory information in their knowledge bases, they do not provide solutions for group decision making. In such cases, conflict resolution involves the preferences of each member and results are achieved after thorough discussion. This is much like argumentation-driven approaches where decisions are backed by justifications. ***Therefore, the hybrid reasoning engine needs to have a querying and answering capability backed by an explanation of conflict resolution and/or the conclusions drawn.***

Another important feature missing in current Semantic Web applications is an ability to make reasoning transparent and easily comprehendible for the decision maker. The decision maker in an enterprise always needs an in-depth understanding of the reasoning process in order to take into account the rationale behind the conclusion

and make appropriate decisions. The monotonic reasoning systems discussed in Section 2.7.1, and the non-monotonic reasoning-based systems discussed in Section 2.7.2 provide no information about the reasoning process and how results are reached. In other words, they do not provide a trail to show how conclusions are reached in the form of a chain, known as a reasoning chain. Additionally, the reasoning process in an enterprise is becoming very complex which requires the reasoning results to be depicted in graphical format to enable a deeper insight and ease of comprehension of the obtained results. Currently, there is no framework that provides a graphical representation of a reasoning chain that can help the decision maker to track the path from the identification of the evidence to the final conclusion and easily identify the basis on which the decision was reached. Therefore, ***there is a need for a methodology that can integrate the output of a hybrid reasoning engine in the form of a reasoning chain that links the facts to a conclusion. Additionally, it should have the functionality to depict the reasoning chain in a graphical format.***

The framework also needs to have a mechanism to export the results of the reasoning engine in a form that is shareable on the WWW and among other enterprise applications. As pointed out in Section 1.4, the Argument Interchange Format (AIF) is an international effort to develop a representational mechanism to exchange argument resources between research groups, tools, and domains, using a semantically rich language (Chesnevar et al., 2006a; Iyad Rahwan, 2009; Rahwan et al., 2007b). The AIF was developed as a commonly agreed upon core ontology. The AIF ontology specifies the basic concepts used to express arguments and the relationship between arguments. ***The generic framework needs a mechanism to annotate the output of the hybrid reasoning engine i.e. the reasoning chain, with the AIF ontology so that it can be used by different applications, either internal or external to the enterprise.***

In an enterprise, various applications are expected to work together to support information exchange, processing, and integration. The results produced by one application may need to be integrated with results of other applications. Such integration of results about a subject is known as knowledge integration. It is important to note that the results of an application may contradict the results of other applications. To explain further, consider a simple example where management asks its departments to provide recommendations on a particular issue. It is possible that each department's recommendations may contradict the recommendations of others. Current defeasible logic-based Semantic Web applications do not provide a solution for knowledge integration. In the literature, argumentation schemes have been proposed that provide a solution to knowledge integration. ***Therefore, the framework needs a methodology, driven by argumentation schemes, to integration knowledge***

that comes from different hybrid reasoning engines into a single reasoning chain to facilitate enterprise-wide decision making.

To the best of my knowledge, there is no proposed framework in the literature that addresses the shortcomings of current defeasible logic-based implementation in the domain of the Semantic Web. Hence, there is need for a framework that can represent reason and integrate information that is incomplete and potentially contradictory.

The above description of the problem points to the proposal of a complete framework for the representation, reasoning and integration of incomplete and contradictory information exists within an enterprise and/or in other enterprises. Based on the above overview and description of the issues, the problem that will be addressed in this thesis is defined as follows:

"Design and development of a generic framework for monological argumentation in Semantic Web applications. Such a framework can be exploited for the development of different Semantic Web applications to represent, reason and integrate information exists within an enterprise and/or in other enterprises for enhanced business intelligence as discussed in Section 1.2".

3.4 Research issues

As previously discussed, the Semantic Web provides solutions for enterprises to exploit the information on the WWW outside their boundaries. However, it is important to note that such information may be incomplete and/or contradictory. As pointed out in Sections 1.5 and 2.8 , the logic layer of the Semantic Web plays an important role where rule-based technologies have been given paramount importance for the development of advanced reasoning capabilities in Semantic Web applications to address complex problems. They extract, transform and integrate information in a platform-independent manner. However, current research does not provide a solution to represent reason and integrate information, either internal or external to an enterprise. Based on the critical evaluation of the existing literature review, the following research issues have been identified:

1. Semantic Web development technologies follow monotonic logic, hence they are incapable of representation and reasoning over incomplete and/or contradictory information. The use of defeasible reasoning-based implementations is not capable to drive reasoning in group decision making scenarios where conflicts may arise among members of the teams during the decision making process.

2. Most Semantic Web-based reasoning engines have their own format for rule and fact expressions. As a result, they cannot be shared or exchanged with other Semantic Web applications, either internal or external to an enterprise. In addition, Semantic Web-based reasoning engines either provide data-driven reasoning or goal-driven reasoning which limits their capability to transform information by reasoning into integrated knowledge.

3. There are some good non-monotonic techniques and technologies in logic programming and argumentation formalisms for handling incomplete and/or contradictory information. However, they have not yet been applied in the area of the Semantic Web.

4. There is no framework that provides a graphical representation of the reasoning process to non-technical decision makers.

5. There is no proposed methodology for the integration of reasoning results i.e. knowledge integration originating from different Semantic Web applications.

In next section, the research objectives of this study are outlined in order to address the research issues identified above.

3.5 Research objectives

The objective of this research is to propose, develop, validate and evaluate a generic framework to provide monological argumentation support in Semantic Web applications. In order to address the primary objective, the research objective can be broken down into the following sub-objectives:

3.5.1 To propose a methodology for incomplete and/or contradictory information representation

1. To propose a rule-based declarative language for incomplete and/or inconsistent information representation on the Semantic Web. Such information representation will enable the information provided by Web users i.e. specifications or preferences, to be taken into account by Web applications and considered in the reasoning process to produce customized results for the decision maker.

2. To propose a translation mechanism to translate the information defined in RuleML to the system's rule-based declarative language. Such a translation will

enable the exploitation of information already existing on the Semantic Web, specified in RuleML.

3. To propose a translation mechanism to translate the data defined on the Semantic Web in the form of OWL/RDF into the system's rule-based declarative language format, keeping the semantic information intact. The translated data are exploited by the rules during the reasoning process.

3.5.2 To propose a methodology for monological argumentation driven-reasoning engine to reason over incomplete and/or contradictory information

1. To propose a hybrid reasoning engine to reason over information represented in the system's rule-based declarative language. The hybrid reasoning engine performs two types of reasoning, firstly, data-driven reasoning for arguments construction, and secondly, goal-driven reasoning for conflicts identification between arguments and their resolution.

2. To propose different conflict resolution algorithms to resolve conflicts between arguments and their counter-arguments. Each conflict resolution algorithm should take into account different conflict resolution criteria in order to address different contexts in the Semantic Web applications.

3.5.3 To propose a methodology for information and knowledge integration

1. To propose a mechanism to integrate the information (i.e. output) of the hybrid reasoning engine and provide its graphical representation to the decision maker for a better understanding of the reasoning process and the results.

2. To propose a mechanism to export reasoning chains in a standard format to other Semantic Web applications and vice versa. This will help to bring inter-operability among different Semantic Web applications and pave the way for knowledge integration.

3. To propose a mechanism to query the reasoning results and obtain an explanation of the results.

4. To propose a methodology to integrate the reasoning chains produced by different Semantic Web applications into a coherent reasoning chain i.e. knowledge

integration.

3.5.4 To exploit the power of a generic framework in different Semantic Web applications as follows:

3.5.4.1 To design and develop an Argumentation-enabled Web-based IDSS (Web@IDSS) for handling structured information

1. Using a case study, identify the importance of Web@IDSS for business intelligence.
2. To propose a conceptual framework for Web@IDSS in order to represent and reason over structured information that may be incomplete and/or contradictory and exists within the enterprise and/or in other enterprises.

3.5.4.2 To design and develop an Argumentation-enabled Web-based IDSS (Web@KIDSS) for knowledge Integration.

1. Using a case study, identify the importance of argumentative reasoning and argumentation schemes for knowledge integration.
2. To propose a conceptual framework for Web@KIDSS in order to integrate the information/results generated by different Semantic Web applications in an enterprise to support intelligent decision making.

3.5.4.3 To design and develop an Argumentation-enabled Web-based IDSS for handling unstructured information

1. To propose and develop a domain ontology for annotation of unstructured information.
2. Using a case study, identify the process for considering unstructured information using the proposed framework by taking into account the business polices of an enterprise or two or more collaborating enterprises. Make use of knowledge representation approach with argumentative reasoning for process map discover from unstructured business policies (KR@PMD).
3. To propose a conceptual framework for KR@PMD by exploiting the power of the generic framework.

3.5.5 To validate and evaluate the proposed framework

1. To validate the functionality of proposed framework for monological argumentation support in Semantic Web applications with the help of case studies and development of Web-based IDSSs

2. To evaluate the proposed framework by performing feature evaluation of Web-based IDSSs identified in above with the existing contemporary software applications.

3.6 Research approach to problem solving

In addressing the stated problem, this thesis focuses on the development and subsequent testing and validation of a methodology defeasible logic programming-based framework for argumentation support in Semantic Web applications. In order to propose a solution for the research issues listed in the previous section, a systematic scientific approach needs to be followed in order to ensure the methodology development is scientifically-based. Therefore, this section gives an overview of the existing scientifically-based research methods and give reasons for choosing a particular research method in this research.

3.6.1 Research methods

There are two broad categories of research approaches, namely

1. the science and engineering approach;

2. the social science approach.

Science and engineering-based research is concerned with confirming theoretical predictions. Cohen (1987) states that in the engineering field, the spirit of 'making something work' is essential and has three levels:

- Conceptual level (level one): creating new ideas and new concepts through analysis.

- Perceptual level (level two): formulating a new method and a new approach through designing and building the tools, environment or system through implementation.

- Practical level (level three): carrying out testing and validation through experimentation with real world examples, using laboratory or field testing.

The science and engineering-based research approach may lead to new techniques, architectures, methodologies, devices or a set of new concepts which together will form a new theoretical framework. It not only addresses the issue of what problems need to be solved, it also proposes a solution.

Social science research methods may be categorised as either quantitative or qualitative. Quantitative research involves extensive data gathering usually using methods such as surveys and statistical analysis of the gathered data in order to prove or disprove various hypotheses that have been formulated. Qualitative research involves in-depth structured or semi-structured interviews that allow one to pursue particular issues of interest that may arise during the interviews. It does not normally involve a large sample of data and the information gathered may not be in a form that readily allows statistical analysis. A typical social science research approach, the use of survey forms, is used to identify problems which are subsequently formulated as hypotheses. The goal of social science research is to obtain evidence to support or refute a formulated hypothesis (McTavish and Loether, 1999; Burstein and Gregor, 1999; Nunamaker et al., 1990) . The research assists the researcher to understand people and social issues, such as culture, within the area of research. Kaplan and Maxwell (2005) argues that the ability to understand a phenomenon within its social and cultural context is forfeited when textual data results are quantified. This kind of research can indicate the extent to which the methodology is or is not accepted and sometimes may be able to give a reason for this. However, unlike engineering-based research, this type of research does not explain what a methodology should be and how to produce a new methodology for problem solving. This research only tests or evaluates a method that has already been produced from science and engineering research.

This thesis deals with the development of a new generic, logic-based framework to support argumentative in Semantic Web applications. Therefore, this research clearly falls into the science and engineering research domain.

3.6.2 Choice of science and engineering-based research method

In this thesis, a science and engineering-based research approach was chosen as the research method for the proposed solution development. An overview of this research method is depicted in Figure 3.1

The research commenced with the identification of the research problems. Then, the relevant literature on topics related to this study were analysed. Based on an

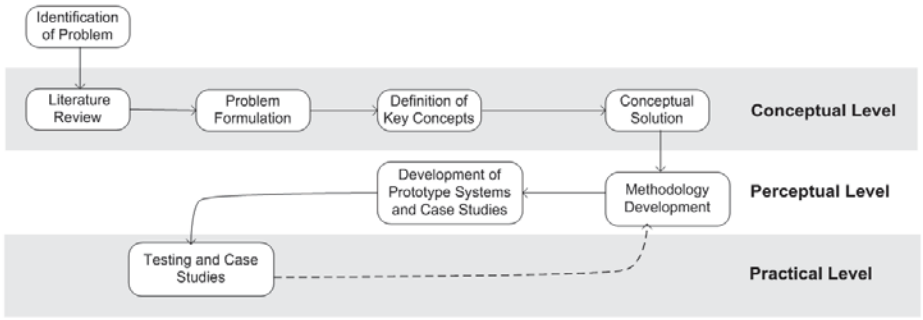


Figure 3.1: Overview of science and engineering-based research method

extensive review of the existing literature, the problem which needs be addressed was formulated. Subsequently, the key concepts to address the problem were defined, taking into account the characteristics of the interaction. These definitions are used when developing the conceptual solution. After this, the conceptual solution to the problem being addressed in this thesis was formulated. All processes from the literature review to the conceptual solution are included in the conceptual level. At the perceptual level, the methodology for argumentation support in Semantic Web applications was developed by taking into account the representation, reasoning and integration of incomplete and/or contradictory information. After this, the prototype systems were engineered and some case studies were developed which are used later to test the proposed methodology. The processes of methodology development and development of prototype systems and case studies constituted the perceptual level of this work. Once the prototype systems had been engineered, they were used together with the developed case studies to validate the proposed methodology. At the practical level, based on the results obtained, the proposed methodology was evaluated and validated. Based on this, the proposed methodology was then fine-tuned. The process of evaluation and validation of the developed methodology constitutes the practical part of this work.

With regard to research output evaluation and validation (Practical), Nunamaker et al. (1990) argues that typical research follows a pattern of problem definition, hypothesis, analysis and argument. In such a scenario, problems are encountered and an analysis is performed in the form of proofs and developed solutions. The results of the analysis and development form the basis of the evaluation of research outcomes. The methodology proposed by Nunamaker et al. (1990) consists of the problem definition, conceptual solution and system prototype processes. This research will adopt the research method proposed by Nunamaker et al. (1990) for the validation and verification of my research output, through proof of concept.

3.7 Conclusion

In this chapter, the formal definition of the problem to be addressed in this thesis was presented. The identified problem was subsequently decomposed and discussed as a set of seven cohesive research objectives in order to address the problem being examined in this thesis. Furthermore, the proposal to implement a science and engineering research methodology in conjunction with the research methodology that uses system development as an information system research methodology was outlined.

Chapter 4 - Solution Overview

4.1 Introduction

As highlighted in Chapters 1 and 3, enterprises need to exploit the information on the WWW to gain better insights into the decision making process by considering the wide spectrum of information available. In order to do this, Semantic Web applications need to have the capability to capture, represent, reason and integrate information from different sources which may be incomplete and/or contradictory. To overcome this challenge, the different research objectives that need to be addressed were elaborated in Section 3.5 . In this chapter, the proposed solution to address these objectives, that is, a logic-based framework that supports monological argumentation in Semantic Web applications (GF@SWA) to enable them to represent, reason and integrate information from different sources which could be incomplete and/or contradictory and utilize this in the decision making process is described in detail.

This chapter is organised as follows: In Section 4.2, a brief overview of the proposed logic-based framework that supports argumentation in Semantic Web applications is given. The framework comprises three layers namely, the *information layer*, the ***a****rgumentation driven* ***i****nformation* ***r****epresentation,* ***r****easoning and* ***i****ntegration* (@IRRI) *layer* and the *applications layer*. In Sections 4.3 - 4.5, each layer of the framework is discussed in detail and an overview of the proposed solution to achieve the objective of each layer is given. Section 4.6 outline three Semantic Web application that exploit the proposed logic-based framework to represent, reason and integration information in order to assist decision maker in decision making process. Section 4.7 concludes the chapter.

4.2 Solution overview for logic-based framework that supports argumentation in Semantic Web applications (GF@SWA)

Figure 4.1 represents the proposed solution architecture of GF@SWA for incomplete and/or contradictory information representation, reasoning and integration in the Semantic Web applications of an enterprise. The solution architecture has the following three layers:

1. Information layer

   The information layer represents the information sources containing information, either in a structured or unstructured format exists within an enterprise and/or in other enterprises published over WWW. Semantic Web applications can consume information from these sources to facilitate the decision making process.

2. @IRRI layer

   This layer comprises a logic-based framework to support monological argumentation in Semantic Web applications for BI by demonstrating the following functionalities:

   (a) representation of incomplete and/or contradictory information;

   (b) hybrid reasoning over underlying information;

   (c) integration of reasoning results produced by a reasoning engine. This process is known as information integration;

   (d) import/export of the integrated information to other Semantic Web applications;

   (e) integration of results produced by different reasoning engines. This process is known as knowledge integration.

3. Applications layer

   This layer contains of a set of different Semantic Web applications that exploit the functionality of the @IRRI layer to represent, reason and integrate incomplete and/or contradictory information present on the information layer in order to support decision makers in their decision making processes.

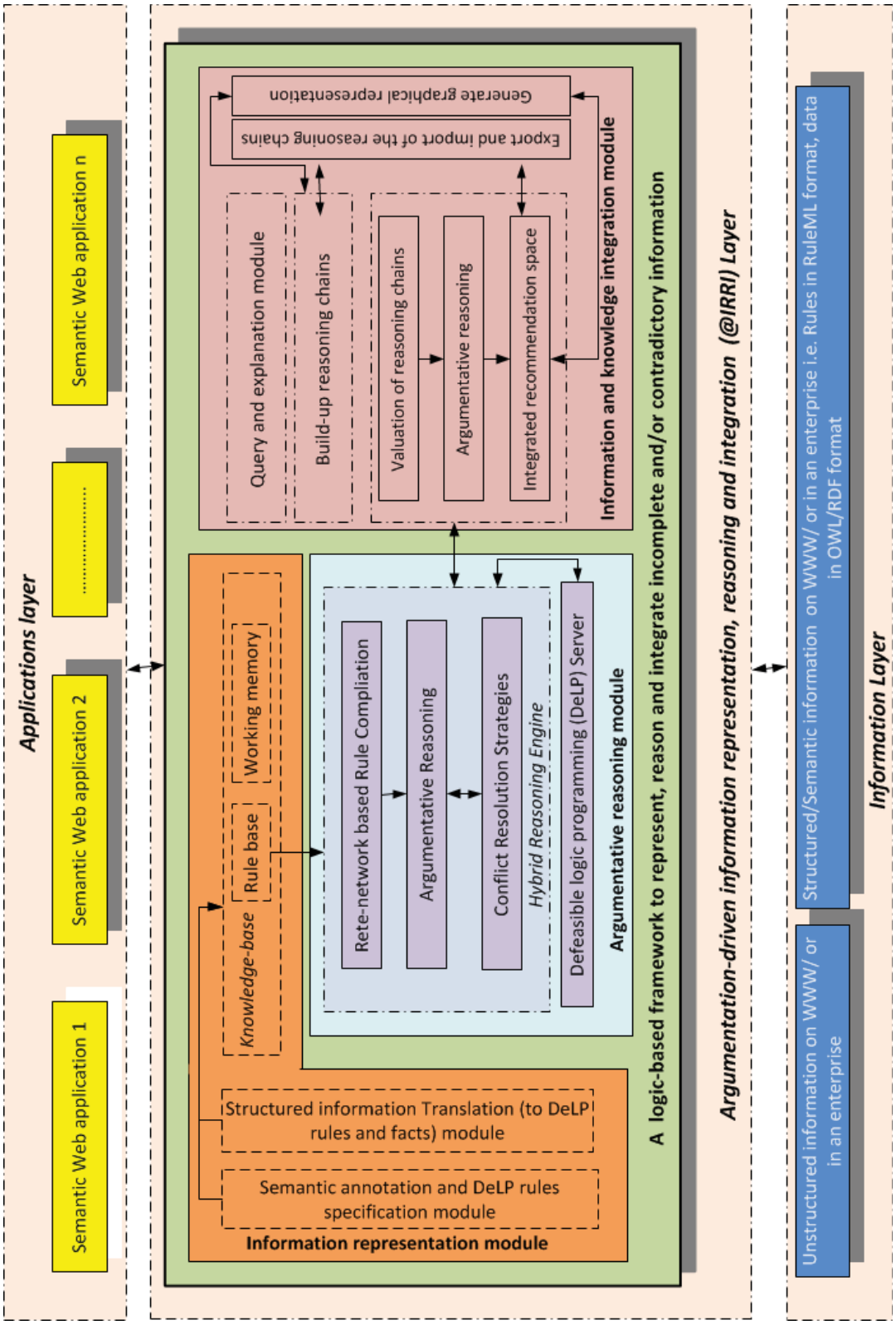


Figure 4.1: Solution overview of GF@SWA to support argumentation in Semantic Web applications

Figure 4.2 is a flowchart diagram depicting the working of the overall proposed solution and how the information on the WWW is taken into account and used by Semantic Web applications in the decision making process. The proposed solution comprises the following three important steps:

1. Semantic Web applications (located at the applications layer) take into account the information (located at the information layer) to provide a wider insight for decision makers and assist them in their decision making process. To achieve this objective, Semantic Web applications exploit the functionality of the *information representation module* of the logic-based framework located at the @IRRI layer. The information representation module helps them to transform the information on the WWW into a format suitable for reasoning and save it in their knowledge bases.

2. To perform reasoning over the information in the knowledge base, the *argumentative reasoning module* of the logic-based framework provides a hybrid reasoning engine. The hybrid reasoning engine performs hybrid reasoning: data-driven reasoning on the underlying information to infer new knowledge; and goal-driven reasoning to identify any conflicts that may arise during the reasoning process and solves them using different conflict resolution strategies.

3. After conflict identification and resolution, the *information and knowledge integration module* of the logic-based framework integrates the output of the reasoning process in the form of a reasoning chain. This module also helps Semantic Web applications to share their reasoning results in the form of a reasoning chain with other Semantic Web applications by exporting them in a standard format. The last function performed by this module is knowledge integration, which involves the integration of different reasoning chains (may be imported from different reasoning engines) into an integrated knowledge chain after resolving any conflicts that may arise during this process.

In the next sections, the working of each of the three layers defined in the overall architecture solution to solve the research problem is discussed in detail.

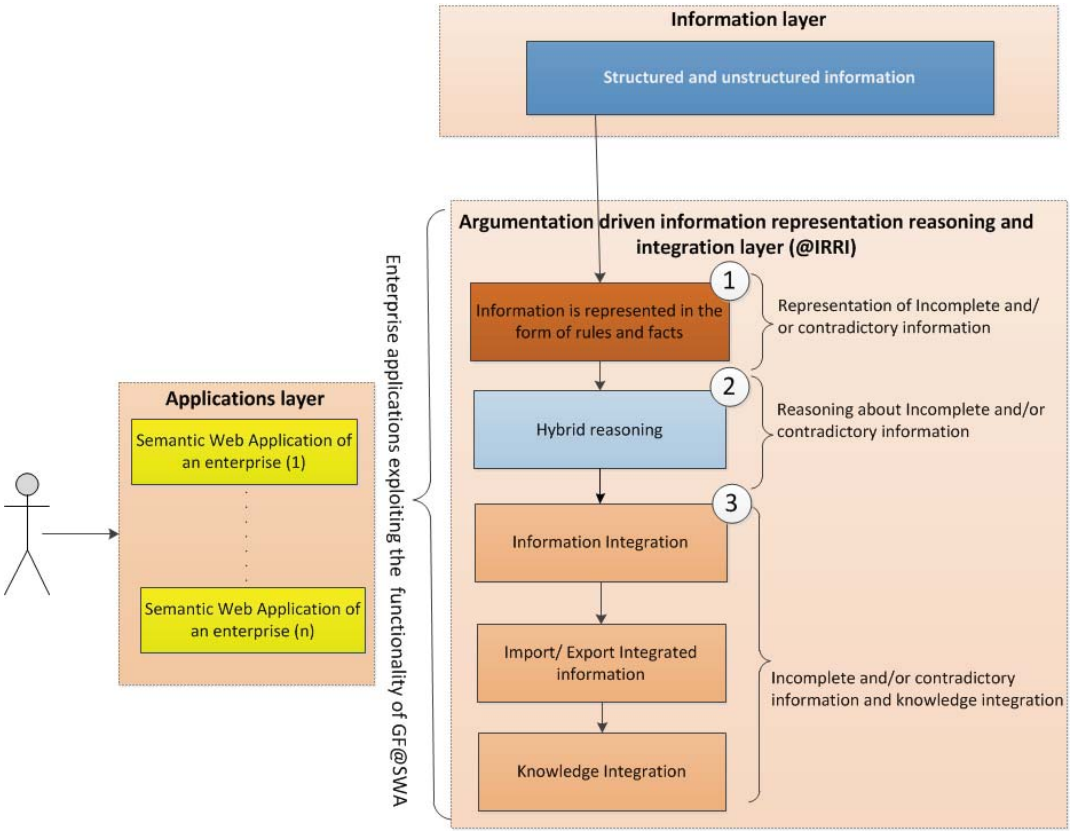


Figure 4.2: Working of the proposed solution for Information representation, reasoning and integration by Semantic Web applications

4.3 Information layer

The information layer represents the information on the Semantic Web, both in an unstructured and structured format and is the primary source of input to Semantic Web applications. The unstructured information on the WWW is in the form of text; both on Web pages and in enterprise policy documents. The structured information on the Semantic Web is categorized as follows:

- static information which remains consistent over a period of time and it is composed of OWL/RDF ontologies, and
- dynamic information which changes over a period of time according to the business needs and strategies such as business polices, business contracts etc., leading to possible conflicts among the underlying information. This information is represented in the RuleML format.

Once the information on the WWW has been identified, that if considered give better insight in the decision making process of an enterprise, then Semantic Web

applications need to exploit the functionality of the @IRRI layer to achieve their desire objective. In the next section, the @IRRI layer is discussed in detail.

4.4 Argumentation-driven information representation, reasoning and integration layer (@IRRI)

In this section, an overview of the @IRRI layer that comprises the logic-based framework to represent, reason and integrate information that could be incomplete and/or contradictory is given. To enable Semantic Web applications to exploit this layer, the overall proposed solution is divided into the following sub-solutions:

1. Solution for incomplete and/or inconsistent information representation.
2. Solution for hybrid reasoning engine with different argumentative-driven conflict resolution strategies.
3. Solution for information and knowledge integration.

In the following section, each of these sub-solutions is described in detail.

4.4.1 Solution for incomplete and/or contradictory Information representation

As discussed in Chapter 3, the current rule-based languages used for information representation in Semantic Web applications follow Open World Assumptions (OWA), as a result of which they are incapable of dealing with incomplete and/or contradictory information. This drawback is addressed in this solution with the help of the information representation module of the logic-based framework. The proposed solution provides the following three possible ways to represent incomplete and/or contradictory information in Semantic Web applications as depicted in Figure 4.3:

1. Specify rules and facts from scratch using the system's rule language format and save them in a knowledge base. In Chapter 5, the specification of rules from scratch using Web-based forms is discussed in detail.
2. Use a translator to translate the structured information and save the translated information in the knowledge base. In Chapter 5, the working of the proposed translators is outlined in detail.

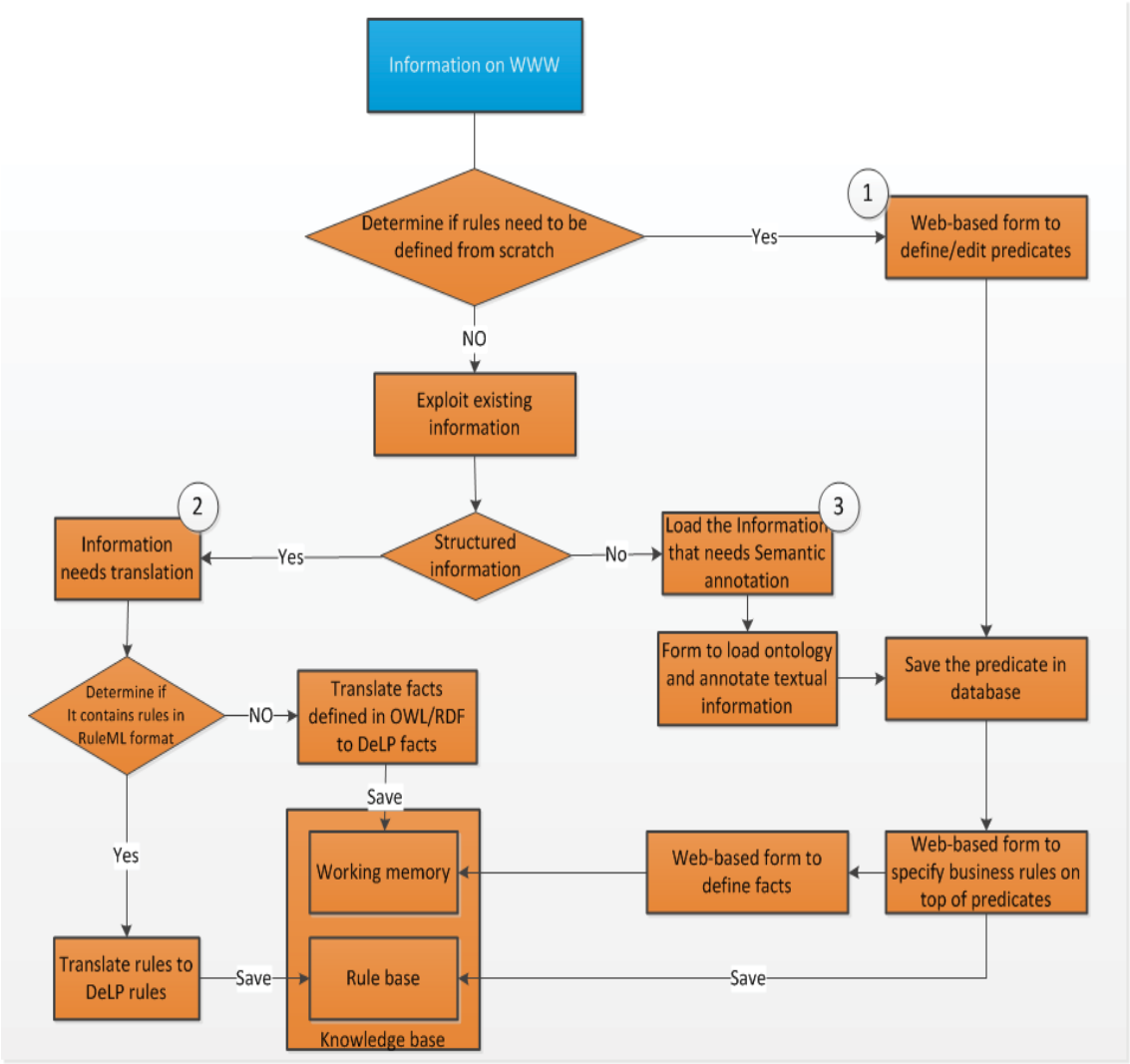


Figure 4.3: Flowchart illustrating steps involved in information representation

3. Use semantic annotation to annotate the unstructured information with the domain ontology, and use the annotated information for the specification of rules and save them in the knowledge base. In Chapter 7, the solution for the semantic annotation of unstructured information (business policies documents) with the domain ontology and then the specification of business rules on top of the annotated business policies concepts is discussed.

To exploit the information on the WWW, the proposed solution for incomplete and/or contradictory information is sub-divided into the following three sub-solutions, each of which corresponds to the objective identified in Section 3.5.1. The sub-solutions are as follows:

1. Selection of a rule-based language for incomplete and/or inconsistent information representation

   After careful analysis of the existing work in Sections 2.5.2.2 and 2.8.2.2, Defeasible Logic Programming (DeLP) which has been used to represent incomplete and/or inconsistent information in software agents was selected. The reasons for this selection are as follows:

   - DeLP is capable of representing incomplete and/or contradictory information.
   - DeLP allows specification of information for reasoning where conflicts may arise at run time which often happens in group decision making.

   In Chapters 5, 6 and 7, different case studies for argumentation support in different contexts are discussed as well as how the DeLP language assists in representing incomplete and/or contradictory information.

2. Solution for a translation mechanism to translate rules defined in RuleML to the system's rule language

   To enable the translation of rules defined in RuleML format, a RuleML translator is proposed that takes a RuleML file as input, parses it and extracts rules from it. It then transforms the extracted rules in DeLP format and saves them in the rule base. In Chapter 5, the working of the proposed RuleML translator is outlined in detail.

3. Solution for the translation of information in OWL/RDF to the system's rule language

   To ensure the facts defined in the OWL/RDF format are available for the reasoning process, the OWL/RDF translator is proposed which translates these facts to DeLP facts and saves them in the working memory. In Chapter 5, the working of the OWL/RDF translator is discussed in detail.

4.4.2 Solution for monological argumentation-driven reasoning engine to reason over incomplete and/or contradictory information

Once the required information for decision making has been captured, the next step is to perform reasoning on it. This will be done by the argumentative reasoning module of the logic-based framework. In this section, an overview is given of the proposed

solution that allows the Semantic Web applications of an enterprise to reason over the underlying information specified using the DeLP language. To achieve this objective, as shown in Figure 4.4 , the following two steps are required:

1. construction of arguments from the knowledge base with the help of data-driven reasoning;

2. once the arguments are constructed, then the arguments are considered for conflict identification and their resolution (if they exist). This objective is achieved by goal-driven reasoning. During goal-driven reasoning, different argumentation-driven conflict resolution strategies are provided to resolve the conflicts between arguments and their counter-arguments. This results in the construction of an 'arguments set' where a priority between arguments and their counter-arguments.

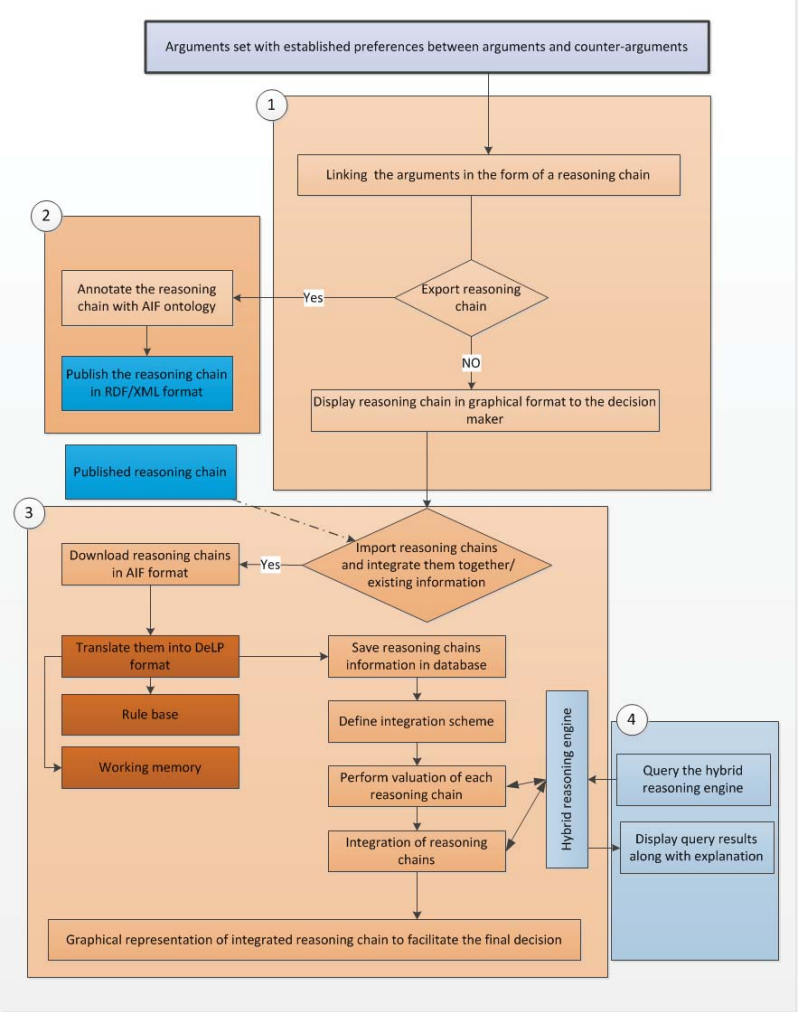


Figure 4.4: Flowchart illustrating steps performed by argumentative reasoning module

To design the complete solution for reasoning over underlying information, the solution for the hybrid reasoning engine is sub-divided into the following two sub-solutions, each of which corresponds to the objectives identified in Section 3.5.2.

1. Solution for developing a hybrid reasoning engine that performs data-driven reasoning for arguments construction and goal-driven reasoning for conflicts identification and their resolution

   As pointed out in Section 2.5.2.2 , DeLP only uses goal-driven reasoning with the objective to serve the decision maker's queries only. It does not provide a solution for data-driven reasoning to infer new knowledge from existing information. The proposed solution overcomes this drawback and provides the functionality of hybrid reasoning over underlying information. The steps involved in this process are as follows:

   - The rules present in the rule base are compiled in the form of a Rete network which makes the information ready for data-driven reasoning. In Chapter 5, the extension of the Rete network to compile rules that may represent incomplete and/or contradictory information is discussed.
   - Data-driven reasoning starts by the introduction of facts in the Rete network. This results in the activation and firing of the rules. The derived facts flow back into the Rete network which, in turn, results in the activation of new rules. This process continues until no more rule/s are activated. During this processes, the activated rules are saved in the arguments set. In Chapter 5, the syntax and semantics for data-driven reasoning over underlying information specified in DeLP format is given. In Chapter 6, the extension of data-driven reasoning by providing the syntax and semantics to support information and knowledge integration in Semantic Web applications is given.
   - Once data-driven reasoning is completed, goal-driven reasoning starts in order to identify and resolve conflicts if they exist between arguments. Goal-driven reasoning provides different conflict resolution strategies to resolve the conflicts between arguments and their counter-arguments and make the information ready for integration. In Chapter 5, goal-driven reasoning using DeLP and the extension which is made to it to enable it to work with data-driven reasoning in Semantic Web applications is described. In Chapter 6, the extension of goal-driven reasoning by providing the syntax and semantics to support information and knowledge integration in Semantic Web applications is discussed.

2. Solution for different argumentation-driven conflict resolution strategies to resolve conflicts between arguments and their counter-arguments

   To assist the decision makers in different decision-making scenarios, the need for different conflict resolutions was identified in Section 3.5.2. This objective is achieved in the proposed framework which provides the following conflict resolution strategies:

   - Generalize specificity-based conflict resolution strategy

     This is a DeLP built-in strategy for conflict resolution which takes into account the 'information specificity' criteria to resolve conflicts between arguments and their counter-arguments . In Chapter 5, this is discussed in detail and an explanation as to how it has been enhanced to resolve conflicts in the context of Web-based Intelligent Decision Support Systems is given.

   - Dung's style-based conflict resolution strategy

     This is also an automated conflict resolution strategy where an argument X (part of a reasoning chain Y) is attacked by a counter-argument Z (not part of reasoning chain Y) and X gets defeated by Z if there is no other argument (in reasoning chain Y) that attack and defeat the counter-argument Z. In Chapter 7, the working of this strategy is described in detail.

   - Fuzzy preferences-based conflict resolution strategy

     The decision makers can provide their preferences to resolve conflicts between arguments and their counter-arguments. These preferences are fuzzy in nature. In Chapter 7, the working of this strategy is described in detail and an explanation of its computational model to resolve conflicts and establish the priority between arguments and their counter-arguments is given.

   - Voting-based conflict resolution strategy

     The decision makers can vote either in favour or against a certain argument which is in conflict with another argument (counter-argument) to resolve the conflict. In Chapter 7, the working of this strategy is described in detail with a discussion of the methodology used to capture the user's and expert's votes and compute them in order to establish the priority between arguments and counter-arguments.

4.4.3 Solution for information and knowledge integration

Once the reasoning over incomplete and/or contradictory information by the hybrid reasoning engine is completed, the next step is to integrate the output of the reasoning engine and display the results to the decision makers in order to assist them in their decision-making process. As shown in Figure 4.5, the following steps are required to transform the information produced by a reasoning engine into a form of a reasoning chains that in turn provides the basis for knowledge integration:

- The output of the hybrid reasoning engine is integrated in the form of a reasoning chain by linking the facts to the conclusions drawn. It also displays the reasoning results in a graphical format for better comprehension of results by decision makers.
- To share the reasoning chain with other Semantic Web application, the reasoning chains is annotated with an AIF compliant 'reasoning chain ontology' and is published over the WWW in RDF\XML format.
- Once the reasoning chains are available in AIF format, they are downloaded and translated to DeLP format in order to integrate them together or with existing reasoning chains generated by Semantic Web applications. This is called knowledge integration.
- The reasoning engine also provides a querying facility where decision makers can query the knowledge base and can obtain an explanation of the results of the query and how this result was achieved.

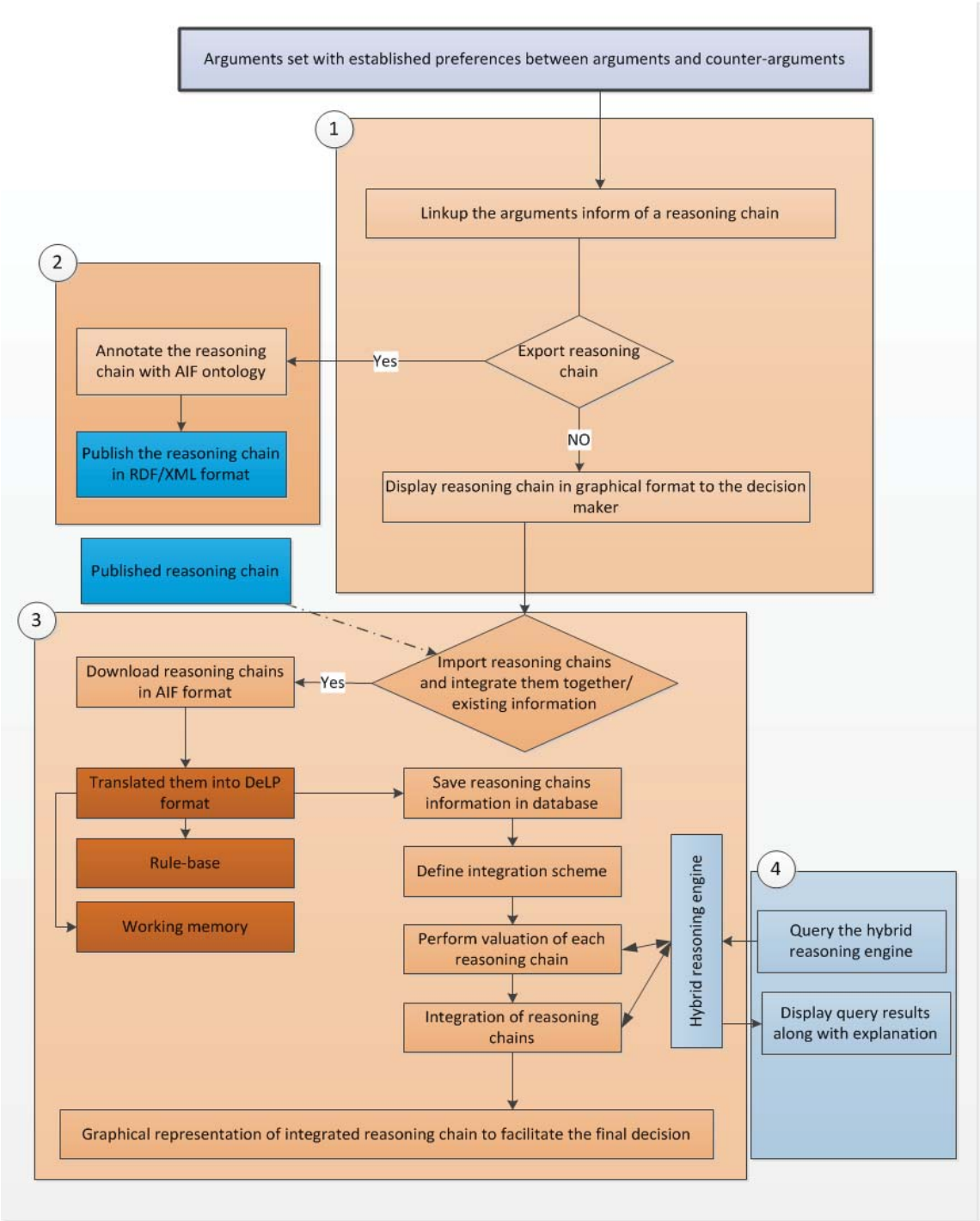


Figure 4.5: Flowchart illustrating steps performed for information and knowledge integration

To design a complete solution for the integration of incomplete and/or contradictory information to facilitate the decision-making process, the solution for information and knowledge integration comprises the following sub-solutions, each of which corresponds

to the objectives identified in Section 3.5.3:

1. Solution for information integration and its graphical representation

   The last step required for an argumentation process is the construction of a conclusion by integrating information (arguments) in the form of a reasoning chain. To achieve this objective, the arguments are linked together in the form of a reasoning chain. In Chapters 5 and 6, the building of reasoning chains in different enterprise contexts is discussed. In Chapter 7, the extension of the graphical representation of a reasoning chain to represent a business process model is described. In Chapter 8, different Semantic Web applications that provide a graphical representation of the reasoning process are discussed.

2. Solution for importing/exporting integrated information to different Semantic Web applications

   To share the results of the hybrid reasoning engine among different semantic Web applications, a functionality to export results in a standard format is needed. To achieve this objective, in Chapter 6, the methodology for the semantic annotation of a reasoning chain with ArgDF ontology[1] is described and an explanation as to how the AIF compliant reasoning chain is serialized in RDF/XML format is given.

3. Solution for knowledge integration

   To obtain a complete picture about a particular subject in an enterprise, the output of different reasoning engines may need to be integrated to facilitate the decision making process in enterprises. To achieve this objective, the proposed solution involves the following steps:

   - Valuation of a reasoning chain. This involves the evaluation of a reasoning chain against the decision maker's defined integration scheme. The results of this evaluation assist the decision maker to decide whether or not to include the underlying reasoning chain for knowledge integration. In Chapter 6, the process of defining the integration scheme and its use to evaluate the reasoning chain is outlined.

   - Argumentative reasoning is performed over a set of reasoning chains that have been selected for knowledge integration. During this process, conflicts are resolved among arguments, keeping in mind the decision maker's preferences. This process also results in the construction of new arguments

[1] http://www.argdf.org/source/ArgDF Protege Ontology.zip

by merging existing arguments that support the same claim. In Chapter 6, the methodology of argumentative reasoning over a set of reasoning chains is outlined.

- Once the argumentative reasoning is completed, the underlying information i.e. reasoning chains, is integrated into a single reasoning chain called the integrated recommendations space. The integrated recommendations space is then displayed to the decision maker in a graphical format to assist him in the decision-making process. In Chapter 6, this process is explained in detail.

4. Solution for querying the knowledge base and obtaining an explanation of results

   To answer the questions of a decision maker which may help him to understand the reasoning process (that is, to obtain an explanation on the conclusion achieved or conflicts resolved), there is need for a querying mechanism to query the knowledge base. To achieve this objective, the proposed solution provides a querying functionality thorough which decision makers can obtain answers to their questions. In Chapters 6 and 7, an explanation on how to query a knowledge base and how the results are displayed is given. In Chapter 8, the different Semantic Web applications that provide a Web-based interface to query the knowledge base and the graphical results are discussed.

4.5 Applications layer

The application layer refers to a set of Semantic Web applications that need to represent, reason and integrate information that could be incomplete and/or inconsistent and exists within the enterprise and/or in other enterprises. These applications exploit the logic-based framework located at @IRRI layer to achieve their objectives.

4.6 Realization of Semantic Web applications using GF@SWA for Business Intelligence

To demonstrate the working capabilities of the GF@SWA i.e. consider incomplete and/or contradictory information coming from various sources and assist the decision maker in the decision making process, three Semantic Web applications are considered in this thesis. They are as follows:

1. **A**rgumentation-enabled **Web**-based **IDSS** for reasoning over incomplete and/or contradictory information (Web@IDSS)

2. **K**nowledge Integration through **A**rgumentative Reasoning by **Web**-based **IDSS** (Web@KIDSS)

3. **K**nowledge **R**epresentation approach with **A**rgumentative reasoning for **P**rocess **M**ap **D**iscovery from Business policies (KR@PMD)

In the following sections, each of these applications is discussed in detail.

4.6.1 Web@IDSS

Section 1.2 identified that the major shortcoming of existing Web-based IDSS is their inability to represent and handle incomplete and/or contradictory structured information spanning across enterprise boundaries. This is particularly important for enterprises which take into consideration the information available on the Web for timely and intelligent decision-making support. So a system is needed that is able to capture the information outside an enterprise's boundaries, identify the goals, conflicts in the information with respect to the goals, resolve these conflicts by reasoning over them and show the basis of the reasoning to the decision maker by which a conclusion is reached.

To overcome the drawbacks of existing systems, Web@IDSS is proposed that exploits the functionality of the logic-based framework present at @IRRI to represent, reason and integrate incomplete and/or contradictory information.

In Chapter 5, a case study is presented and a conceptual framework for Web@IDSS is proposed to represent, reason and integrate information across enterprise boundaries. The main functionalities of the @IRRI layer exploited by Web@IDSS are as follows:

1. Exploit the functionality of the *information representation module* of the logic-based framework to represent the information in DeLP format.

2. Exploit the functionality of the *argumentative reasoning module* to perform data-driven reasoning over underlying information for arguments construction and goal-driven reasoning for conflicts identification and their resolution.

3. Exploit the functionality of the *information and knowledge integration module* to provide information integration, an explanation of the reasoning results and to export the reasoning results in AIF format to enable them to be shareable over the WWW.

In Chapter 8, the functional validity of Web@IDSS is discussed with the help of use cases which are tested on the developed application.

4.6.2 Web@KIDSS

Section 1.2 showed that due to the monotonic nature of the layered development of the Semantic Web, Web-based IDSS lacks the capability to represent, reason and integrate incomplete and contradictory information. This, in turn, renders an enterprise incapable of *knowledge integration*; that is, the integration of information about a subject that could be incomplete, contradictory and distributed among different Web-based IDSS within or across enterprises. So a system is needed that can consider the reasoning chains produced by different hybrid reasoning engines located within and/or beyond an enterprise boundaries and provides solution for knowledge integration.

To overcome the drawbacks of existing systems, a Web@KIDSS is proposed that exploits the functionality of the logic-based framework present at @IRRI to represent, reason and integrate incomplete and/or contradictory information.

In Chapter 6, a case study is outlined and a conceptual framework for Web@KIDSS is proposed to represent, reason and integrate information across enterprise boundaries. The main functionalities of the @IRRI layer exploited by Web@KIDSS are as follows:

1. Exploit the functionality of the *information representation module* of the logic-based framework to represent the information in DeLP format.
2. Exploit the functionality of the *argumentative reasoning module* to perform data-driven reasoning over underlying information for argument construction and goal-driven reasoning for conflict identification and resolution.
3. Exploit the functionality of the *information and knowledge integration module* to

   (a) import and transform the published AIF reasoning chains in DeLP format

   (b) integrate knowledge which involves the evaluation of the reasoning chain, and conducting argumentative reasoning over a set of reasoning chains followed by their integration.

In Chapter 8, the functional validity of Web@KIDSS is discussed with the help of use cases which are tested on the developed application.

4.6.3 KR@PMD

Section 1.2 identified that in an enterprise unstructured information accounts for around 80% of the total information and it ranges from customer reviews, customer buying preferences for new product, business policies of an enterprise or collaborating enterprises etc, which when considered by applications can provide better insights in the decision making process according to their needs. However, it is also possible that such information may be in different formats and potentially incomplete and/or contradictory within themselves or with the existing information in an enterprise. So a system is needed that is able to capture the unstructured information, reason and resolve conflicts followed by integration and grpahical representation of integrated information to the decision maker to assist him in decision making process.

To overcome the drawbacks of existing system, KR@PMD is proposed that exploits the functionality of logic-based framework present at @IRRI to represent, reason and integrate incomplete and/or contradictory information and show it in graphical format to the decision maker.

In Chapter 7, a case study is outlined and a conceptual framework for KR@PMD is proposed that analyses the unstructured information i.e. business policies, of an enterprise) for discovering a business process map. The main functionalities of the @IRRI layer exploited by KR@PMD are as follows:

1. Exploit the functionality of the *information representation module* of the logic-based framework to represent the information in DeLP format.

2. Exploit the functionality of the *argumentative reasoning module* to perform data-driven reasoning over underlying information for argument construction and goal-driven reasoning for conflict identification and resolution.

3. Exploit the functionality of the *information and knowledge integration module* for information integration and its graphical representation as a business process map.

In Chapter 8, the functional validity of KR@PMD is discussed with the help of use cases which are tested on developed application.

4.7 Conclusion

In this chapter, an overview is given of a solution to address the research objectives in this thesis i.e. to support monological argumentation in Semantic Web applications.

The overall architecture was presented and the information layer, the @IRRI layer and the applications layer was discussed in detail.

In the next chapter, the exploitation of the logic-based framework (located at the @IRRI layer) by Web-based Intelligent Decision Support Systems (located at the applications layer) is discussed in order to represent, reason and integrate incomplete and/or contradictory information exists within an enterprise and/or in other enterprises.

Chapter 5 - Argumentation-enabled Web-based Intelligent Decision Support System (Web@IDSS)

5.1 Introduction

In any enterprise, information is one of the essential components required for decision making. Traditional information systems have been used by enterprises to consider the underlying information of an enterprise and assist them in this process. However, these systems are basic and are inflexible in responding to current situations such as:

- Dealing with the huge increase of information. In recent years, there has been a huge increase in the amount of information available, termed the tsunami of data (Brodie, 2008b,a). In order to make informed decisions, enterprises may have to consider a huge volume of information as that may contain hidden informed knowledge. So, information systems need to process this information autonomously and make it available to decision makers to assist them in the decision-making process.
- Dealing with information that may be across and beyond an enterprise's boundaries. For example, in the context of Customer Relation Management (CRM) software, for the development of new products, considering information such as expert knowledge, customer opinions, reviews about existing products and services etc. in the decision-making process may lead to better results.

In order to overcome these issues, Decision Support Systems (DSS) (such as individual DSS and Group DSS) were developed that assist in a wide range of enterprise-wide decision-making processes (Power, 2002; Power and Sharda, 2009). To consider the multi-site nature of decision making due to the widespread adoption of the

WWW, Web-based DSS (Yao et al., 2001) were developed by which decision makers that are spread across different locations can collaborate in the decision-making process (Vahidov and Kersten, 2004; Silverman et al., 2001; Toni, 2007). By using Semantic Web technologies, Web-based DSS, with help of ontologies, can understand and consume information which exists outside an enterprise's boundaries. The challenge that now confronts the current Web-based DSS systems is: *how to take into account the information exists within an enterprise and/or in other enterprises that may be potentially incomplete and/or contradictory (within themselves and/or with the existing information in an enterprise) and utilize it in their decision-making process*.

To address this challenge, as mentioned in Section 3.3, Web-based DSS have been developed that are based on defeasible reasoning to represent incomplete information and reason using pre-defined preferences from a single user's point of view to resolve conflicts (Antoniou and Bikakis, 2007; Bassiliades et al., 2004; Grosof et al., 2002). However, these systems fail to address the problem in the context when information may come from different sources such as in group decision making where there is more than one decision maker involved in the decision-making process where conflicts may arise between the members of the group due to their different viewpoints. To address this challenge, I propose a framework for an Argumentation-enabled Web-based Intelligent DSS (Web@IDSS). The proposed framework will use logic-based language for information representation and argumentation-driven reasoning to identify and resolve conflicts in the information coming from different sources, followed by information integration to assist a decision maker in his decision-making process. This will advance the research in Web-based DSS as depicted in Figure 5.1.

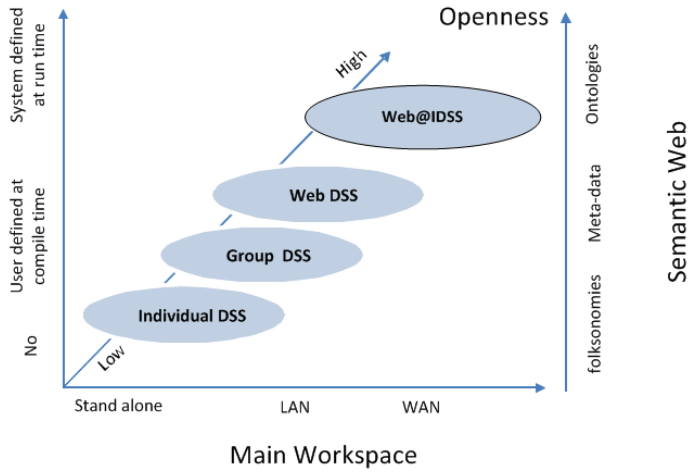


Figure 5.1: Evolution towards Argumentation-enabled Web-based IDSS (extended from (Lee and Chung, 2005))

The organization of this chapter is as follows: in Section 5.2, the problem to be addressed is outlined by using a case study that highlights the requirements and challenges for Web-based DSS in an enterprise. In Section 5.3, an overview of the proposed framework for Argumentation-enabled Web-based Intelligent DSS (Web@IDSS) is given. From Sections 5.4 to 5.6, each component of the proposed framework is explained in detail and the ways in which it provides a solution to the problem highlighted in the case study is discussed. Section 5.7 concludes the chapter.

5.2 Case study for problem definition

To explain the problem with an example, consider a scenario where Mr David is a marketing manager of an enterprise A. He is responsible for formulating and suggesting business strategies to increase sales of the company's products (existing and new) and generate revenue for the Chief Executive Officer (CEO) of the enterprise A. Enterprise A intends to manufacture a new product (say Product B). To increase the enterprise's revenue from this project, one of the important aspects that Mr. David identifies is "the greater the discount that an enterprise A receives from the supplier, the cheaper the new product", and negotiation plays an important part in securing the maximum discount. Mr. David identifies that the materials for manufacturing the product will be sourced from 'N' different suppliers, each of whom offer varying levels of discount. Mr David would like to select a supplier that may give his enterprise the maximum discount and he needs to justify his selection to the CEO of the company.

To achieve his objective, Mr David needs to analyse the business policies of each supplier against his company's requirements along with the feedback provided by the other users (companies) about the raw materials provided by the suppliers as shown in Figure 5.2 . During this process, Mr. David will come across different challenging situations such as follows:

- There might be conflicts between the supplier's policies and an enterprise A's business requirements.
- There may be conflicts within the supplier's business policies.
- A situation may arise where Mr. David may require some information for decision making which is not available at the time of decision making.

In order to overcome the above mentioned challenging situations, Mr. David requires a Web-based DSS that will assist him to overcome these challenges. The Web-based DSS should have the following functionalities:

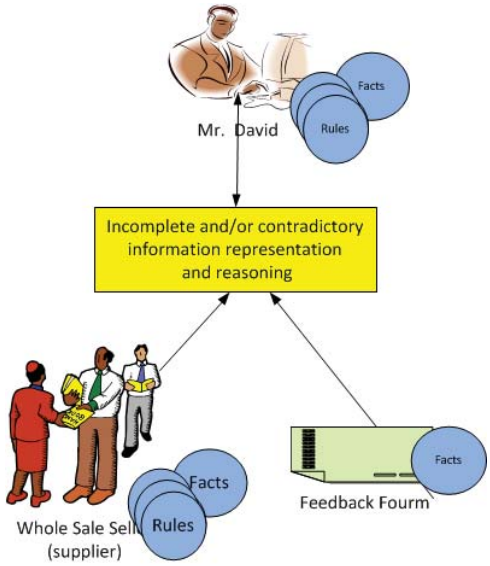


Figure 5.2: Analyses of the business policies of a supplier and feedback provided by the other users (companies) by Mr. David

1. an interface to define the requirements in the form of business rules such as 'Purchase product from supplier only if product feedback is good' and certain facts or information to realize those rules;

2. an interface to download the supplier's product information and public policies with details on the possible discount that can be given on their products and services;

3. the capability to download feedback or reviews from other users (companies) on the suppliers' products from a third party forum such as Amazon;

4. situations may arise where the business policies of a supplier may be incomplete or negotiation is required between the supplier and an enterprise A to resolve conflicting interests. The Web-based DSS should be able to cater for these and provide a means of resolving these conflicts, with a justified explanation, during the reasoning process;

5. the capability to provide a graphical representation of the reasoning process and the result in order to make them easily understandable by non-technical persons such as CEOs.

To have such functionalities, a Web-based IDSS is needed that is able to capture the information outside an enterprise's boundaries, identify the goals, identify any conflicts in the information with respect to the goals, resolve these conflicts by reasoning over

them and show the basis of the reasoning by which a conclusion is reached. The current Web-based DSS are not able to represent, reason and integrate the information that is required for the abovementioned tasks. Therefore, to address this challenge, Mr. David's requirements, which should be incorporated in Web-based IDSS, are formalized as follows:

- A declarative, logic-based language for specification of the business requirements of an enterprise.
- The declarative language should have the capability to represent incomplete and contradictory information (i.e. business rules and facts).
- An inference mechanism that can perform reasoning pertaining to incomplete and/or contradictory information in the knowledge base.
- Graphical representation of results obtained from the reasoning process to assist in decision making.
- Justifiable explanation of the results obtained after the reasoning and conflict resolution has occurred.

Assumption

- Enterprise A, the supplier and the feedback forum share a common vocabulary defined in RDF/XML format and the predicates defined in the vocabulary are used for the specification of business rules and policies. Therefore, the information taken into account by the Web-based IDSS is structured information.

To achieve the abovementioned objectives, in the next section, a Web@IDSS framework is proposed that can represent, reason and integrate incomplete and/or contradictory information which exists within an enterprise and/or in other enterprises to assist the decision maker in the decision-making process.

5.3 Proposed framework for Argumentation-enabled Web-based IDSS (Web@IDSS)

In this section, the solution for an Argumentation-enabled Web-based IDSS is proposed to represent, reason and integrate incomplete and/or contradictory information exits within an enterprise and/or in other enterprises. Figure 5.3 represents the proposed framework and consists of three layers as follows:

1. Information layer

   The information layer represents the structured information identified by the decision maker to be considered during the decision-making process. This information may include:

   - Business policies of an enterprise that provides different products and services published on the WWW.
   - Feedback of users published on the WWW about the products and services offered by an enterprise.

2. @IRRI layer

   This layer comprises a logic-based framework that enables a Web-based DSS to deal with information which is potentially incomplete and/or contradictory, and to process and consider it for decision making. It provides different modules to represent or translate the information into DeLP format, perform hybrid reasoning for arguments construction from underlying information followed by conflicts resolution and then integrate the information obtained from the hybrid reasoning to assist the decision maker in the decision-making process. The modules are as follows:

   (a) *The Information representation module* is responsible for

   - the pre-processing of potentially incomplete and/or contradictory information, and
   - the translation of pre-processed information to DeLP format and saving it in the knowledge base.

   (b) *The Argumentative reasoning module* performs hybrid reasoning over information saved in the knowledge base. The hybrid reasoning engine performs two types of reasoning such as:

   - data-driven reasoning for arguments construction, and
   - goal-driven reasoning for conflicts identification followed by their resolution.

   (c) *The Information and knowledge integration module* is responsible for

- performing integration of the output of the hybrid reasoning in the form of a reasoning chain;
- categorization of the reasoning chains based on the types of arguments they are built on;
- graphical representation of the reasoning chain.

3. Web-based decision support systems (Web-based DSS) layer

This layer consists of Web-based DSS such as Web@IDSS, that exploits the @IRRI layer and the information layer to achieve its objectives.

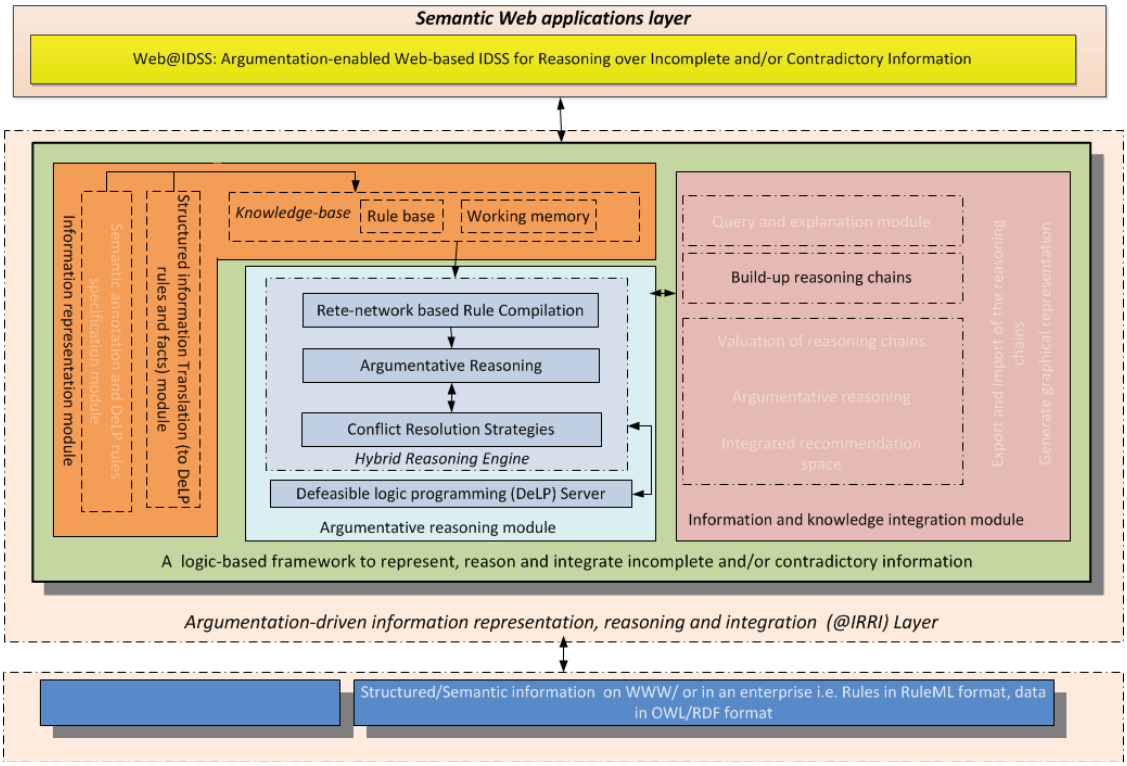


Figure 5.3: Proposed conceptual framework with highlighted components exploited by Web@IDSS

Before explaining the working of the proposed framework, in the next sub-section, several important definitions and concepts are introduced that are pivotal to understand the working of the proposed framework for Web@IDSS.

5.3.1 Important definitions

In this section, the important concepts that encompass *syntax and semantics* for DeLP to make it suitable for information representation, reasoning and integration

in Semantic Web applications are defined as follows:

5.3.1.1 DeLP Language

DeLP language is a set Φ containing a set of predicates $\mathcal{P}$, a set of functions $\mathcal{F}$, an infinite set of variables $\mathcal{X}$, a finite set of symbols $\mathcal{S}$, and a set of labels $\mathcal{L}$. Mathematically, language is defined as follows:

$$\Phi = \{\mathcal{P}, \mathcal{F}, \mathcal{X}, \mathcal{L}, \mathcal{S}\} \quad \text{Equation (5.1)}$$

The language supports two types of negation: *strong negation*, represented by the symbol $\sim \in \mathcal{S}$ to represent contradictory knowledge, and *weak negation* which represents negation as failure represented by the symbol $not \in \mathcal{S}$ which is used to represent incomplete information.

5.3.1.2 Working memory

A collection of facts is known as working memory. Considering a set $\mathcal{P}$ of predicates and an infinite set of variables $\mathcal{X}$, a fact is a ground predicate $f \in \mathcal{P}$,or a negated ground predicate $\sim f \in \mathcal{P}$. A set of facts, i.e working memory is represented by $\mathcal{WM}$. Mathematically, working memory is defined as follows:

$$\mathcal{WM}=\{f \cup \sim f \mid f,\sim f \; are\, ground\, predicates\} \quad \text{Equation (5.2)}$$

where a ground predicate is a predicate whose input arguments are constant. The predicate p(a, b) and *not* p(a, b) are ground predicates. Facts represent the current state of the world and these provide some sort of evidence as a basis for activating the rules of inference to infer new facts. If there are no facts in the system, then no inference rules will be activated.

5.3.1.3 Production rule

Production rules are rules of the form *IF Condition DO Action*, where Condition queries the working memory containing the facts on which the rules operate. Formally, a production rule $\mathcal{A}$ is of the form: [rule identifier] [rule body] [type of inference rule] [conclusion]. Mathematically, a production rule is defined as follows:

$$[\mathcal{A}]\, \nabla \vdash \alpha \quad \text{Equation (5.3)}$$

where

- [rule identifier]: $\mathcal{A} \in \mathcal{L}$ is used as the identifier or name of the production rule;
- [rule body] ∇ is a pattern in the body of a production rule $\mathcal{A}$. A pattern is a tuple of predicates i.e. $\nabla \subseteq \mathcal{P}$**,** and defined as $\nabla = (C_i, ..., C_j)$ where $0 < i < j$, C_i is a predicate in a pattern;
- [conclusion] α is a predicate whose instances could be intuitively considered to be added to the working memory when the rule is fired during argument construction defined later on; and
- [type of inference] $\vdash$ indicates the inference that associates the rule body with the conclusion.

The production rule represents a reasoning step for α from a tuple of predicates $\{C_1, ..., C_n\}$. The language supports two types of inferences in production rules. One is strict inference represented by the symbol $\rightarrow \in \mathcal{S}$ and the second is defeasible inference represented by the symbol $\dashrightarrow \in \mathcal{S}$. Strict inference is used to represent information about which there is no ambiguity, whereas defeasible inference is used to represent ambiguous or tentative information. Strong negation is allowed at the conclusion of the rule, whereas weak negation is allowed only in the body of the rule.

5.3.1.4 Rule base

The set of production rules is known as the rule base, denoted by $\mathcal{R}$. Mathematically, the rule base is defined as follows:

$\mathcal{R}$= {*production rule*} ..... Equation (5.4)

5.3.1.5 Strict production rule

Strict production rules are the rules in the classical sense: when a rule's conditions are true, apply the rules and reach a conclusion. These rules are used to represent an inference mechanism from conditions to conclusion without any doubt. Most of the time, these rules are constructed from terms such as 'should be', 'must be', 'must' and their opposite terms. Formally, a production rule $\mathcal{S} \in \mathcal{R}$ is a strict production rule of the following form if the rule is based on strict inference.

$$[\mathcal{S}]\, \nabla \rightarrow \alpha \quad \text{..... Equation (5.5)}$$

The strict production rule $\mathcal{S} \in \mathcal{R}$ is used to represent truthful information which contains no ambiguity. Consider rule r1 which states that 'if a person is innocent and has no crime history then he is not guilty' and rule r2 which states that 'if someone is not guilty, then he is free'. These rules can be represented as strict production rules thus:

- $[r1] innocent(X), hasCrimeHistory(X, no) \rightarrow (X)$
- $[r2] notguilty(X) \rightarrow free(X)$.

5.3.1.6 Defeasible production rules

Defeasible rules or refutable rules are those that link the set of conditions to a conclusion with a certain doubt, and therefore could be refuted by contrary evidence. This type of rule is indicated by words like 'usually', 'presumably', or 'sufficiently' or we could intuitively feel that it is refutable. Formally, a production rule $\mathcal{D} \in \mathcal{R}$ is a defeasible production rule of the following form:

$$[\mathcal{D}]\, \nabla \dashrightarrow \alpha \quad \text{..... Equation (5.6)}$$

A defeasible production rule $\mathcal{D} \in \mathcal{R}$ is used to represent tentative information which may change in due course. Consider rule r3 that states: 'assume that someone is innocent whenever it has not been proven that he is guilty' and rule r4 that states: 'generally, do not cross the railway tracks if it ca not be proven that no train is coming'. These rules can be represented as defeasible production rules as follows:

- $[r3]$ $not\ guilty(X)$--→ $innocent(X)$.
- $[r4]$ $not\ {\sim}train\_is\_coming$ --→$\sim$ $cross\_railway\_tracks(X)$.

5.3.1.7 Argumentative production system

An argumentative production system is defined as a system that allows representation and execution (i.e. reasoning) of both strict and defeasible production rules. It consists of a knowledge base (i.e. consisting of working memory and a rule base) and a hybrid reasoning engine. An argumentative production system is formally defined as follows:

$\mathbb{P}$=($\mathcal{WM}$,$\mathcal{R}$,Args) ..... Equation (5.7)

- where $\mathbb{P} \in \mathcal{L}$ is a label to identify the argumentative production system.
- $\mathcal{WM}$ represents the initial collection of facts in the argumentative production system.
- $\mathcal{R}$ is the set of rules comprising both strict and defeasible production rules in the argumentative production system.
- $Args$ is an active argument set which contains arguments generated during the argument construction phase, which will be defined later. Prior to the argument construction phase, the $Args$ is an empty set.

5.3.1.8 Consistency

A set of rules is consistent if and only if there are no two rules with mutually contradictory predicates as their conclusion. Mathematically, this is represented as follows:

$R_{consis}= \{\forall r, s \in \mathcal{R} | \, if \, r \vdash \alpha \, then \; s \nvdash \sim \alpha\}$ ..... Equation (5.8)

5.3.1.9 Arguments construction

Arguments construction is defined as a recursive process which involves the interpretation of production rules with function match($\mathcal{WM}$, $\mathcal{R}$) $\in \mathcal{F}$ which looks for rules from a rule base whose pattern matches the facts in $\mathcal{WM}$ and, on a successful match, executes the production rule which then adds the rule's conclusion i.e. ground predicate, to the working memory and instance of the production rule i.e. argument, to the argument set i.e. $Args$. Such a reasoning process is also known as data-driven reasoning. The argument construction process continues until all the matched rules in the knowledge base have been processed. This interpretation of a production rule is also known as the 'firing of a rule'.

$\forall r \in \mathcal{R} \; \{\nabla \in r, \alpha \in r, r \notin Args \mid if match(\nabla, \mathcal{WM}) \; then \; \mathcal{WM}\prime = \mathcal{WM} \cup \alpha' \; and \; Args = Args \cup r'\}$ ..... Equation (5.9)

where α' is the ground predicate and r' is the interpreted rule by function $match(\mathcal{WM}, \mathcal{R}) \in \mathcal{F}$. The $Args$ contains interpreted rules or fired rules known as arguments.

5.3.1.10 Strict argument

A fired production rule in an argument set with strict inference is called a 'strict argument'. Mathematically, this is represented as follows:

$[\mathcal{S}] \; \beta_1,.....,\beta_n \rightarrow \alpha$ ..... Equation (5.10)

where

1. $\mathcal{S} \in \mathcal{L}$ is the label of the argument

2. α is a ground predicate known as the 'claim of an argument'. Function $claim(\mathcal{S})$ $\in \mathcal{F}$ returns the claim of a given argument $\mathcal{S}$.

3. β_i is a ground predicate known as the premise of an argument, supporting the claim of an argument. Function $premises(\mathcal{S})$ returns a set of argument premises $\mathcal{S}$.

4. $\rightarrow$ represents a strict inference from the set of premises to the claim.

5.3.1.11 Defeasible argument

A fired production rule in an argument set with defeasible inference is called a 'defeasible argument'. Mathematically, this is represented as follows:

$$[\mathcal{D}]\beta_1, ..., \beta_n \dashrightarrow \alpha \quad \text{..... Equation (5.11)}$$

where

1. $\mathcal{D} \in \mathcal{L}$ is the label of an argument.

2. α is a ground predicate known as the 'claim of an argument'. Function $claim(\mathcal{D})$ $\in \mathcal{F}$ returns the claim of a given argument $\mathcal{D}$

3. β_i is a ground predicate known as the premise of an argument, supporting the claim of an argument. Function $premises(\mathcal{D})$ returns a set of argument premises $\mathcal{D}$

4. $\dashrightarrow$ represents defeasible inference from the set of premises to the claim.

To avoid any fallacies in the argumentation process, the following restrictions on strict and defeasible argument structure are considered:

1. A premise in an argument cannot simultaneously be a conclusion i.e. $\beta_i \notin \alpha$.

2. A negation of a claim cannot become the premise of a claim i.e. $\beta_i \neq \sim \alpha$.

3. There is no redundancy of a premise in a pattern. $\beta_i \neq \beta_j$ where $1 < i, j <$ n.

5.3.1.12 Counter-argument

An argument r counter-argues argument s if and only if $claim(r)$ is inconsistent with $claim(s)$ or $claim(r)$ is inconsistent with the $premises(s)$. Mathematically, a counter-argument is defined as :

$$\forall r, s \ \{ \ if(!Consistent(claim(s), claim(r)) \ then \ r \ \Diamond s\} \ \ldots\ldots\ldots\ldots \text{ Equation (5.12)}$$

where $\Diamond$ is used to represent the counter-argument relationship between two arguments.

If argument r counter-argues argument s such that $claim(r)$ is inconsistent with $claim(s)$, it is called a 'direct counter-argument', and if argument r counter-argues s such that $claim(r)$ is inconsistent with $premises(s)$, then it is called an 'indirect counter-argument'. Mathematically, direct and indirect counter-arguments are represented as follows:

$$\forall s,r\{ \ if \ !Consistent(claim(s), claim(r)) \ then \ s \Diamond_{direct} r\} \ \ldots\ldots\ldots\ldots \text{ Equation (5.13)}$$

$$\forall s,r\{if \ !Consistent(claim(s), premises(r)) \ then \ s \Diamond_{indirect} r\} \ \ldots\ldots\ldots \text{ Equation (5.14)}$$

A strict rule cannot counter-argue another strict rule because of the definition of consistency.

5.3.1.13 Static defeat

Under certain conditions, an argument r defeats its counter-argument s by establishing its priority over its counter-argument. Such defeat is known as a 'static defeat'. The conditions for static defeat are as follows:

- If a strict argument counter-argues a defeasible argument, the strict argument always defeats a defeasible argument. In other words, the strict argument has

higher priority than the defeasible argument. Mathematically, this is represented as follows:

$\forall d, s \in Args\{if\ s, d\ are\ strict\ and\ defeasible\ arguments, respectively \mid s \lozenge_{direct} d$ then $s > d\}$ ..... Equation (5.15)

- If a defeasible argument directly counter-argues a strict argument, then the strict argument defeats the defeasible argument. Mathematically, this is represented as follows:

$\forall s, d \in Args\{if\ s, d\ are\ strict$ and $defeasible\ arguments, respectively \mid d \lozenge_{direct} s$ $then\ s > d\}$ ..... Equation (5.16)

5.3.1.14 Dialectical tree

If an argument $\mathcal{A}$ counter-argues argument $\mathcal{B}$, and no static defeat exists, then a dialectical tree (as defined by (Garcia and Simari, 2004)) for argument $\mathcal{A}$ is constructed to determine whether argument $\mathcal{A}$ defeats argument $\mathcal{B}$ or vice versa.

Let $\mathcal{A}$ be an argument. A dialectical tree for argument $\mathcal{A}$ is $\Sigma(\mathcal{A}, \mathrm{h})$ where h is claim($\mathcal{A}$), is recursively defined as follows:

(1) A single node labeled with an argument ($\mathcal{A}$, h) with no counter-argument is by itself a dialectical tree for ($\mathcal{A}$, h). This node is also the root of the tree.

(2) Suppose that $\Sigma(\mathcal{A}, \mathrm{h})$ is an argument with counter-arguments ($\mathcal{A}_1$,h_1), ($\mathcal{A}_2$,h_2),.....,($\mathcal{A}_n$,h_n), The dialectical tree for ($\mathcal{A}$, h), $\Sigma(\mathcal{A}, \mathrm{h})$ is constructed by labeling the root node with ($\mathcal{A}$, h) and by making this node the parent of the root of dialectical trees for ($\mathcal{A}_1$,h_1), ($\mathcal{A}_2$,h_2),.....,($\mathcal{A}_n$,h_n) i.e. $\Sigma(\mathcal{A}_1, \mathrm{h}_1)$,$\Sigma(\mathcal{A}_2, \mathrm{h}_2)$,.....,$\Sigma(\mathcal{A}_n, \mathrm{h}_n)$. Figure 5.4 depicts the graphical representation of the dialectical tree.

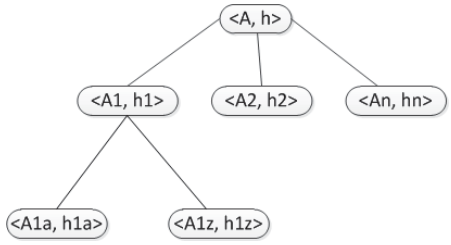


Figure 5.4: Pictorial representation of a dialectical tree

5.3.1.15 Marking of dialectical tree

To identify the priority between an argument and its counter-argument, the dialectical tree is marked as either *defeated or undefeated* as shown in Figure 5.5 . If the dialectical tree of an argument is marked defeated, then the argument has less priority over its counter-argument and vice versa. The marking of the dialectical tree (as defined by (Garcia and Simari, 2004)) is a two-step process as follows:

(1) Leaves of $\Sigma(\mathcal{A}, \mathrm{h})$ are U-nodes.

(2)Let $(\mathcal{B}, \mathrm{q})$ be an inner node of $\Sigma(\mathcal{A}, \mathrm{h})$. Then (B, q) will be a U-node iff every child of $(\mathcal{B}, \mathrm{q})$ is a D-node. The node $(\mathcal{B}, \mathrm{q})$ will be a D-node if it has at least one U-node as a child.

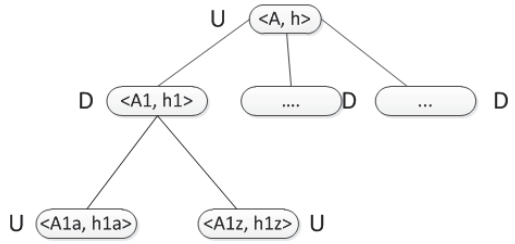


Figure 5.5: Pictorial representation of a marked dialectical tree

5.3.1.16 Dynamic defeat

If an argument r counter-argues argument s and no static defeat exits, then dynamic defeat is computed. Let $\Sigma_U(\mathcal{A}, \mathrm{h})$ be marked dialectical tree for argument $\mathcal{A}$ and $\Sigma_D(\mathcal{B}, \tilde{}\mathrm{h})$ is marked dialectical tree for its counter-argument $\mathcal{B}$, then argument $\mathcal{A}$ establishes its priority over its counter-argument $\mathcal{B}$ known as dynamic defeat. The dynamic defeat results in the establishment of the priority of an argument over its counter-argument which is known as a dynamic priority. Mathematically, dynamic priority is defined as follows:

$\forall r, s \in Args\{if\ r \Diamond s,\ \Sigma_U(r,h)\ ,\ \Sigma_D(s, \sim h)\}\ then\ r > s$ ..... Equation (5.17)

If an argument $\mathcal{A}$ has an undefeated dialectical tree i.e. $\Sigma_U(\mathcal{A}$, h) and it counter-argues an argument $\mathcal{B}$ which also has an undefeated dialectical tree i.e. $\Sigma_U(\mathcal{B}$, $\backsim$h), then neither argument A nor B can establish its priority over the other, resulting in a blocked situation. Such arguments are referred to as blocking arguments.

5.3.1.17 Sub-argument

Given an argument set $Args$, an argument s is a sub-argument of r if and only if claim(s) $\subseteq$ $premise(r)$ and, if there exists say, counter-argument g, then the marked dialectical tree of an argument s is undefeated and the marked dialectical tree of argument g is defeated. Mathematically, the condition for a sub-argument can be represented as follows:

$\forall r,s,g\ \{\ if\ (claim(s) \subseteq premise(r)\ and\ \ if(s \Diamond g)\ then\ s > g)\ then\ s\ \xi\ r\}$ .. Equation (5.18)

where ξ used to represent the sub-argument relationship between two arguments.

The sub-argument is a supporting argument and it must have the following characteristics:

1. argument s is consistent w.r.t argument r;
2. There is no $premise(s)$ such that $premise(s) \subseteq claim(r)$.

A sub-argument that provides support to another argument results in a chaining of arguments.

5.3.1.18 Reasoning chain

An argument $\mathcal{A}$ supported by a chain of sub-arguments produces a reasoning chain $\lambda_{\mathcal{A}}$=($\mathcal{A}_1$, ...,$\mathcal{A}_n$) for an argument $\mathcal{A}$. The claim of supported argument $\mathcal{A}$, is called a 'result' of the reasoning chain and the chain of sub-arguments is called a 'support' for the result of the reasoning chain. Mathematically, a reasoning chain is defined as follows:

$\forall r,\ s \in Args\ \{\ if(\ s\ \xi\ r)\ then\ \lambda_{(r,j)} = \lambda_{(r,,j)} \cup\ s$ ..... Equation (5.19)

where ξ is used to represent a sub-argument relationship and $\lambda_{(r,j)}$is used to represent a reasoning chain with result j. The reasoning chain should have the following characteristics:

1. The reasoning chain is consistent (i.e., there is no contradiction in the result and support for the result).
2. There is no defeated argument in a reasoning chain.
3. Two blocking arguments cannot be in the same reasoning chain.

5.3.1.19 Strict reasoning chain

A reasoning chain is considered to be strict if all the arguments in the reasoning chain are strict arguments. Mathematically, a strict reasoning chain can be represented as follows:

$\forall r, s \in \lambda_{(r,j)}\{r,s\ are\ strict\ arguments\}$ ..... Equation (5.20)

This reasoning chain cannot be directly counter-argued by other reasoning chains. However, this reasoning chain can counter-argue and defeat the rest of the reasoning chains in an argumentative production system.

5.3.1.20 Defeasible reasoning chain

A reasoning chain is a defeasible reasoning chain if all arguments in the reasoning chain are defeasible arguments. Mathematically, defeasible reasoning chains can be represented as follows:

$\forall d, f \in \lambda_{(r,j)}\{d\ ,\ f\ are\ defeasible\ arguments\}$ ..... Equation (5.21)

This reasoning chain can counter-argue or can be counter-argued by other reasoning chains in an argumentative production system. The defeasible arguments must be undefeated and consistent within the defeasible reasoning chain.

5.3.1.21 Mixed reasoning chain

A reasoning chain is a mixed reasoning chain if it has a least one defeasible and one strict argument. Mathematically, a mixed reasoning chain can be represented as follows:

$\forall r, s \in \lambda_{(r,j)}\{\exists r\ that\ is\ a\ defeasible\ argument\ ,\ \exists s\ that\ is\ a\ strict\ argument\}$..Equation (5.22)

5.3.1.22 Dependent reasoning chains

A reasoning chain is dependent upon other reasoning chains if there is at least one common sub-argument. If the common argument is a strict argument, then a reasoning chain is known as a strictly dependent reasoning chain; if it is defeasible argument, then it is weakly dependent and medium dependent if it contains more than one common argument and those common arguments include both strict and defeasible arguments. Mathematically, this is represented as follows:

$$if(\lambda_{(J,j)} \cap \lambda_{(H,h)}) \neq \varnothing\ then\ \lambda_{(J,j)} and \lambda_{(H,h)} are\ dependent\ reasoning\ chains.....$$

Equation (5.23)

5.3.2 Working of the proposed framework for Web@IDSS

In this section, the working of the proposed framework for Web@IDSS that can perform argumentative reasoning over incomplete and/or contradictory information which exists within an enterprise and/or in other enterprises is discussed and considered in decision making. As mentioned in Section 4.4.1, the proposed framework uses

the DeLP language to represent incomplete and/or contradictory information in a declarative format, and uses a hybrid reasoning engine to reason over it. Figure 5.6 presents a flowchart diagram of the working of the proposed framework. The sequence of steps in the proposed framework are as follows:

1. Information representation in DeLP format

   The Web@IDSS, located at the Web-based DSS layer, takes into account the structured information located at the information layer. To achieve this objective, Web@IDSS exploits the functionality of the *information representation module* of the logic-based framework located at @IRRI layer. This module helps the Web@IDSS to translate the structured information in RuleML format into DeLP rules (also called as production rules) and saves them in the rule base i.e $\mathcal{R}$. It also translates the structured information in OWL/RDF format to DeLP facts and saves them in the working memory i.e. $\mathcal{WM}$. Two translators have been developed to achieve this task as follows:

   - RuleML translator

     It translates the information specified in RuleML format to DeLP format. In most cases, the business rules of an enterprise are specified in RuleML format.

   - OWL/RDF translator

     It translates the information specified in OWL/RDF format to DeLP format. In most cases, the customer opinions, reviews/feedback about products and services offered by an enterprise are specified in OWL/RDF format.

2. Argumentative production system to perform hybrid reasoning

   Once the knowledge base (i.e. rule base containing DeLP rules and working memory containing DeLP facts) is formed, Web@IDSS exploits the functionality of the *argumentative reasoning module* of the logic-based framework to reason over the information present in the knowledge base. This process involves the following steps:

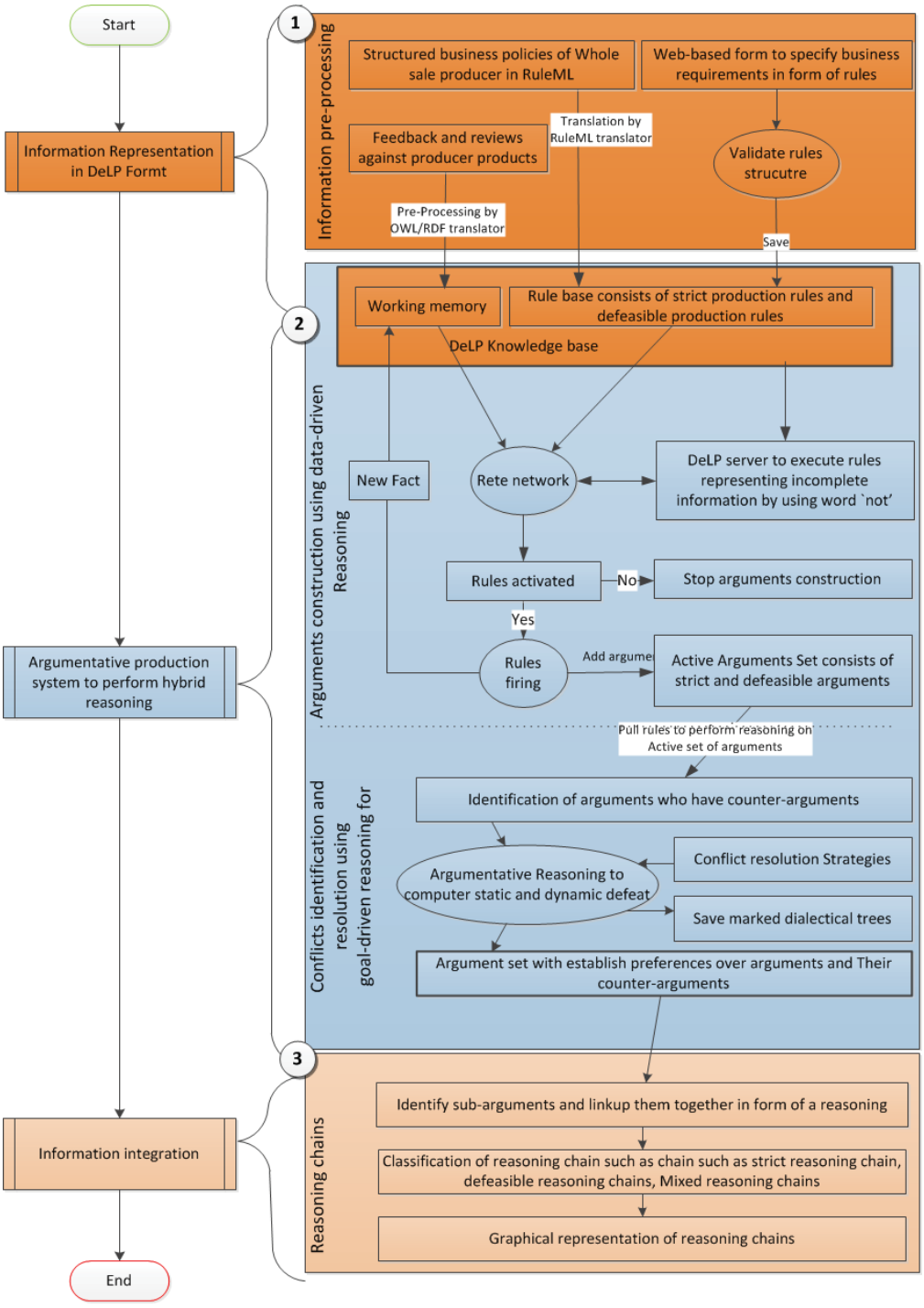


Figure 5.6: Flowchart illustrating steps performed by Web@IDSS for information representation, reasoning and integration

- Arguments construction using data-driven reasoning.

  As mentioned previously, the information in the knowledge base may be potentially incomplete and/or contradictory (representing different

viewpoints against a single issue) and current Web-based DSS are not able to perform reasoning over it. As a result, they can't assist the decision maker in the decision-making process. So, there is need to perform reasoning over such information and transform it into a format that is easily understood by the decision maker and can assist him in the decision-making process. In the proposed framework, this objective is achieved by transforming the incomplete and/or contradictory information in the knowledge base into a set of arguments. These arguments represent the different viewpoints in the underlying information in a declarative format. In the proposed framework, the construction of arguments involves two steps:

- Compilation of DeLP rules in the form of a Rete network.
- Perform data-driven reasoning by introducing certain DeLP facts from the working memory to the Rete network.

Data-driven reasoning results in the construction of a set of arguments supporting different conclusions. Two types of arguments which are constructed during this phase are strict arguments and defeasible arguments.

- Conflict identification and resolution using goal-driven reasoning

  Once the construction of arguments is complete, arguments which have counter-arguments are identified in order to resolve the conflicts between them and determine which one of them is defeated. To achieve this objective, two types of defeats are defined:

  - static defeat: a strict argument defeats a defeasible argument;
  - dynamic defeat: when there are two defeasible arguments in conflict with each other, then goal-driven reasoning is performed that uses a 'generalize specificity' conflict resolution strategy to resolve the conflict between them. During this process, a dialectical tree is constructed against the defeasible arguments that are in conflict and afterwards, each dialectical tree is marked as defeated or undefeated. These marked dialectical trees are used by the argumentative production system to resolve the conflict. The marked dialectical tree is then saved for future use, such as to provide an explanation for conflict resolution.

3. Information integration

   Once the conflicts have been resolved, Web@IDSS exploits the functionality of the *information and knowledge integration module* of the logic-based framework to integrate the information obtained from hybrid reasoning and display it to the decision maker. This process involves the following steps:

   - Construction of reasoning chains

     Once the hybrid reasoning is finished and conflicts have been resolved between arguments, the arguments need to be linked in the form of a chain. This module provides the functionality to link these arguments (supporting a conclusion) in the form of a chain know as a reasoning chain. During the construction of reasoning chains, different arguments supporting different conclusions result in the construction of different reasoning chains.

   - Categorization of reasoning chains

     After the construction of reasoning chains, the next step performed by this module is to classify them on the basis of arguments upon which they are built. The four categories defined in the proposed framework to categorise the reasoning chains are as follows:

     - strict reasoning chains: composed of strict arguments only;
     - defeasible reasoning chains: composed of defeasible arguments only;
     - mixed reasoning chains: composed of at least one strict argument and one defeasible argument;
     - dependable reasoning chains: composed of at least one argument that is shared with another reasoning chain.

   - Graphical representation of reasoning chains

     The last functionality performed by this module is the graphical representation of reasoning chains for the decision maker. This will assist the decision maker in understanding the conclusion of the reasoning process and how that conclusion has been reached. Additionally, such representation of a reasoning process will help the decision maker to easily communicate the results to non-technical people such as the CEO of an enterprise.

In the next sections, the working of each of these steps defined in the proposed framework for Web@IDSS will be discussed in detail.

5.4 Information representation in DeLP format

As discussed in Section 4.4.1, the DeLP language is used in the proposed framework to represent incomplete and/contradictory information in Web@IDSS. The structured information is in RuleML and OWL/RDF format. There is need for a translation mechanism to translate that information into DeLP format and use it in the decision-making process. The proposed framework addresses this drawback with the help of the *information representation module* of the logic-based framework located at the @IRRI layer. There are two ways to represent information in Web@IDSS as shown in the Figure 5.7. They are:

1. Information pre-processing

   During information pre-processing, the structured information is parsed and translated to DeLP format by using either the RuleML or OWL/RDF translator and is saved in the knowledge base.

2. Web-based form to specify DeLP rules and facts

   There may be some situations where there is no existing information available to be translated into DeLP format. In such cases, the decision maker has to specify the production rules by himself, depending on the objectives he wants to achieve. In the proposed framework, this objective is achieved by using a Web-based form to specify the DeLP rules and facts from scratch and saving them in the knowledge base.

In the next-subsections, each of these will be discussed in detail.

5.4.1 Information pre-processing

Information processing is carried out with the help of the following two translators:

(a) RuleML translator

RuleML supports different rule types via the 'implies' element and allows them to be named using the 'oid' element. RuleML syntax has been extended to express defeasible rules and superiority relations (Bassiliades et al., 2004) (Pham et al., 2008). A '@ruletype' attribute has been added to the 'implies' element, allowing it to take one of three values: strict rule, defeasible rule or defeater. For the translation of rules to DeLP format, the RuleML translator performs the following steps:

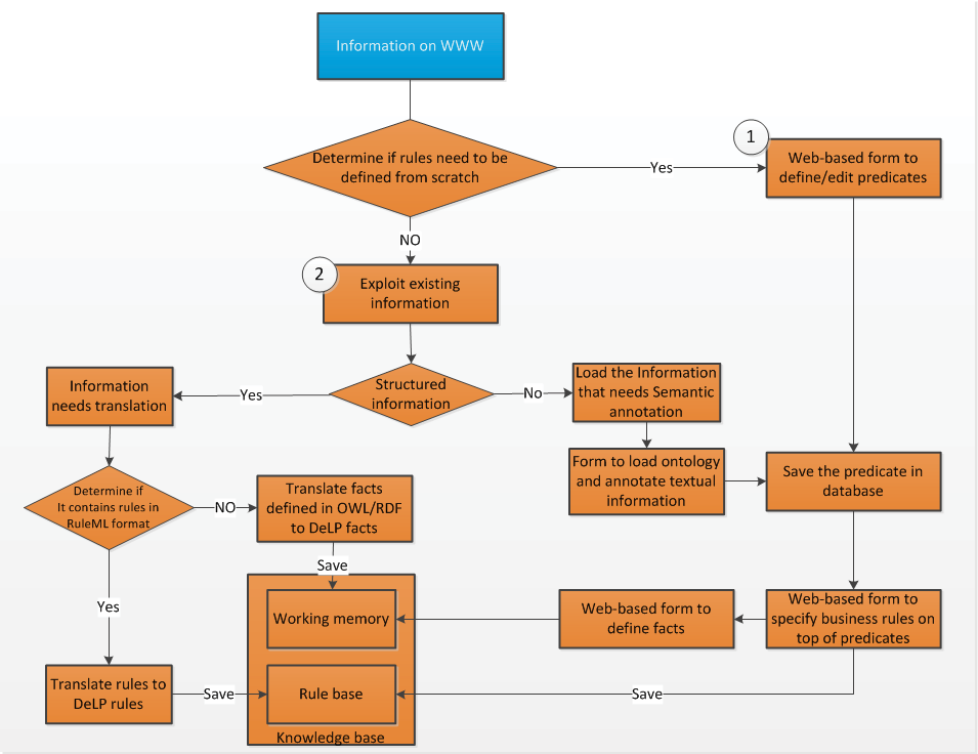


Figure 5.7: Flowchart illustrating steps for information representation in Web@IDSS

- It loads the RuleML file (XML format) and starts its paring from the root element.
- It iterates through all the rules specified in the RulML file and by looking at '@ruletype' tag, it classifies them as either strict production rules or defeasible production rules. If '@ruletype' is absent, then it considers that production rule as strict.
- Then, it takes up each parsed production rule and starts building production rules in DeLP format. Information with the 'Rel' tag is captured as a predicate in the body of the production rule and information with the 'Var' tag is captured as the subject and object variables in a predicate. If a parsed production rule head contains the 'Neg' tag, this is captured with the symbol '$\sim$' in the rule's conclusion. It this tag is found in the body of the parsed rule, it is captured with the symbol 'not' in the rule's body.
- After translation of each parsed rule to DeLP format, the production rules are then saved in the knowledge base.

The RuleML translator also saves other information about the RuleML file such

as file URL, the number of rules translated, the owner/creator of rules etc., in a database for their profiling.

To explain with an example, consider the case study mentioned in Section 5.2 where Mr. David has to download and consider the business policies of a supplier in the decision-making process. Figure 5.8 shows the representation of the business policy of a supplier in RuleML.

businesspolicy.txt - Notepad

File Edit Format View Help

```
<?xml version="1.0" encoding="UTF-8"?>
<RuleML xmlns="http://www.ruleml.org/0.91/xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ruleml.org/0.91/xsd http://debii.curtin.edu.au/~naeem/delp-valid-xsd.xsd">
    <Assert>
        <Implies ruletype="defeasiblerule">| <oid><Ind>d9</Ind></oid>
            <head>
                <Atom><Rel>purchase</Rel><Var>X</Var><Var>Y</Var></Atom>
            </head>
             <body>
                 <Atom><Rel>shopper</Rel><Var>X</Var></Atom>
                 <Atom><Rel>product</Rel><Var>Y</Var></Atom>
                <Atom><Rel>havefeedback</Rel><Var>F</Var><Var>Y</Var></Atom>
                <Atom><Rel>reviewRate</Rel><Var>G</Var><Var>F</Var></Atom>
            </body>
        </Implies>
        <Implies ruletype="defeasiblerule">
            <oid><Ind>d1</Ind></oid>
            <head>
                <Atom><Rel>giveDiscount</Rel><Var>X</Var></Atom>
            </head>
            <body>
                <Atom><Rel>shopper</Rel><Var>X</Var></Atom>
                <Atom><Rel>purchase</Rel><Var>X</Var><Var>Y</Var></Atom>
            </body>
        </Implies>
        <Implies ruletype="defeasiblerule">
            <oid><Ind>d2</Ind></oid>
            <head>
                <Neg>
                    <Atom><Rel>giveDiscount</Rel><Var>X</Var></Atom>
                </Neg>
            </head>
            <body>
                    <Atom><Rel>shopper</Rel><Var>X</Var></Atom>
                    <Atom><Rel>purchase</Rel><Var>X</Var><Var>Y</Var></Atom>
                <Atom>
                <Neg>
                    <Rel>advancePayment</Rel><Var>X</Var><Var>Y</Var>
                </Neg>
               </Atom>
            </body>
        </Implies>
        <Implies ruletype="defeasiblerule">
            <oid><Ind>d3</Ind></oid>
            <head>
                <Atom><Rel>giveDiscount</Rel><Var>X</Var></Atom>
            </head>
            <body>
                    <Atom><Rel>purchase</Rel><Var>X</Var><Var>Y</Var></Atom>
                    <Atom><Rel>shopper</Rel><Var>X</Var></Atom>
                    <Atom><Rel>bulkOrder</Rel><Var>X</Var><Var>Y</Var></Atom>
            </body>
        </Implies>
        <Implies ruletype="defeasiblerule">
            <oid><Ind>d5</Ind></oid>
           <head>
                <Neg>
                    <Atom><Rel>gstFree</Rel><Var>Y</Var></Atom>
                </Neg>
            </head>
            <body>
                <Atom><Rel>product</Rel><Var>Y</Var></Atom>
                <Atom><Rel>eShop</Rel><Var>Z</Var></Atom>
                    <Atom>
                  <Neg>
                    <Rel>packaging</Rel><Var>Y</Var><Var>Z</Var>
                  </Neg>
                </Atom>
            </body>
        </Implies>
        <Implies ruletype="strictrule">
            <oid><Ind>s1</Ind></oid>
            <head>
           <Atom><Rel>normalDiscount</Rel><Var>X</Var></Atom>
```

Figure 5.8: Business policy of the supplier specified in RuleML format

The Web@IDSS downloads and translates the supplier's policies specified in the RuleML to DeLP rules and saves them in the rule base. Illustration 5.1 represents

the set of DeLP rules extracted from the supplier's policy RuleML file.

$$\mathcal{R} = \left\{ \begin{array}{l} [d2]\ shopper(X), \text{product(Y)}, not\ advancePayment(X,Y) \dashrightarrow \sim giveDiscount(X) \\ [d3]\ shopper(X), purchase(X,Y), bulkOrder(X,Y) \dashrightarrow giveDiscount(X). \\ [d4]\ eShop(Z), packaging(Y,Z) \dashrightarrow gstFree(Y). \\ [d5]\ eShop(Z), not\ packaging(Y,Z) \dashrightarrow \sim gstFree(Y) \\ [s2] gstFree(Y), giveDiscount(X) \rightarrow ordinaryDiscount(X) \\ [s1]\ not\ gstFree(Y), giveDiscount(X) \rightarrow normalDiscount(X) \\ [d7]\ shopper(X), normalDiscount(X) \dashrightarrow platinumDiscount(X) \\ [d8]\ shopper(X), normalDiscount(X), plansSlowToPay(X) \dashrightarrow \sim platinumDiscount(X) \end{array} \right\} Illustration(5.1)$$

$\mathcal{R}$, in illustration 5.1, represents the rule base comprising strict and defeasible production rules. The rules with labels 's1' and 's2' are strict production rules, whereas the rest are defeasible production rules. Table 5.1 provides a description of each production rule in a natural language format.

rule label	Description
d2	If a shopper does not pay in advance for the product, he may not receive a discount
d3	If a shopper purchases a product in bulk, he may be given a discount
d4	If a product needs packaging in the shop, then GST may apply
d5	If a product does not need packaging, then GST may not apply
s2	If GST applies and the shopper has been given a discount, then he must receive an ordinary discount
s1	If there is no information about GST, then the shopper must be given an ordinary discount.
d7	If the shopper is given a normal discount, then he may be eligible to receive a platinum discount
d8	If the shopper plans to pay in installments (i.e. slow to pay) then he may be not be given a platinum discount.

Table 5.1: Description of the supplier's production rules translated by the RuleML translator

(b) OWL/RDF translator

Translation of OWL/RDF information into DeLP facts using the following steps:

(a) OWL/RDF information is transformed with the help of the SWI-Prolog RDF Parser (Wielemaker, 2011) into an intermediate triple format i.e. *rdf (Subject, Predicate,Object).*

(b) The intermediate triple format is further processed to transform the RDF statements into *Predicate(Subject, Object)* format.

(c) The facts in *Predicate(Subject, Object)* format are then saved in the knowledge base. The type attribute is further translated using the following formula: type(X,C)→C(X)

To explain with an example, consider the case study mentioned in Section 5.2, where Mr. David has to consider the customer's reviews/feedback about a supplier's product in his decision-making process. Figure 5.9 details the information representing the customer reviews/feedback about the supplier's product in RDF/XML format (represented as step 1). The OWL/RDF translator translates this information into an intermediate format (represented as step 2) and then into DeLP format (represented as step 3).

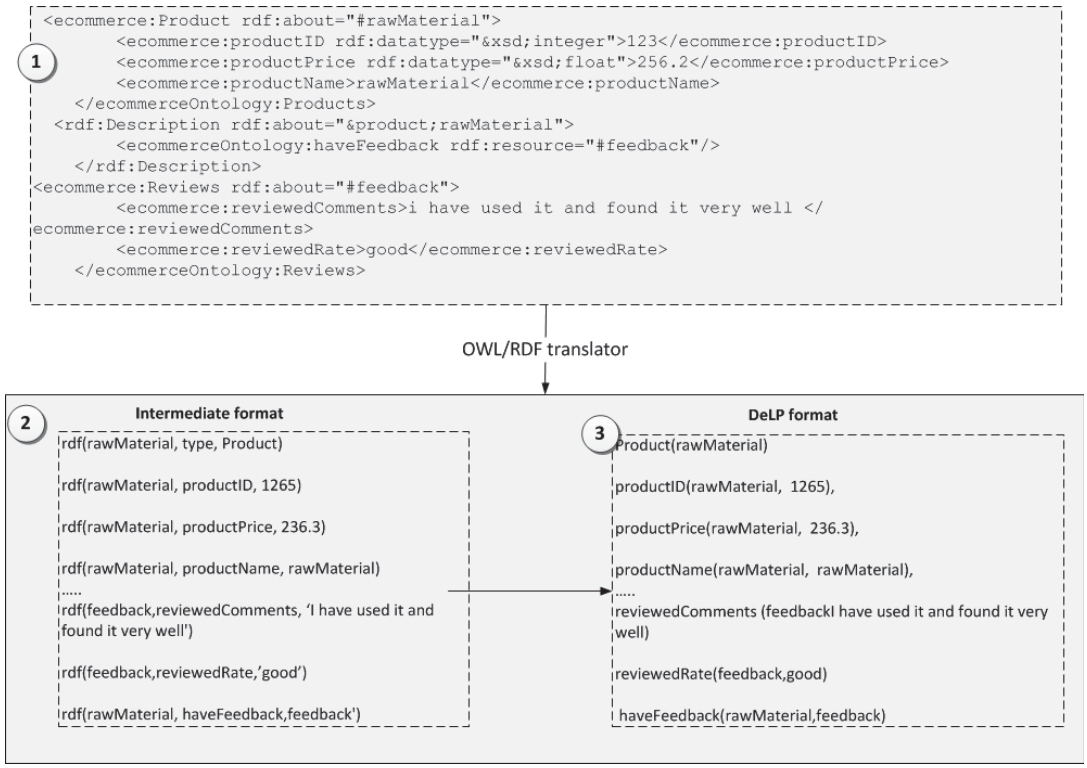


Figure 5.9: Pictorial representation of the process for translation of information in OWL/RDF format to DeLP facts

After translation of the customer's feedback, the OWL\RDF saves the DeLP facts in the working memory. Illustration 5.2 shows some of the DeLP facts that Mr. David will consider during the process of decision making.

$$\mathcal{WM} = \left\{ \begin{array}{l} eShop(BigW), product(rawMaterial) \\ havefeedback(rawMaterial, feedback), \\ reviwedRate(feedback, good) \end{array} \right\} \ldots\ldots\ldots\ldots Illustration(5.2)$$

Table 5.2 provides a description of each DeLP fact shown in illustration 5.2.

DeLP fact	Description
eShop(BigW),	The eShop i.e. supplier providing the product, is BigW.
product(rawMaterial)	The product is raw material
havefeedback(rawMaterial,feedback)	The raw material has some feedback
reviwedRate(feedback,good)	The feedback is good.

Table 5.2: Description of reviews/feedback by customer about supplier's production translated by OWL/RDF translator

5.4.2 Web-based form to specify DeLP rules and facts

Another way by which information can be represented in DeLP format is by using the Web-based form of Web@IDSS as shown in Figure 5.10. The Web-based form provides a GUI for the decision maker to define/edit DeLP rules and facts and saves them in the rule base and working memory, respectively. Using this form, Mr David can define his business requirements in the form of rules. For example, Mr David would like to purchase a product from a supplier who has good feedback from its customers. The following defeasible production rule captures his requirement in DeLP format:

- $[d9]$ $shopper(X)$, $product(Y)$, $havefeedback(Y,Z)$, $reviewedRate(Z,good)$ $-\!\!\rightarrow purchase(X,Y)$

Similarly, he wants to receive a discount on the purchase he may make. The following defeasible production rule captures his requirement in DeLP format:

- $[d1] shopper(X), purchase(X,Y)$ $-\!\!\rightarrow$ $giveDiscount(X)$.

Additionally, he also wants to specify that he may purchase the product in bulk. In DeLP language, such a parameter can be represented as a fact i.e. 'bulkOrder' using Web@IDSS form and saves it in the knowledge base.

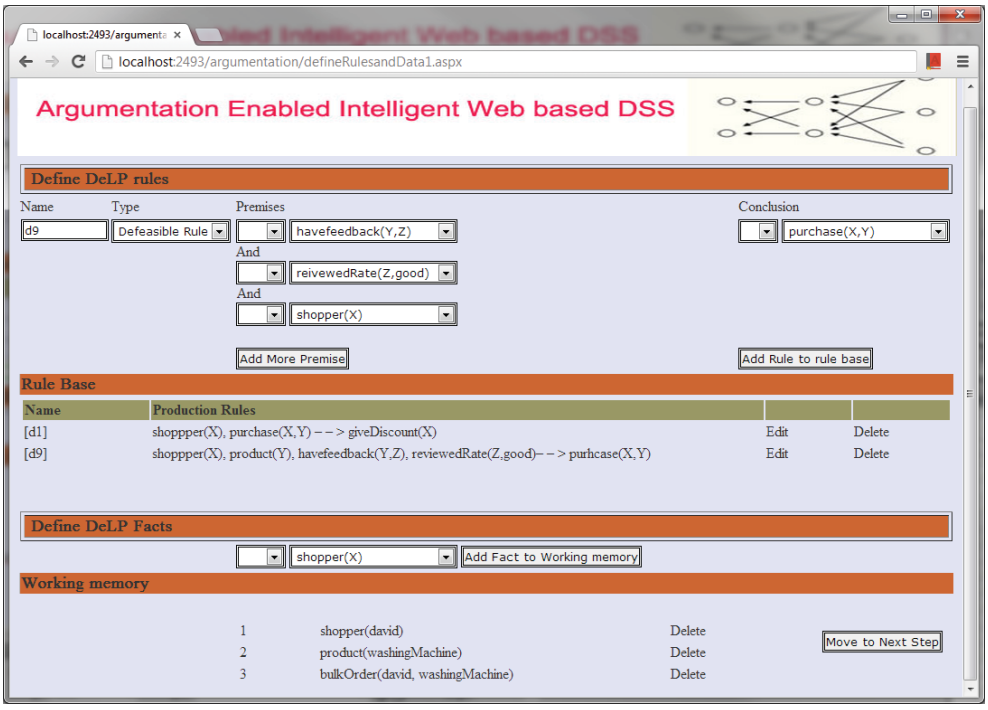


Figure 5.10: Web-based form for the decision maker to specify DeLP rules and facts

5.5 Argumentative Production System to perform hybrid reasoning

Once the required information for decision making has been captured in the knowledge base, then the next step is to perform reasoning over it. To address this objective, there is need for a hybrid reasoning methodology that can reason over the captured information and resolve any conflicts that may arise during the reasoning process. Web@IDSS achieves this objective with the help of an argumentative production system [1] that exploits the functionality of the *argumentative reasoning module* of the logic-based framework located at @IRRI layer. Figure 5.11 illustrates the steps performed by the argumentative production system for hybrid reasoning over the captured information. These steps are as follows:

[1] In Section 3.3.1, some important definitions which will help the reader to understand the design and working of the argumentative production system are introduced.

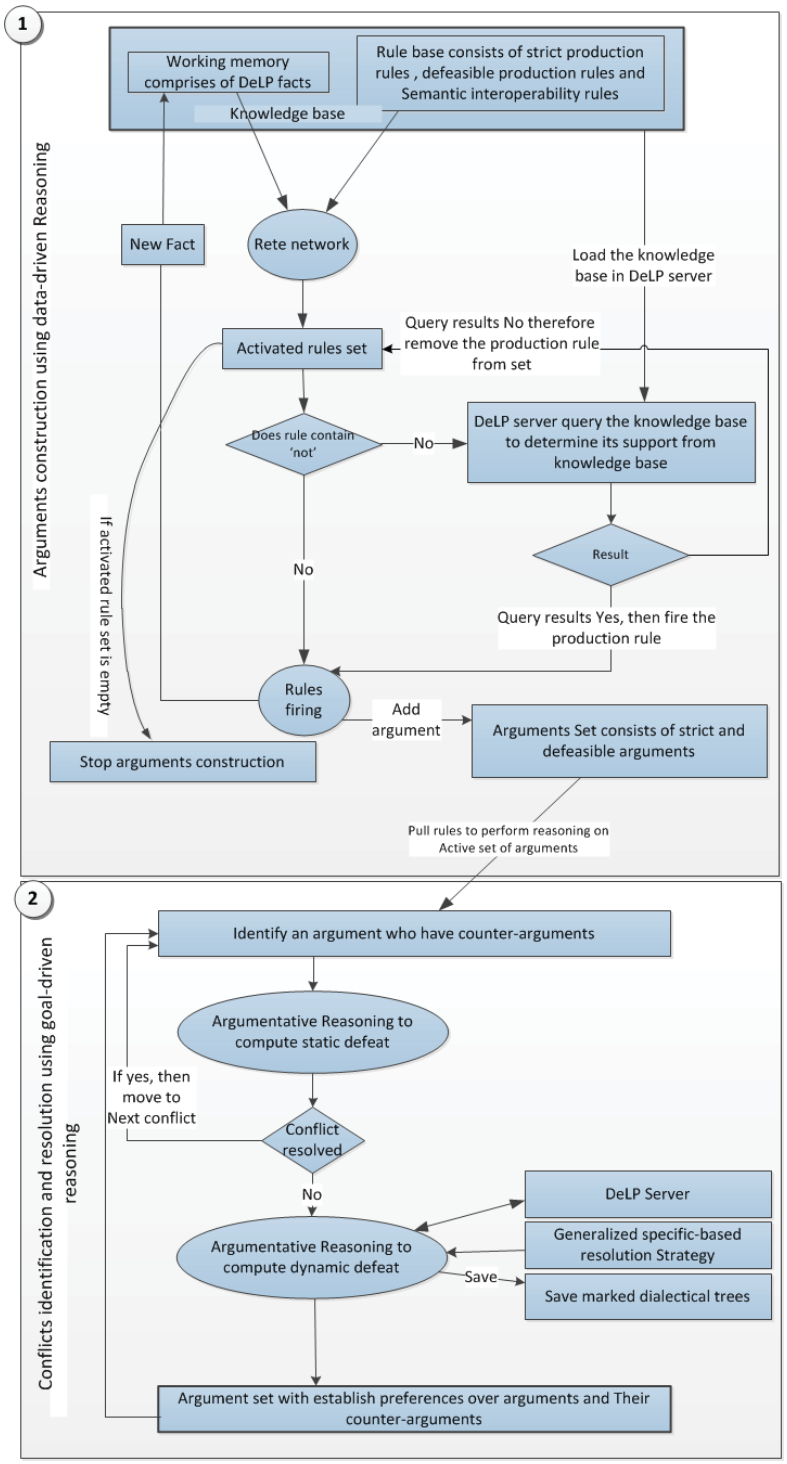


Figure 5.11: Flowchart illustrating steps performed by Web@IDSS during hybrid reasoning

1. Arguments construction using data-driven reasoning

   This step is further divided into the following two sub-steps:

- The first step is the compilation of production rules in the rule base in the form of a Rete network. In the proposed framework, the Rete network has been extended to represent incomplete and/or contradictory information as Rete nodes in the Rete network. Additionally, the single production rule execution strategy of the Rete algorithm has been extended to execute all production rules that are activated during data-driven reasoning.

- The second step is to perform data-driven reasoning over underlying information by passing the facts in the working memory through the Rete network. This results in the activation of production rules. The activation of production rules is followed by the firing of production rules. However, if the activated production rules' body represents some predicate starting with the symbol 'not', then before its firing, a query is sent to the DeLP server to compute its truthfulness by querying the knowledge base. If the query returns yes, then the production rule is fired, otherwise the activated production rules will be removed from the activated rule set. The firing of production rules results in the addition of new facts to the working memory and the instance of the production rule is stored as an argument in the 'argument set'. Data-driven reasoning is a recursive process that continues until no further production rules are activated.

2. Conflicts identification and their resolution using goal-driven reasoning

   Conflicts identification and their resolution is a recursive process consisting of the following three steps:

   - Identification of an argument and its counter-argument.

   - If static defeat exits, then the conflict between argument and its counter-argument is resolved by establishing a preference between them and control flows back to the first step.

   - In the case where static defeat does not exist between an argument and its counter-argument, then dynamic defeat is computed by using the 'Generalize Specificity' conflict resolution strategy. The outcome of the dynamic defeat computation is the marked dialectical trees of an argument and its counter-argument that helps the argumentative production system to establish the priority between them. Once the priority has been established, the arguments are saved again in the argument set.

In next sub-sections, each of these steps will be discussed in detail.

5.5.1 Arguments construction using data-driven reasoning

The construction of arguments from the knowledge base is a two-step process as follows:

1. Compilation of production rules in the form of a Rete network

   The arguments construction process starts with the compilation of the production rules present in the rule base as a Rete Network. A general Rete Network (Cirstea et al., 2004) consists of a network of nodes, each of which represents one or more predicates that make up the body of the production rules as shown in Figure 5.12. The three important nodes are as follows:

   - One-input nodes: These nodes are located at the first level of the Rete network and the facts from the working memory enter the Rete network through them. The different one-input nodes are as follows:
     - AssertCondition : The claim of a production rule is represented using this type of node.
     - RetractCondition : The claim can also be represented using this type of node if it may need to be removed later on from the working memory. In simple words, if contradictory information appears during the reasoning process, the general Rete network allows the removal of contradictory facts from the working memory in order to keep it consistent.
     - PositiveCondition: The predicates that make up the body of a production rule are represented using this type of node.
   - Two-input nodes: These are second level nodes in the Rete network and facts coming from the one-input node flow through to the two-input node and results in their activation.
   - Terminal nodes: These are the last level nodes, each of which represent the claim of a production rule. When all the incoming two-input nodes to the terminal nodes are activated, it results in the activation of terminal nodes and the instantiated claim represented by a terminal node is added to the working memory.

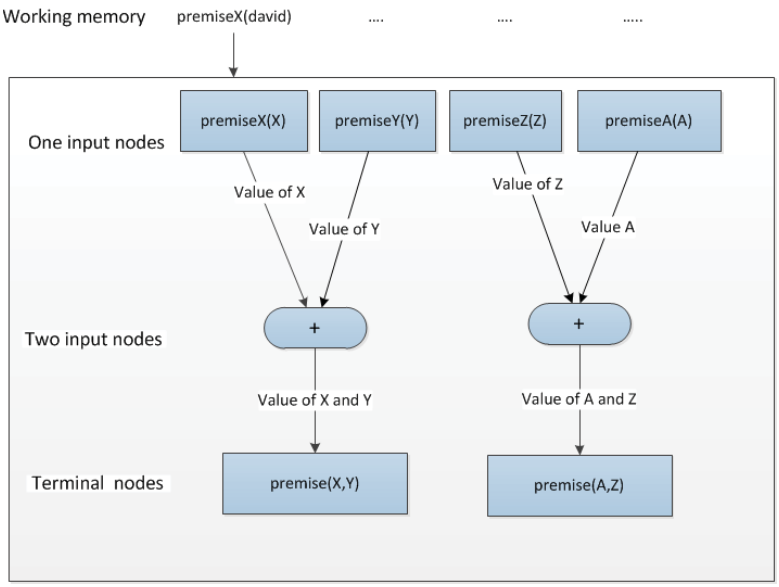


Figure 5.12: Simplified representation of the compilation of production rules in a general Rete network

In the proposed framework, the general Rete network has been extended to represent incomplete and/or contradictory information as Rete nodes in the network. The extensions made to one-input nodes are as follows:

- AssertCondition: The one-input nodes have been extend to represent contradictory information by introduction strong negation i.e.$\sim$, as an attribute in the AssertCondition class.
- NegativeConditionNAF: A new type of one-input node was introduced to indicate incomplete information represented by the symbol 'not'.

```
Variable x = new Variable("x");
Variable y = new Variable("y");
Variable z = new Variable("z");

Production prod1 = new Production();
prod1.LHS.Add(new PositiveCondition("", "shopper", x));
prod1.LHS.Add(new NegativeConditionNAF("", "advancePyament", y));
prod1.RHS.Add(new AssertCondition(y, "giveDiscount", x));
```

Figure 5.13: Code snippet that shows a production rule with NegativeConditionNAF

To explain the compilation of production rules in a Rete network, consider the rule base shown in illustration 5.1 where defeasible production rules d2 and d3 are translated from the supplier's business policies and defeasible production rule d1 is specified by Mr. David using the Web-based form of Web@IDSS as shown in Figure 5.10. Figure 5.14 shows the compilation of these three production rules

in the form of a Rete network. The predicates that make up the body of the production rules such as bulkOrder(X,Y), shopper(X) etc are represented as one input node and the claim of the production rules d1, d2 and d3 are depicted as terminal nodes. The nodes in between the one-input node and the terminal nodes are represented as two-input nodes.

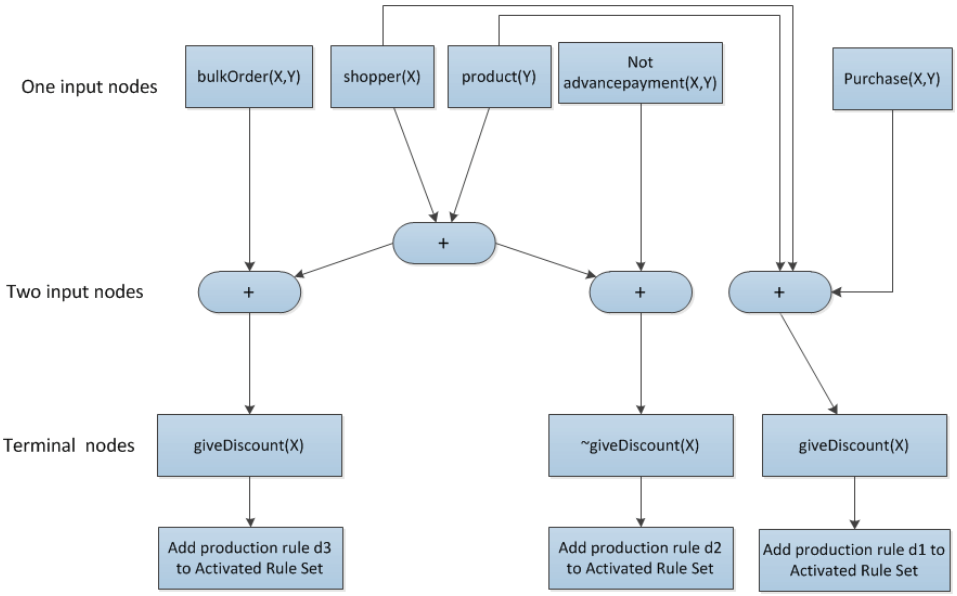


Figure 5.14: Compilation of production rules in the form of a Rete network in Web@IDSS

2. Perform data-driven reasoning over underlying information by passing the facts in the working memory through the Rete network

Once the production rules are compiled in the form of a Rete network, the next step is to perform data-driven reasoning by passing the facts in the working memory through the one-input nodes in the Rete network. This process, called data-driven reasoning, results in the activation of production rules called arguments (as defined in Section 5.3.1.9). Figure 5.15 illustrates the two important steps that are performed recursively during data-driven reasoning for the construction of arguments.

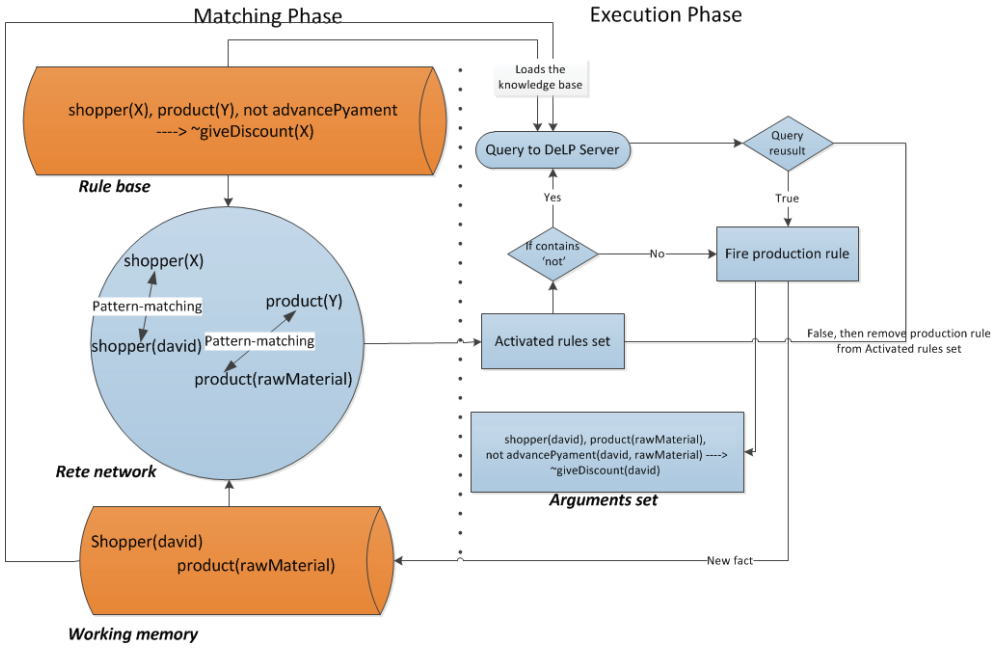


Figure 5.15: Data-driven reasoning by passing the facts through the Rete network in Web@IDSS

These steps are as follows:

- Matching phase: During this phase, pattern-matching is performed between the DeLP facts and the one input-node. If a pattern is matched, then the one-input node is activated and it forwards the value of the attributes to the two-input node. When two input-nodes receive the attribute value from all the incoming input nodes, it results in the activation of the respective terminal node.

- Execution phase: Once the terminal node is activated, the respective production rule is added to the Activated Rules set. The activation of production rules is followed by the firing of production rules. However, if the activated production rule's body represents a predicate starting with the symbol 'not', then before it is fired, a query is sent to the DeLP server to compute its truthfulness by querying the knowledge base. If it returns true, then the production rule is fired. Otherwise, it will be removed from the activated rules set. Firing the production rule will:

  - add a new fact to the working memory, and

  - add an instance of the rule to the argument Set.

It is important to note that in a general Rete network, data-driven reasoning works only on the production rules specified by an individual and the reasoning engine executes only one rule during a one match-execute cycle. If two rules are activated, the reasoning engine fires a production rule which has a higher preference specified by an individual at compilation time. In the proposed framework, the single rule execution strategy of the Rete network is removed as shown in Figure 5.16. Therefore, if two contradictory production rules are activated, both will fire and instances of both production rules i.e. arguments, are added to the argument set.

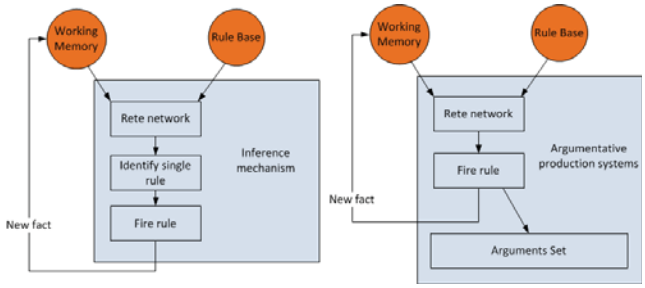


Figure 5.16: Comparison of a standard Rete with a single rule execution strategy (left) with the extended Rete without the strategy (right)

Data-driven reasoning will stop when no more production rules are activated. A key issue to be noted here is that such new inferred facts may conflict with the existing knowledge base. The purpose is to retain contradictory information instead of eliminating it, in order to obtain better insight when deciding on business strategies.

To explain the working of data-driven reasoning for argument construction over underlying information as shown in Figure 5.15 , consider the following production rule in the rule base:

- $[i]shopper(X), product(Y), not\ giveDiscount(X)$ $advancePayment(X,Y)$ $\dashrightarrow\!\!\sim$

Further consider a working memory that contains facts such as $shopper(david)$ and $product(Y)$. During the matching phase, pattern-matching is performed in the Rete network between the DeLP facts and one-input nodes that comprise the body of the production rules. On successful matching, the production rule is activated and is added into the Activated Rules set. Once the matching phase is finished, then execution phase is started. During the execution phase, if the activated production rule represents some

incomplete information such as that represented by production rule i i.e. $not\ advancePayment(X, Y)$, then the predicate representing the incomplete information is passed as a query (i.e. $not\ advancePayment(X, Y)$) to the DeLP server. The DeLP server loads the knowledge base and executes the query on it. The current knowledge base does not have any information regarding advanced payment, therefore the DeLP query returns true and production rule 'i' is fired. The firing of the production rule adds the derived fact (e.g. derived predicate $\sim$ $giveDiscount(david)$ as shown in Figure 5.15), into the working memory and the instance of the production rule (e.g. an argument $shopper(david), product(rawMaterial), notadvancePyament(david, rawMaterial)$ $\dashrightarrow$ $giveDiscount(david)$ as shown in Figure 5.15), is added to the arguments set.

Algorithm 1 demonstrates data-driven reasoning over underlying information by passing the facts in the working memory through the Rete network. It takes in the production rules specified in DeLP format and results in the construction of a set of arguments.

```
Data: DeLP rules and DelP facts .
Result: Arguments.
initialization;
Construct Rete network alpha and beta nodes etc ;
initialize Rete network;
bool constructArgument ←true;
ActiveRules ActiveRuleSet;
ActiveArgumentSet ArgsSet;
repeat
    foreach rule re ∈ KnowledgeBase do
        if match(re, WM)=true then
            ActiveRuleSet ←ActiveRuleSet ∪ re;
        else
            constructArgument ←false;
        end
    end
    foreach re ∈ ActiveRuleSet do
        WM ← WM ∪ claim(r);
        ArgSet ← ArgSet+interpretationOf(re); end
    until constructArgument ←true ;
```

Algorithm 5.1: Argument construction using data-driven reasoning

To explain argument construction using data-driven reasoning, consider an argumentative production system that captures information identified in the case study discussed in Section 5.2. This argumentative production system is named a $\mathbb{WEBIDSS}$. In light of the definition of an argumentative production system in Section 5.3.1.7, $\mathbb{WEBIDSS}$ can be defined as follows:

$$\mathbb{WEBIDSS} = (\mathcal{WM}, \mathcal{R}, Args) \quad \text{Equation (5.24)}$$

where

$$\mathcal{WM} = \left\{ \begin{array}{l} shopper(david), eShop(BigW), product(rawMaterial) \\ havefeedback(rawMaterial, feedback), \\ reviwedRate(feedback, good), bulkOrder(david, rawMaterial) \end{array} \right\}. \; Illustration(5.3)$$

$$\mathcal{R} = \left\{ \begin{array}{l} [s2]gstFree(Y), giveDiscount(X) \rightarrow ordinaryDiscount(X) \\ [s1]not\ gstFree(Y), giveDiscount(X) \rightarrow normalDiscount(X) \\ [d1]shopper(X)product(Y), purchase(X,Y) \dashrightarrow giveDiscount(X) \\ [d2]shopper(X)notadvancePayment(X,Y) \dashrightarrow \sim giveDiscount(X) \\ [d3]shopper(X), purchase(X,Y), bulkOrder(X,Y) \dashrightarrow giveDiscount(X). \\ [d4]eShop(Z)packaging(Y,Z) \dashrightarrow gstFree(Y). \\ [d5]eShop(Z)notpackaging(Y,Z) \dashrightarrow \sim gstFree(Y) \\ [d7]shopper(X)normalDiscount(X) \dashrightarrow platinumDiscount(X) \\ [d8]shopper(X)normalDiscount(X)slowToPay(X) \\ \dashrightarrow \sim platinumDiscount(X) \\ [d9]shopper(X)product(Y)havefeedback(Y,Z) \\ reviwedRate(Z, good) \dashrightarrow purchase(X,Y) \end{array} \right\} Illustration(5.4)$$

$$Args = \{\} \ldots\ldots Illustration(5.5)$$

The $\mathcal{WM}$ represents the working memory of $\mathbb{WEBIDSS}$ and contains DeLP facts as shown in illustration 5.3. $\mathcal{R}$ represents the rule base of $\mathbb{WEBIDSS}$ and contains production rules in DeLP format as shown in illustration 5.4. $Args$ represents the arguments set of $\mathbb{WEBIDSS}$ and contains no argument as shown in illustration 5.5.

The $\mathbb{WEBIDSS}$ captures all the information i.e. feedback about the supplier's production and Mr. David's facts in $\mathcal{WM}$, the business policies of the service provider and Mr. David as production rules in $\mathcal{R}$, and an empty arguments set. Given the definition of consistency in Section 5.3.1.8, in $\mathbb{WEBIDSS}$ the set {s1,s2} is consistent, whereas set {d1,d2} is inconsistent. It is important to note that production rule 'd1' is defined by Mr. David and argument 'd2' is a production rule representing the supplier's policy.

After arguments construction using data-driven reasoning over information in $\mathbb{WEBIDSS}$, it results in the following:

$$\mathbb{WEBIDSS} = (\mathcal{WM}\prime, \mathcal{R}, Args) \quad \text{Equation (5.25)}$$

where $\mathcal{WM}\prime$ represents the new state of the working memory after the addition of the new inferred facts. The argumentative production system with the updated working memory and populated with the argumentation is as follows:

$$\mathcal{WM}\prime = \left\{ \begin{array}{l} shopper(david), eShop(BigW), product(rawMaterial) \\ havefeedback(rawMaterial, feedback), \\ reviwedRate(feedback, good), bulkOrder(david, rawMaterial) \\ purchase(david, rawMaterial), \sim gstFree(rawMaterial), \\ giveDiscount(david), \sim giveDiscount(david), \\ normalDiscount(david), platinumDiscount(david), \end{array} \right\} \ldots\ldots\ldots\ldots\ldots\ Illustration(5.6)$$

$$\mathcal{R} = \left\{ \begin{array}{l} [s2] gstFree(Y), giveDiscount(X) \rightarrow ordinaryDiscount(X) \\ [s1] not\ gstFree(Y), giveDiscount(X) \rightarrow normalDiscount(X) \\ [d1] shopper(X) product(Y), purchase(X, Y) \dashrightarrow giveDiscount(X) \\ [d2] shopper(X) notadvancePayment(X, Y) \dashrightarrow \sim giveDiscount(X) \\ [d3] shopper(X), purchase(X, Y), bulkOrder(X, Y) \dashrightarrow giveDiscount(X). \\ [d4] eShop(Z) packaging(Y, Z) \dashrightarrow gstFree(Y). \\ [d5] eShop(Z) notpackaging(Y, Z) \dashrightarrow \sim gstFree(Y) \\ [d7] shopper(X) normalDiscount(X) \dashrightarrow platinumDiscount(X) \\ [d8] shopper(X) normalDiscount(X) slowToPay(X) \\ \dashrightarrow \sim platinumDiscount(X) \\ [d9] shopper(X) product(Y) havefeedback(Y, Z) \\ reviwedRate(Z, good) \dashrightarrow purchase(X, Y) \end{array} \right\} \ldots\ldots\ldots\ Illustration(5.7)$$

$$Args = \left\{ \begin{array}{l} [d1]\ shopper(david),\ purchase(david, rawMaterial) \\ \dashrightarrow giveDiscount(david) \\ [d2]\ shopper(david),\ not\ advancePayment(david,\ rawMaterial) \\ \dashrightarrow \sim giveDiscount(david) \\ [d3]\ shopper(david), purchase(david, rawMaterial), \\ bulkOrder(david, rawMaterial) \dashrightarrow giveDiscount(david). \\ [d5] eShop(BigW),\ not\ packaging(BigW, rawMaterial) \dashrightarrow \\ \sim gstFree(rawMaterial) \\ [s1] not\ gstFree(rawMaterial), giveDiscount(david) \\ \rightarrow normalDiscount(david) \\ [d7]\ shopper(david),\ normalDiscount(david) \dashrightarrow platinumDiscount(david) \\ [d9]\ shopper(david), product(rawMaterial), \\ havefeedback(rawMaterial, feedback), \\ reviewRate(feedback,\ good) \dashrightarrow purchase(david, rawMaterial) \end{array} \right\} \ldots Illustration(5.8)$$

Illustration 5.8 represents the set of arguments constructed during the arguments construction phase. From illustration 5.8, it can be seen that argument 's1' is a strict argument and the rest of the arguments i.e. d1, d2, d3, d5, d7 and d9 are defeasible arguments. These arguments represent the viewpoints of the supplier against Mr.

David's business requirements in relation to whether to give him a discount and if so, how much. To explain with help of example, consider the following three defeasible arguments from illustration 5.8:

- $[d1] shopper(david), product(rawMaterial),$
  $purchase(david, rawMaterial)$ --⇢ $giveDiscount(david)$

- $[d2] shopper(david), product(rawMaterial),$
  $not\ advancePayment(david, rawMaterial)$ --⇢∼ $giveDiscount(david)$

- $[d3] shopper(david), purchase(david, rawMaterial),$
  $bulkOrder(david, rawMaterial)$ --⇢ $giveDiscount(X)$

Argument 'd1' represents Mr. David's viewpoint on receiving a discount. It states that he purchases the raw material, and as a result, he expects a discount. However, argument 'd2' from the supplier states that Mr. David purchased the product but did not make a payment in advance, so he may not receive a discount. However, another arguments 'd3' from the supplier states that Mr. David purchased the product and placed a bulk order, so he may receive a discount.

5.5.2 Conflicts identification and their resolution using goal-driven reasoning

Once the argument construction process is complete, the conflicts identification and resolution phase is initiated. This is a recursive process that consists of the identification of an argument and its counter-argument and resolving the conflict between them. During this process, the following two types of defeats are computed in order to resolve conflicts between arguments:

1. Static defeat

   As defined in Section 5.3.1.13, a static defeat exits between an argument and its counter-argument if a strict argument defeats its defeasible counter-argument. This results in the establishment of the priority of a strict argument over a defeasible counter-argument. To explain static defeat with help of an example, consider an argument s1 from illustration 5.8 as follows:

   - $[s1] not\ gstFree(rawMaterial),$
     $giveDiscount(david)$--⇢ $normalDiscount(david)$

Further consider a defeasible argument that states that if Mr. David is eligible for a discount, then he may not be given a normal discount as he is a bad customer. The defeasible argument is represented as follows:

- [d11] $badCustomer(david)$,$giveDiscount(david)$--→∼ $normalDiscount(david)$

Argument 's1' is now in conflict with argument 'd11', however, there exits static defeat between them because a strict argument always defeats its defeasible counter-argument. Therefore, Mr. David will receive a normal discount.

2. Dynamic defeat by using the Generalize Specificity conflict resolution strategy to resolve conflict

   If static defeat does not exist between an argument and its counter-argument, the argumentative production system resolves the conflict between them by computing dynamic defeat. To achieve this objective, the argumentative production system takes into account the DeLP built-in *General Specificity conflict resolution strategy* (Garcia and Simari, 2004). In this strategy, an argument (itself or with the help of other arguments) is considered specific over its counter-argument if it requires less information to reach the final result. To understand the working of this strategy, consider the following two arguments:

   (a) $[p1r2]executiveManager(X)$ --→ $approveTravel(X)$ which states that if X is an executive manager, then he may approve travel.

   (b) $[p1r1]executiveManager(X), universityOfficer(Y), authorise(X,Y)$ --→∼ $approveTravel(X)$ which states that if X is an executive manager and he authorises Y who is a university officer to approve travel, then X may not approve travel.

   Considering the General Specificity conflict resolution strategy, the <p1r1, ∼ $approveTravel$>argument is more specific than the <p1r2, $approveTravel$> if the following two conditions hold:

   (a) for all strict rules i.e. H and facts F, H ⊆ F, if $approveTravel(jon)$ is derived defeasibly and no strict derivation of $approveTravel(jon)$ exists, then the defeasible derivation of ∼ $approveTravel(jon)$ exists, and

   (b) there exists H′ ⊆ F such that on basis of H′ drives defeasibly ∼ $approveTravel(jon)$ and there is no strict derivation of it and it does not drive defeasibly $approveTravel(jon)$.

Using General Specificity, the conflict between each argument and its counter-argument is resolved and the priority is saved in the knowledge base. The next step is to build the dialectical trees (as defined in Section 5.3.1.14) in order to identify whether this priority is supported by the entire knowledge base of the argumentative production system (in simple words, is there any argument that may counter-argue the preferred argument). This objective is achieved as follows:

- The claim of a preferred argument is submitted to the DeLP server (along with its preference over the counter-argument) and in return, it gives the marked dialectical tree (as defined in Section 5.3.1.15) of an argument. Similarly, the same procedure is performed for its counter-argument.
- Once the argumentative production system has the marked dialectical trees for both the argument and its counter-argument, an argument is preferred over its counter-argument if the marked dialectical tree of an argument is undefeated and the marked dialectical tree of counter-argument is defeated. In the case where the marked dialectical tree of both the argument and its counter-argument are undefeated, those arguments are considered blocking arguments and the system needs human intervention to resolve the conflict between them.

Algorithm 2 provides the detailed working of dynamic defeat using the General Specificity conflict resolution strategy to resolve conflicts by taking into account algorithm 3 i.e. building and marking dialectical trees. The marked dialectical tree for argument d1 with undefeated status is represented as $\Sigma_U(d1,\text{giveDiscount(david)})$.

Data: Arguments set.
Result: Preference establishment
initialization;
foreach arg_i *in* $ArgSet$ **do**
  if $arg_i \Diamond arg_{i+1}$ **then**
    $\Sigma_{status}(arg_i,h_i) \leftarrow$ BuildDialecticalTree(arg_i,h_i);
    $\Sigma_{status}(arg_{i+1},h_{i+1}) \leftarrow$ BuildDialecticalTree(arg_{i+1},h_{i+1});
    if $\Sigma_D(arg_i,h_i)$ *and* $\Sigma_U(arg_{i+1},h_{i+1})$ **then**
      $arg_{i+1} > arg_i$;
    end
    if $\Sigma_U(arg_i,h_i)$ *and* $\Sigma_D(arg_{i+1},h_{i+1})$ **then**
      $arg_i > arg_{i+1}$;
    end
    if $\Sigma_B(arg_i,h_i)$ *and* $\Sigma_B(arg_{i+1},h_{i+1})$ **then**
      $arg_i <> arg_{i+1}$;
    end
  end
  end

Algorithm 5.2: Dynamic defeat using the General Specificity conflict resolution strategy

Data: $(\mathcal{A}, h)$
Result: $\Sigma_{status}(\mathcal{A}, h)$
Let C $\leftarrow$ get all counter-arguments of $(\mathcal{A}, h)$; **if** $C \neq \emptyset$ **then**
  while *there is no* $\Sigma_U(\mathcal{A}_i, h_i) \in C$ **do**
    for *every argument in* C **do**
      Let $(\mathcal{A}_i, h_i) \leftarrow$ minimal non-labelled element
      BuildDialecticalTree($(\mathcal{A}_i, h_i)$) getting result as $\Sigma(\mathcal{A}_i, h_i)$;
      Put $\Sigma(\mathcal{A}_i, h_i)$ ξ $(\mathcal{A}, h)$
    end
    if *there exist some* $\Sigma_U(\mathcal{A}_i, h_i)$ **then**
      Set $\Sigma_D(\mathcal{A}, h)$;
    else
      Set $\Sigma_U(\mathcal{A}, h)$;
    end
  end
else
  $\Sigma(\mathcal{A}, h) = (\mathcal{A}, h)$;
  Set $\Sigma(\mathcal{A}, h) \leftarrow defeated$;
end

Algorithm 5.3: Building and marking of a dialectical tree

To explain conflict identification and resolution using goal-driven reasoning with the help of an example, consider illustration 5.8, where argument 'd1' counter-argues argument 'd2', and argument 'd3' counter-argues argument 'd2', as shown in Figure 5.17.

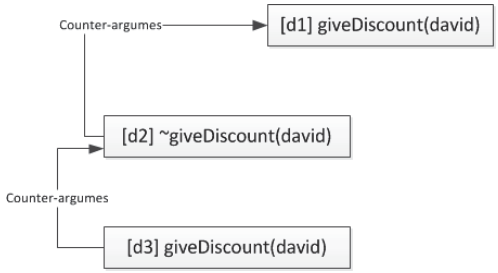


Figure 5.17: Pictorial representation of arguments and their counter-arguments from illustration 5.8

The WEBIDSS resolves the conflicts between the arguments and their counter-arguments using the General Specificity conflict resolution strategy. This task involves the following two steps:

(a) Identify conflict and establish the priority between an argument and its counter-arguments.

In this step, an argument e.g. d2, which is more specific than its counter-argument e.g. d1, defeats its counter-argument and results in priority establishment represented as follows: d2 >d1, as shown in Figure 5.18. Similarly, argument 'd3' defeats argument 'd2' which results in its priority establishment as follows: d3 > d2.

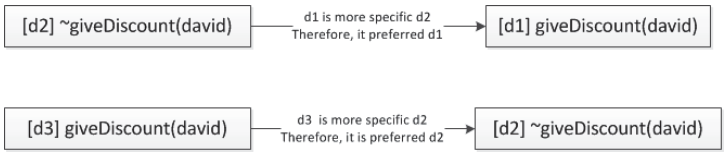


Figure 5.18: Pictorial representation of preference between arguments using Generalize Specificity

(b) Building and marking of dialectical tress to obtain the priority status of an argument over its counter-argument by considering the entire knowledge base.

During this phase, the claims of argument d1 i.e. $giveDiscount(david)$ and the claims of its counter-argument d2 i.e. $\sim$ $giveDiscount(david)$, along with the priority between them is sent to the DeLP server to perform goal-driven reasoning and it returns their marked dialectical trees. The construction of marked dialectical trees is defined in Section 5.3.1.15. Figure 5.19 shows a

marked dialectical tree for argument 'd1' and 'd2'. In the figure, an argument is represented in the short form e.g. $[d1]giveDiscount(david)$ where [d1] is the label of the argument and $giveDiscount(david)$ is the claim of the argument.

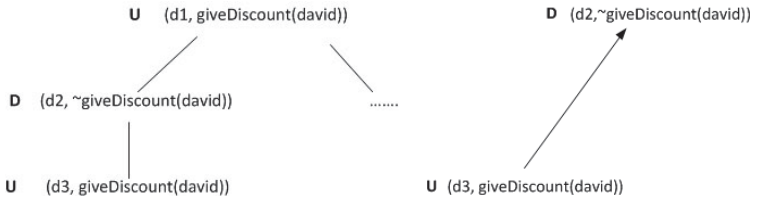


Figure 5.19: Pictorial representation of undefeated marked dialectical tree for argument d1 (left), defeated marked dialectical tree for argument d2 (right)

It is evident from the figure that the marked dialectical tree for argument 'd1' is undefeated and the marked dialectical tree for argument 'd2' is marked as defeated in the tree. Therefore, argument d1 with the undefeated marked tree is preferred over argument 'd2' which has a defeated marked dialectical tree. It is important to note that in the undefeated marked dialectical tree for argument 'd1', argument 'd2' is marked as defeated. However, before the construction and marking of dialectical trees (as shown in Figure 5.18), argument 'd2' was preferred over argument 'd1'. However, during this step, argument 'd2' is attacked by argument 'd3'. Argument 'd3' is more specific than argument 'd2', therefore, 'd3' is preferred over argument 'd2' (as shown in Figure 5.18). The preference of argument 'd3' over 'd2' results in the revival of argument 'd1'. Therefore, in simple words, argument 'd3' supports argument 'd1' to withstand the attack of argument 'd2'. As a result, the marked dialectical tree for argument 'd1' is undefeated.

Once the marked dialectical trees are computed, argument 'd1' with the undefeated marked dialectical tree is preferred over its counter-argument 'd2' with the defeated marked dialectical tree. Such dialectical analysis of arguments helps decision makers such as Mr. David to understand that even though he may not have paid in advance (i.e. represented as argument $[d2]shopper(david), product(rawMaterial),$
$not\ advancePayment(david, rawMaterial)$ $\dashrightarrow\!\!\sim$ $giveDiscount(david))$, by placing the bulk order (i.e. represented by argument $[d3]shopper(david), purchase(david, rawMaterial),$
$bulkOrder(david, rawMaterial)$ $\dashrightarrow$ $giveDiscount(X))$, he may be offered a discount.

5.6 Information integration

Once the argumentative production system has performed hybrid reasoning over the underlying information, the next step is to integrate the results of hybrid reasoning and display the results to the decision maker in a graphical format to assist him in the decision-making process. To achieve this objective, Figure 5.20 illustrates the following important steps in the proposed framework for Web@IDSS to integrate information:

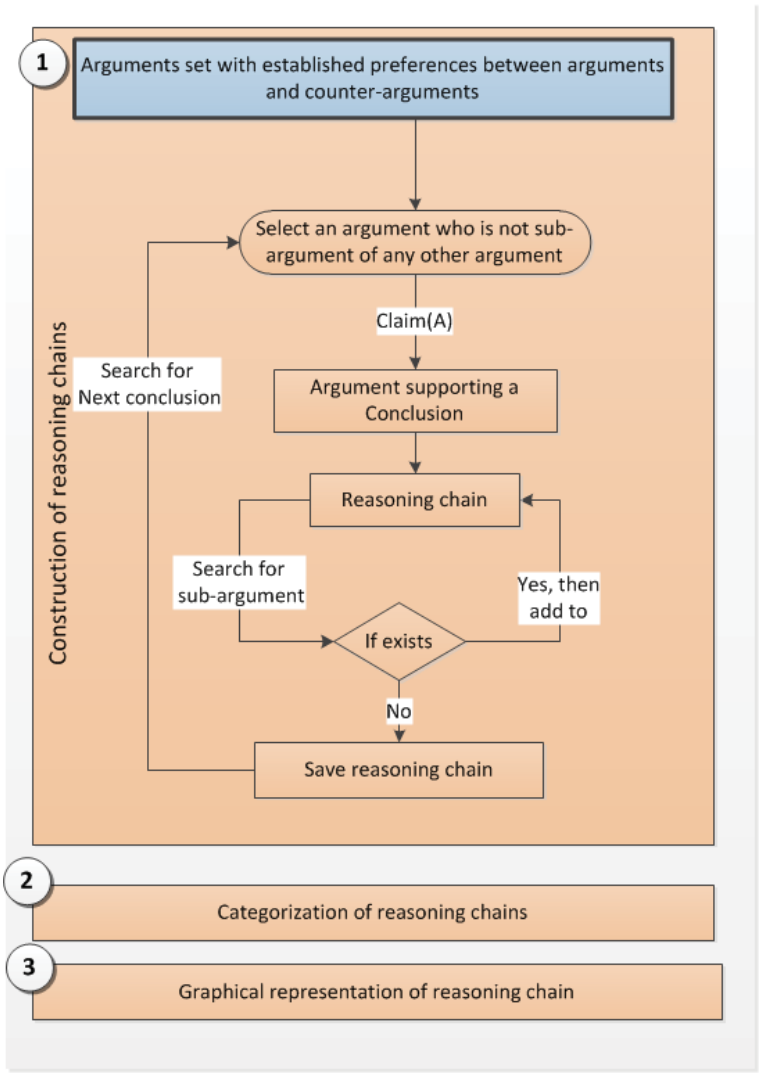


Figure 5.20: Flowchart illustrating steps performed by Web@IDSS for information integration

1. Construction of reasoning chains

   During this step, the arguments are linked together in the form of a reasoning chain. Such reasoning chains help to link the initial information that triggers the

reasoning process to reach the final conclusion. It also explains the important steps/information derived during reasoning to reach the final conclusion.

2. Categorization of reasoning chains

   Reasoning chains represent a decision supported by an argument or chain of arguments. Depending on the nature of argument/s supporting the decision, the argumentative production system can categorise a reasoning chain either as strict, defeasible, mixed or dependable. Such categorization of reasoning chains helps the decision maker to identify the strength/weakness of the information supporting the final decision.

3. Graphical representation of a reasoning chain

   During this step, the reasoning chain is depicted in a graphical format provide the decision maker with more easily comprehendible results. The graphical representation of results also helps him to easily communicate the results to higher authorities who may be non-technical people.

In the next subsections, each of these steps will be discussed in detail.

5.6.1 Construction of reasoning chains

The first step in information integration is the construction of reasoning chains. This process involves the following steps:

1. all sub-arguments with undefeated dialectical trees are linked together as a reasoning chain. This process will continue until all possible arguments are linked to form a reasoning chain;

2. the top argument i.e. conclusion of the reasoning chain is called the 'result' of the reasoning chain, and the chain of arguments supporting the top argument are called to support the conclusion;

3. ensure the reasoning chain is consistent (i.e., there is no contradiction in the result and support for the result).

Algorithm 4 provides the working of the construction of a reasoning chain by Web@IDSS.

Data: $(\mathcal{A}, h)$
Result: $\lambda_{(\mathcal{A},h)}$
1 Let S ← get all sub-arguments of $(\mathcal{A}, h)$;
2 **if** $S \neq \emptyset$ **then**
3 **foreach** $(\mathcal{A}_i, h_i) \in S$ **do**
4 **if** *noCounterArgument*$(\mathcal{A}_i, h_i)$ *or* $\Sigma_U(\mathcal{A}_i, h_i)$ **then**
5 BuildReasoningChain($(\mathcal{A}_i, h_i)$) ;
6 Put $\lambda(\mathcal{A}_i, h_i)$ ξ $(\mathcal{A}, h)$;
7 **end**
8 **end**
9 **else**
10 $\lambda_{(\mathcal{A},h)} = (\mathcal{A}, h)$;
11 **end**

Algorithm 5.4: Construction of a reasoning chain

To explain the construction of a reasoning chain, Figure 5.21 depicts a graphical representation of a reasoning chain constructed from the arguments shown in illustration 5.8. The reasoning chain is divided into two parts, namely, the result of the reasoning chain i.e. platinum discount for Mr. David, and support for the result i.e. supporting information that backs the decision to give a platinum discount to Mr. David. By using such a graphical representation, Mr. David can identify that

- he may receive a discount by placing a bulk order (i.e. $[d3]shopper(david), purchase(david, rawMaterial), bulkOrder(david, rawMaterial)$ --→ $giveDiscount(X)$ and is depicted as 1 in Figure 5.21);
- he can identify that the raw material he needs are not GST free, so as a result, he may be given a normal discount (i.e. $[s1]not\ gstFree(Y), giveDiscount(X) \rightarrow normalDiscount(X)$ and it is depicted as 1,2 and 3 in Figure 5.21);
- lastly, he can identify that if he receives a normal discount, there is a chance he may receive a platinum discount as well (i.e. $[d7]shopper(david), normalDiscount(david)$ --→ $platinumDiscount(david)$ as depicted as 3 and 4 in Figure 5.21).

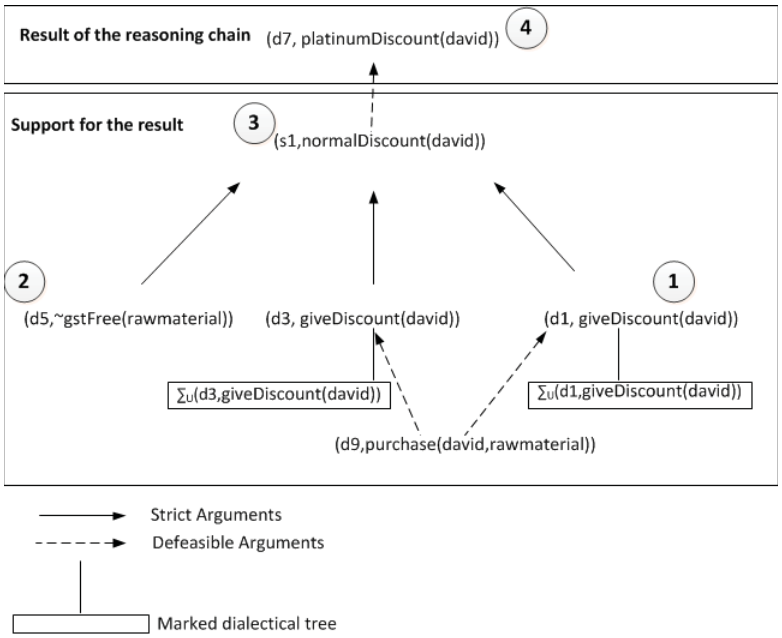


Figure 5.21: Pictorial representation of mixed reasoning chain generated from arguments show in illustration 5.8

As mentioned in Section 5.3.1.18 where the formal definition of a reasoning chain was provided, it was pointed out that a reasoning chain should adhere to the following characteristics:

1. The reasoning chain should be consistent (i.e., there is no contradiction in the result and support for the result). Therefore, for example, $giveDiscount(david)$ and $\sim giveDiscount(david)$ will not belong to one reasoning chain, but each one of them can belong to different reasoning chains and those reasoning chains represent alternative paths or choices.

2. There is no defeated argument in a reasoning chain.

3. Two blocking arguments cannot be in the same reasoning chain.

5.6.2 Categorization of reasoning chains

A reasoning chain represents a set of small decisions linked to support the final conclusion. Depending on the nature of the arguments that are constructed during the reasoning process, the reasoning chains can be classified into the following different categories:

- Strict reasoning chain

Such reasoning chains represent a decision process which cannot be challenged, even when new arguments are introduced into the argumentative production system.

- Defeasible reasoning chain

  Such reasoning chains represent decisions which can be challenged by the introduction of new arguments in argumentative production systems.

- Mixed reasoning chain

  Such reasoning chains have less weak points compared to defeasible reasoning chains which may be challenged by the introduction of new arguments in the argumentative production system which may result in a different conclusion.

- Dependent reasoning chain

  A reasoning chain is called dependent if it shares information with other reasoning chains. Such shared information points provide an alternative path in the decision-making process. If such information points are challenged by the introduction of new arguments in the argumentative production system, then the conclusions may change dramatically. Figure 5.22 shows two reasoning chains $\lambda_{(j3,h)}$and $\lambda_{(s4,j)}$ sharing a common argument i.e. (s3,g).

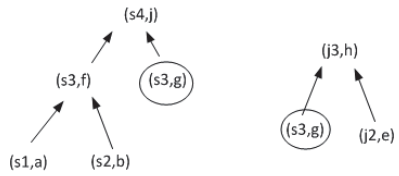


Figure 5.22: Pictorial representation of dependent reasoning chains $\lambda_{(j3,h)}$and $\lambda_{(s4,j)}$

To explain the categorization of reasoning chains with help of an example, consider a reasoning chain categorized as a mixed reasoning chain by WEBIDSS, and as depicted in Figure 5.21. Such categorisation of a reasoning chain will help Mr David to identify the weak points (defeasible arguments) providing support to the overall conclusion. If new information arises later on, it may result in the defeat of those defeasible arguments, leading to different conclusions. For example, if he does not purchase in bulk, he will not only lose the opportunity to receive a discount, it will result in his failure to receive a normal discount and eventually a platinum discount.

5.6.3 Graphical representation of a reasoning chain

The last functionality performed by the information and knowledge integration module of the logic-based framework is the graphical representation of reasoning chains. To explain with help of an example, consider Figure 5.23 which represents the graphical implementation of a reasoning chain depicted in Figure 5.21. The important features of the graphical representation of a reasoning chain are as follows:

- The reasoning chain is represented as an inverted tree.
- An argument is represented in short form e.g. $[s1]normalDiscount(david)$ where [s1] is the label of the argument and $normalDiscount(david)$ is the claim of the argument.
- The *facts* are depicted as oval shapes, the *arguments* are depicted as rectangular shapes. The defeasible inference is depicted with a dotted arrow and strict inference as a straight arrow.

Such graphical representation helps Mr David to understand the whole reasoning process that results in a conclusion i.e. plantinumDiscount(david). He can easily identify the weak points (represented as defeasible inference) in a reasoning process where, if new information arises, this may result in the retraction of existing information, eventually leading to different results.

The graphical representation of the reasoning process takes into account the business policies of a supplier i.e. BigW, and the customer's feedback on its products in the decision-making process. In order to formulate a strategy for product B, Mr David has to perform the same activity with the rest of the suppliers in order to identify the supplier who may offer a maximum discount. As a result, he will obtain 'n' number of reasoning chains, each of which provides a different degree of discount under different conditions. By going through the graphical representations of the reasoning chains, Mr David can easily identify a supplier who may offer him a maximum discount considering his business requirements and the conditions with more strict rules. The graphical representation of the reasoning process will also help him to communicate his decision to the enterprise's CEO about why and how he reached the decision to select a particular supplier for raw material for the development of a new product.

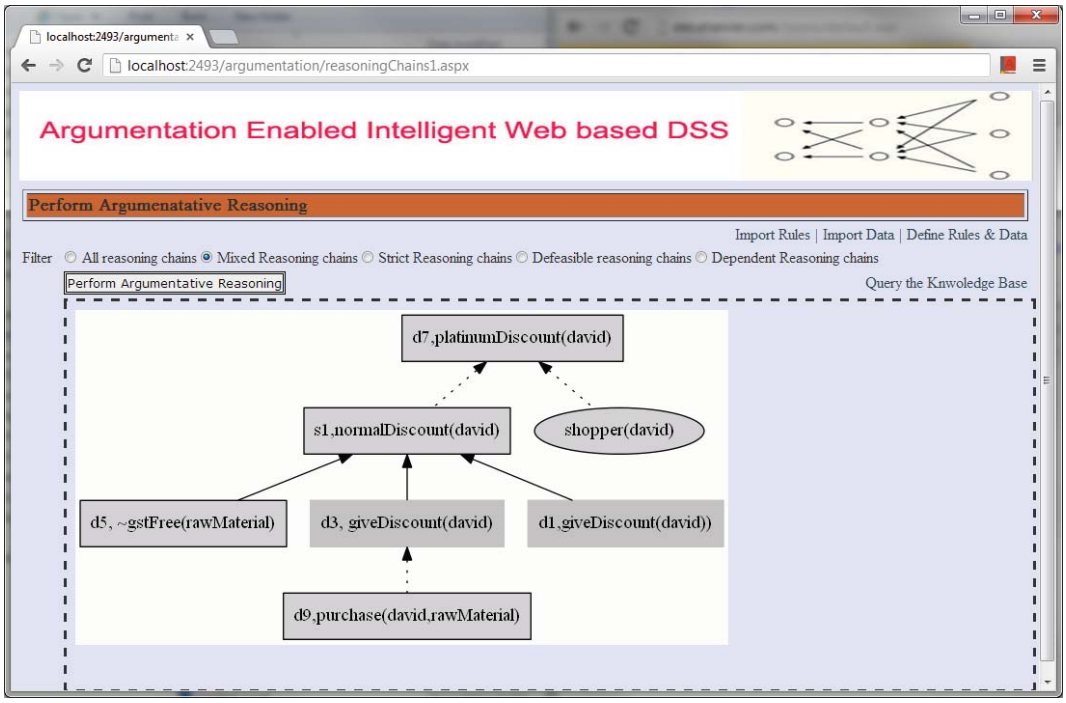


Figure 5.23: Graphical representation of the reasoning chain generated by Web@IDSS

5.7 Conclusion

In this chapter, the major shortcomings of the existing Web-based DSSs are addressed i.e. inability to represent and handle incomplete and/or contradictory information exists within an enterprise and/or in other enterprises. This is particularly important for those enterprises who take into consideration the information available on the Web for timely and accurate decision support. To overcome the limitations, the syntax and semantics of the DeLP language to represent, reason and integration information in Semantic Web applications is defined. A conceptual framework for 'Argumentation-enabled Web IDSS' is proposed and its workings are described in detail with the help of a case study.

In Chapter 3, I outlined certain research objectives that need to be addressed in order to support argumentation in Semantic Web applications. Some of the objectives have been addressed in this chapter and how they can be applied in Semantic Web applications for decision making when underlying information is inconsistent and incomplete are as follows:

- A methodology for incomplete and inconsistent information representation
  - A rule-based declarative language i.e. defeasible logic programming (DeLP) was selected for incomplete and/or inconsistent knowledge representation

on the Semantic Web. DeLP allows information representation i.e. specifications or preferences, that can be taken into account by the Web-based DSS and considered in the reasoning process to produce customized results for the decision maker.

  - A RuleML translator that translates the information defined in the RuleML to DeLP format was proposed. Such translation enables the exploitation of information which already exists on the Semantic Web specified in RuleML format.

  - An OWL/RDF translator that translates the information defined on the Semantic Web in the form of OWL and RDF into DeLP facts. The translated data is exploited by the rules during the reasoning process.

- A methodology for an argumentation driven-reasoning engine to reason over incomplete and inconsistent information

  - A hybrid reasoning engine to reason over information represented in DeLP format was proposed. The hybrid reasoning engine performs two types of reasoning: firstly, data-driven reasoning for argument construction and goal-driven reasoning for conflict identification between arguments and resolution.

  - A methodology to resolve conflicts among arguments by using the DeLP built-in Generalize Specificity conflict resolution strategy was proposed.

- Proposed a methodology for Information Integration

  - A mechanism to integrate the information being produced by different argumentation-driven hybrid reasoning engines was proposed and its graphical representation was provided to the decision maker to enhance their understanding of the reasoning process and results.

Chapter 6 - Enterprise Knowledge Integration through Argumentation-enabled Intelligent Decision Support Systems (Web@KIDSS)

6.1 Introduction

In Chapter 5, an Argumentation-enabled Web-based IDSS (Web@IDSS) was proposed to help decision makers represent, reason and integrate information that exits within their enterprise and/or in other enterprises and assist them in the decision-making process. This information integration is called as Enterprise Information Integration (EII).

With the current proliferation and widespread adoption of e-business, enterprises conduct business on an global level, and are often involved in collaborations and mergers with other enterprises on a global scale (Norta and Eshuis, 2010; Alaranta and Henningsson, 2008). In such cases, the decision/reasoning results produced by an enterprise's Web-based DSS might need to be integrated with other Web-based DSS located within the enterprise and/or in other enterprises to obtain a comprehensive picture of the problem at an enterprise level to enable business managers to obtain better business insights and make better decisions. This information integration is called as Enterprise Knowledge Integration (EKI) as depicted in Figure 6.1. EKI may be defined as an integration of the decisions/results generated from different Web-based DSS that may be potentially incomplete and/or contradictory into a single coherent knowledge to support either intra-enterprise decision making (i.e. decision making processes involving the departments of an enterprise) or inter-enterprise

decision making (i.e. decision-making processes involving departments located in different enterprises). However, as pointed out in Section 2.8, the current generation of Web-based DSS are not able to represent, reason and integrate information that may be potentially incomplete and/or contradictory to provide support for either the intra-enterprise or inter-enterprise decision-making process. Due to this limitation, they do not provide any solution for EKI. To address this, in this chapter, a framework for Argumentation-enabled Web-based IDSS for EKI (Web@KIDSS) is proposed that provides a solution for EKI to facilitate either the intra-enterprise or inter-enterprise decision-making process.

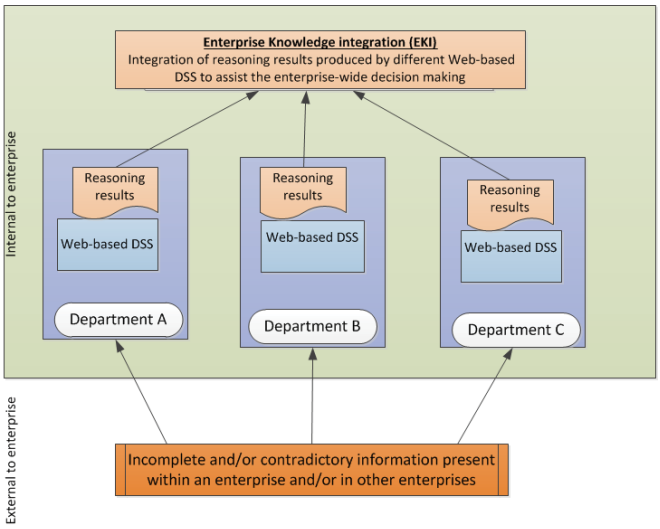


Figure 6.1: Interaction of an enterprise's internal and external environment for Enterprise Knowledge Integration (EKI)

The proposed framework imports the reasoning chains generated by Web@IDSS and extends its functionality to publish them in a standard format over the enterprise's intranet or the Internet. It then transforms the standard reasoning chains to DeLP format, evaluates them against the decision maker's defined criteria defined as an integration scheme which is then followed by their integration using argumentative reasoning. The development of Web@KIDSS will advance the research in Web-based DSS as depicted in Figure 6.2.

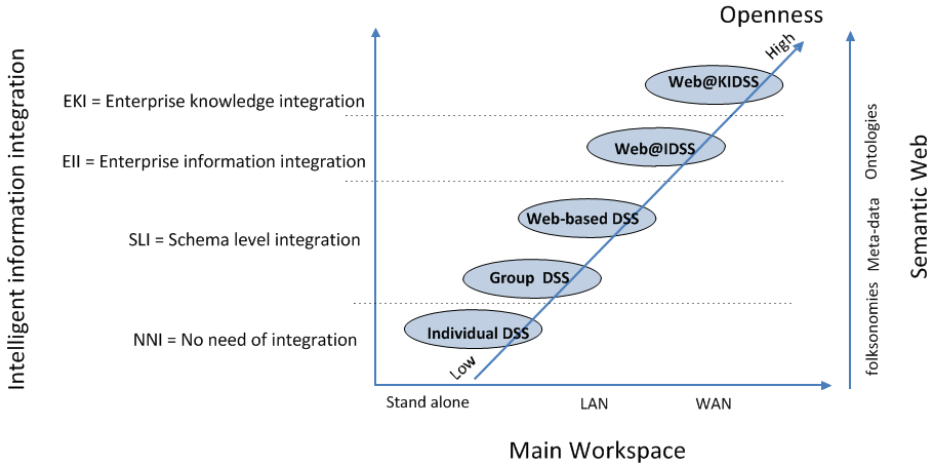


Figure 6.2: Evolution towards Intelligent Information Integration in an enterprise

The organization of this chapter is as follows: in Section 6.2, an outline of the problem to be addressed by using a case study to highlight the requirements and challenges for EKI in enterprise-wide decision making is given. In Section 6.3, an overview of the proposed Argumentation-enabled Web-based IDSS for EKI (Web@KIDSS) is provided. In Sections 6.4 to 6.6, each component of the proposed framework is explained in detail and how it provides a solution to the problem highlighted in the case study is discussed. Section 6.7 concludes the chapter.

6.2 Case study for problem definition

To explain the problem, consider the example of enterprise ABC, comprising different departments such as Information Technology (IT), Marketing (Mar) and Human Resources (HR), which has decided to relocate its departments to a new site. The business manager responsible for overseeing the move has instructed the manager of each department to provide recommendations and justifications for engaging the services of the XYZ relocation service to assist with the move by considering information such as:

- business-related information for the XYZ relocation service provider published on the Web along with the department's business requirements, and
- customer feedback on the services provided by the XYZ relocation service provider

Figure 6.3 depicts the possible interaction between the internal environment of enterprise ABC and the external environment that comprises information such as XYZ

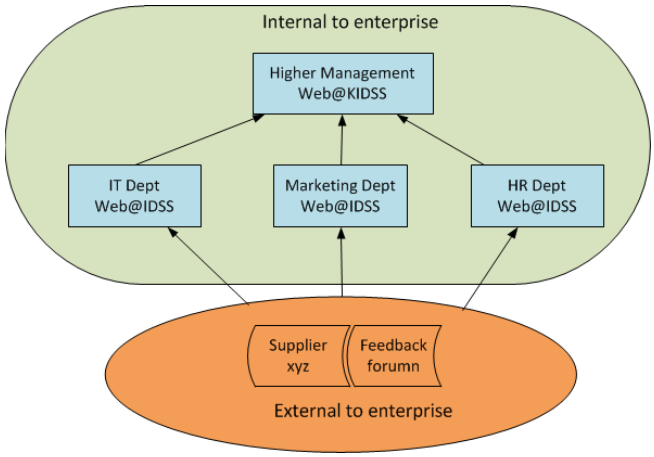


Figure 6.3: Interaction of enterprise ABC with external environment

business policies and customer feedback on the services provided XYZ etc. In order to generate recommendations for relocation service provider XYZ, each department's manager uses Web@IDSS to represent, reason and integrate XYZ business information with their departmental information (requirements) and customer feedback. Once each department in the ABC enterprise forwards its recommendation to the business manager, the latter will need to integrate the diverse recommendations about relocation service provider XYZ from each department into a coherent knowledge base which could help the business manager to reach a final decision, i.e. whether or not to engage the services of the XYZ relocation service provider. The challenges that confront the business manager are as follows:

- recommendations from each department need to be in a standard, shareable format;
- recommendations from each department need to be evaluated by testing them against the decision maker's defined criteria. Such evaluation helps the decision maker to decide on the scope of the recommendation chain and whether or not to consider it for EKI;
- conflicts within and between the different recommendations need to be addressed in relation to the inter-enterprise or intra-enterprise decision making process.

In order to overcome the abovementioned challenges, the business manager requires a Web-based IDSS that offers the following functionalities:

- an interface to download the recommendations and save them in the knowledge base;

- an interface to define and apply an integration scheme over the recommendations saved in the knowledge base in order to evaluate their scope and whether or not to consider them for EKI;
- a reasoning mechanism that can resolve the conflicts between diverse recommendations and integrate them into an integrated knowledge base that may assist the decision makers in the intra-enterprise or inter-enterprise decision-making process;

To develop a Web-based IDSS with the abovementioned functionalities, the business manager's requirements for a Web-based DSS for EKI are formalized as follows :

- sharing of recommendations in a standard format, such as AIF, that is understandable by other Web-based DSS in the inter-enterprise or intra-enterprise decision-making process;
- integration of recommendations which involves the definition and application of the integration scheme and argumentative reasoning to identify and resolve conflicts;
- justifiable explanation of the reasoning process and the results achieved;
- capability to provide a graphical representation of the reasoning process performed to achieve EKI in order to make it easily understood by non-technical persons such as the business manager.

Assumptions

- Enterprises ABC and XYZ and the feedback forum share a common vocabulary defined in OWL/RDF format and the predicates defined in the vocabulary are used for the specification of business rules.
- A declarative language for specifying the imported recommendation in the form of reasoning chains for knowledge integration in an enterprise.
- A declarative language with the capability of representing incomplete and contradictory information represented by reasoning chains.
- Information integration via an inference mechanism that can perform reasoning pertaining to incomplete and contradictory information from different sources.
- Reasoning chains have been produced by different departments of an enterprise by using Web@IDSS.

6.3 Proposed framework for Argumentation-enabled Web-based IDSS for Enterprise Knowledge Integration (Web@KIDSS)

In this section, the proposed framework for the Argumentation-enabled Web-based IDSS for EKI is presented i.e. integrating the decisions/recommendations generated by different Web-based DSS into a coherent knowledge base to support the inter-enterprise or intra-enterprise decision making process. The proposed framework, presented in Figure 6.4 consists of three layers as follows:

1. Information layer

   The information layer represents the structured information identified by decision makers for consideration during the decision-making process. This information comprises different reasoning chains published over the enterprise's intranet or the Internet by different Web-based DSS located both within the enterprise and/or in other enterprises.

2. @IRRI layer

   This layer comprises a logic-based framework that enables Semantic Web applications such as Web-based DSS, to deal with information in the form of reasoning chains (e.g. recommendations generated by different Web@IDSS) which are potentially incomplete and/or contradictory and considers them for inter-enterprise or intra-enterprise decision making. It provides different modules to transform a reasoning chain into a standard, shareable format such as Argument Interchange Format (AIF) for publication on an enterprise's intranet or the Internet. Additionally, it also provides a solution for considering different published reasoning chains, integrating them after applying the integration scheme and performing argumentative reasoning to resolve conflicts between them followed by representing them graphically to the decision maker to assist him in the decision-making process. The modules are as follows:

   (a) *Information and knowledge integration module* provides functionalities for:

   - the transformation of a reasoning chain generated by a Web@IDSS into AIF format and its publication in OWL/RDF format over the enterprise's intranet or the Internet. This ensures information from heterogenous sources is in a standard format to achieve EKI.

- Enterprise knowledge integration which involves the following steps:
  - Import the published reasoning chains in AIF format and transform them into DeLP format (DeLP rules and facts) and save them in the knowledge base. Such transformation of reasoning chains from AIF to DeLP format enables the hybrid reasoning engine to perform the next steps.
  - Valuation of the reasoning chains which involves the following steps:
    - Re-construction of the reasoning chains from the knowledge base with the help of the *argumentative reasoning module*. The re-constructed reasoning chains are then modelled in the form of an argument by using the Toulmin model for the argument's structure. The collection of reconstructed, modelled reasoning chains is called the *recommendations space*.
    - definition and application of the integration scheme on the recommendations space with the help of the *argumentative reasoning module*. This is to identify and consider only those reasoning chains for the next step that adhere to the decision maker's required criteria. The reasoning chains that adhere to the decision maker's defined criteria are saved in the *valued recommendations set*.
    - a Web-based form that displays the results produced during the valuation of a reasoning chain.
  - Generation an integrated recommendations space which involves the following steps:
    - identification and resolution of conflicts between the arguments in the valued recommendations set with the help of the *argumentative reasoning module*.
    - construction of new arguments once the conflicts have been resolved. This involves combining the premises of the arguments that support the same claim. The new arguments are saved in the valued recommendations set.

* identification of unique conclusions/claims in the valued recommendations set followed by linking the supporting arguments to reach a conclusion/claim to form a reasoning chain. The construction of reasoning chains is carried out with the help of the *argumentative reasoning module*.

– Graphical representation of the integrated recommendations space to assist the decision maker in the decision-making process. Additionally, this will provide functional support to the decision maker to query the knowledge base.

(b) *Argumentative reasoning module* is exploited by the information and knowledge integration module for the valuation of the reasoning chains and the generation of the integrated recommendation space.

3. Semantic Web applications layer

This layer consists of Web@KIDSS which exploits the @IRRI layer and the information layer to achieve its objectives.

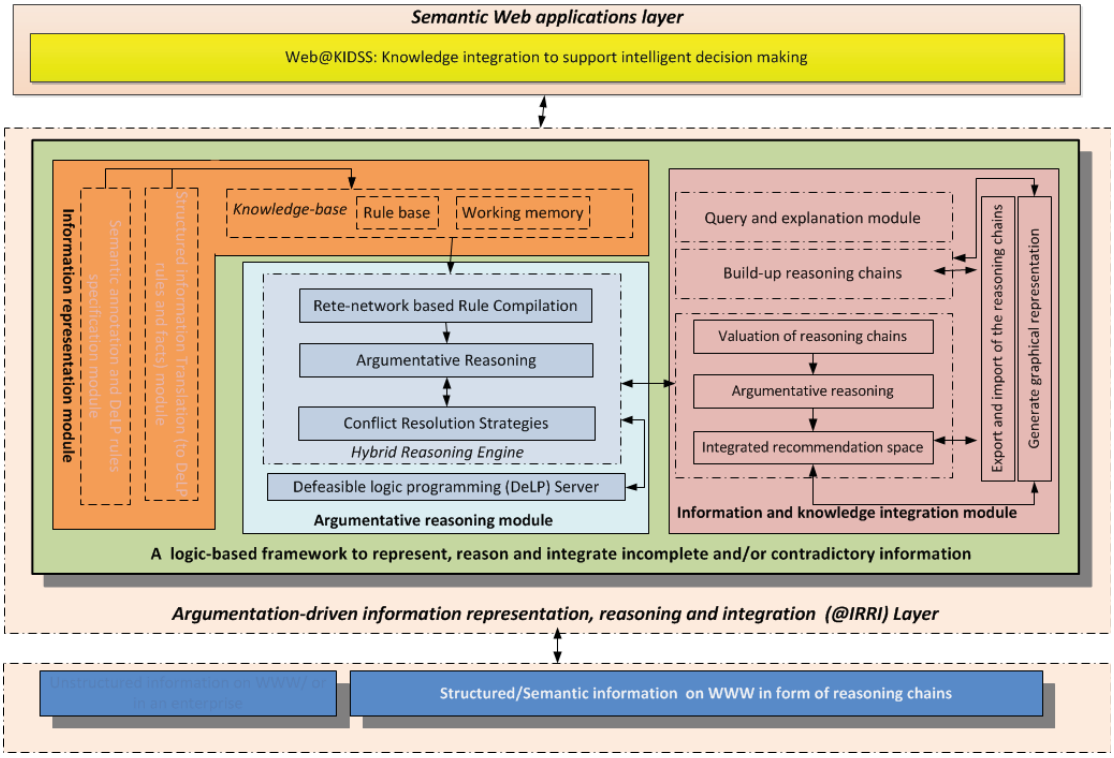


Figure 6.4: Proposed framework with highlighted components exploited by Web@KIDSS

Before explaining the working of the proposed framework, in the next sub-section, important definitions and concepts are introduced to assist the understanding of the working of Web@KIDSS.

6.3.1 Important definitions

In the following section, the formal *syntax and semantics* for Enterprise Knowledge Integration through the Argumentation-enabled Web-IDSS (Web@KIDSS) are defined.

6.3.1.1 AIF argument network

Rahwan et al. (2007b) define an argument network Φ as a graph G and set of forms F consisting of

- a set $\mathcal{N}$ of vertices (or nodes) comprising I-Nodes and S-Nodes; and
- a binary relation $\stackrel{edge}{\longrightarrow}$: $\mathcal{N}$ x $\mathcal{N}$ representing edges between nodes
- a binary relation using an inference scheme from set F

such that $\nexists\ (i,\ j) \in \stackrel{uses(RA-node,scheme)}{\longrightarrow}$ where both $i \in \mathcal{N}_1$ and $j \in \mathcal{N}_1$.

6.3.1.2 Argumentative production system as an argument network

Given an argument graph G and set of forms F in an argument network Φ , a Web@IDSS argument network AG is defined as follows: ($\mathcal{WM}$, $\mathcal{R}$, $Args$) where

- $\mathcal{WM}$: a set of information nodes i.e. $\mathcal{N}^I_{i,\ldots,n}$, where I represents the information nodes and i represents the index of the node.
- $\mathcal{R}$: a set of production rules or specifications to establish links between $\mathcal{N}^I_i$ nodes through S node such that $\nexists\ (i,\ j) \in \stackrel{edge}{\longrightarrow}$ where both $i \in \mathcal{N}_1$ and $j \in \mathcal{N}_1$
- $Args$: a set of arguments derived from $\mathcal{R}$, where each argument establishes a linked set of premises ($\mathcal{N}^I_i$) to a claim($\mathcal{N}^I_j$) through S node. Based on the forms of AIF ontology , the strict argument and defeasible argument are defined as follows:

  (Strict argument) : $\mathcal{N}^I_i$,.....,$\mathcal{N}^I_j \stackrel{Uses(RA,deductiveScheme)}{\longrightarrow} \mathcal{N}^I_k$

  (Defeasible argument): $\mathcal{N}^I_m$,.....,$\mathcal{N}^I_n \stackrel{Uses(RA,defeasibleScheme)}{\dashrightarrow} \mathcal{N}_o$

The binary relation $\xrightarrow{edge}$: $\mathcal{N}$ x $\mathcal{N}$representing edges between nodes in Web@IDSS can be categorized as follows:

- Counter-argument: $\mathcal{N}_i^I \overset{Uses(CA-Node)}{\dashrightarrow} \sim \mathcal{N}_j^I$ such that $\mathcal{N}_i^I$ is a counter-argument $\mathcal{N}_j^I$
- Static defeat: $\mathcal{N}_i^I \overset{Uses(PA-Node)}{\dashrightarrow} \mathcal{N}_j^I$ such that $\mathcal{N}_i^I$ is has priority over $\mathcal{N}_j^I$
- Dynamic defeat: $\mathcal{N}_i^I \overset{Uses(PA-Node)}{\dashrightarrow} \mathcal{N}_j^I$ such that $\mathcal{N}_i^I$ has priority over $\mathcal{N}_j^I$
- Sub-argument: To represent the sub-argument relationship in AIF format, a blank-node is added into the argument network i.e. $\mathcal{N}_i^I \overset{Uses(Blank-Node)}{\dashrightarrow} \mathcal{N}_j^I$ such that $\mathcal{N}_i^I$ (claim of an argument) is a sub-argument of $\mathcal{N}_j^I$ (premise of an argument).

6.3.1.3 Predecessor and Successor Nodes in the network

Given a graph AG consisting of a set of nodes $\mathcal{N}$ and a relation $\mathcal{S} \subseteq \mathcal{N} \times \mathcal{N}$defining the set of edges between the nodes, for each node n $\in \mathcal{N}$, the set of its predecessor and successor nodes is defined as follows:

- A predecessor node : $\{x \in \mathcal{N} \mid (x, n) \in \mathcal{S}\}$,
- A successor node : $\{x \in \mathcal{N} \mid (n, x) \in \mathcal{S}\}$.

6.3.1.4 Recommendations space

A collection of recommendations, each in the form of a reasoning chain $\lambda_{(identifer,result)}$ contributed by source 'i' is known as a recommendations space. Mathematically, a recommendations space is defined as follows:

$$\Theta = \sum_{i=0}^{n} \left\{ [i]\lambda_{(identifer,result)} \right\} \quad \text{..... Equation (6.1)}$$

6.3.1.5 Integration scheme

An integration scheme, the decision maker's defined argumentation scheme (Katie Atkinson, 2008), is a tuple with the following form:

$$\mathcal{IS} = –name, (premise_i, ...premise_n),\ conclusion,\ criticalquestions, variant\}$$

..... Equation (6.2)

where

- name is the label of the scheme which identifies the scheme;
- premise is a set of facts to be matched;
- sConclusion is the result of the scheme;
- sCriticalquestion is a set of queries;
- svaraint is a boolean flag for conflict blocking. If svariant is true, the conflicts are blocked and the reasoning chain will not be considered for any further processing; whereas if the flag is false, then reasoning chains with conflicts are still considered for further processing.

The critical questions can be categorized as exceptions and assumptions. The premises provide reasons for accepting the conclusion only if the assumptions are true and there are no exceptions. If either an assumption is false or an exception is true, unless the premises provide reasons for accepting the conclusion, the conclusion would not be valid (Katie Atkinson, 2008). Thus, both assumptions and exceptions attack the conclusion of the scheme.

6.3.1.6 Valuation operator and valued reasoning chain

The application of the integration scheme to a reasoning chain is termed 'valuation of a reasoning chain' and the resulting reasoning chain is called a valued reasoning chain. Mathematically, the valuation operator ◊ is defined as a binary operator as follows:

$$[\text{rc1}]\lambda^{val}_{(A,a)} = \left\{[rc1]\lambda_{(A,a)} \lozenge \mathcal{IS}\right\} \quad \text{.....Equation (6.3)}$$

During the valuation of a reasoning chain, all the premises and critical questions originating from the integration scheme are executed on the corresponding reasoning chain. If the premises match the reasoning chain and queries return true on the execution over the reasoning chain, the reasoning chain is considered to be a valued reasoning chain. The reasoning chain is still considered valued if the reasoning chain premise does not match or queries return false, but the conflict blocking flag, i.e. svariant is false.

6.3.1.7 Focus operator

$\otimes$ is a binary operator, such that

$$[\text{rc1}]\lambda^{val}_{(A,a)} \otimes [rc2]\lambda^{val}_{(B,a)} \quad \text{Equation (6.4)}$$

is called a 'focus operator'. This corresponds to AND operator. If two arguments belonging to different reasoning chains have the same claim, the application of the focus operator produces these arguments in a resulting set.

6.3.1.8 Merge operator

$\boxplus$ is a binary operator (Fan et al., 2010), such that

$$[\text{ar1}]\text{a,b,c} \dashrightarrow d \boxplus [ar2]\, e, b, c \dashrightarrow d \quad \text{Equation (6.5)}$$

is called a 'merge operator'. This corresponds to the OR operator and it applies to arguments that make the same claim. The application of this operator results in the construction of a new argument that carries all the unique premises of the arguments and links them to a common claim.

6.3.1.9 Unique operator

$\odot$ is a binary operator, such that

$$[\text{rc1}]\lambda^{val}_{(A,a)} \odot [rc2]\lambda^{val}_{(B,a)} \quad \text{Equation (6.6)}$$

is called a 'unique operator'. The application of the unique operator on reasoning chains and returns all those arguments whose claim is unique between the reasoning chains.

6.3.1.10 Conflict operator

is a binary operator, such that

$$[\text{rc1}]\lambda^{val}_{(A,a)} \oslash [rc2]\lambda^{val}_{(B,a)} \quad \text{Equation (6.7)}$$

is called a 'conflict operator'. The application of this operator to reasoning chains will return the set of arguments along with their counter-arguments and undefeated or blocking dialectical trees.

6.3.1.11 Preference operator

is a binary operator such that

$$[a]giveDiscount(XYZ) > [b] \sim giveDiscount(XYZ) \quad \text{Equation (6.8)}$$

is known as a 'preference operator'. The decision maker can define a preference relation explicitly for an argument and its counter-arguments.

6.3.1.12 Integrated recommendations space

The integration of recommendations, each in the form of a valued reasoning chain $\lambda^{val}{}_{(identifer,result)}$ contributed by a source 'i' is known as an integrated recommendations space. Mathematically, an integrated recommendations space is defined as follows:

$$\Theta_{integrated} = \sum_{i=0}^{n} \left\{ [i]\lambda^{val}_{(identifer,result)} \right\} \quad \text{..... Equation (6.9)}$$

6.3.1.13 Query

A query 'q' consists of a predicate, and can be executed on the argument set 'Args' with the help of function executeQuery(q) $\in \mathcal{F}$ to check the support for the predicate in the recommendations space.

6.3.2 Working of the proposed framework for Web@KIDSS

In this section, the working of the proposed framework for the integration of reasoning results produced by different Web@IDSS located both within the enterprise and/or in other enterprises after resolving the conflicts between them to assist the decision maker in the decision making process is discussed. Figure 6.5 presents a flow chart of the working of the proposed framework. The sequence of steps in the proposed framework is as follows:

1. Publication of EII in a standard format

   The Web@IDSS needs to publish decisions/results in a shareable format over the enterprise's intranet or the Internet so that they can be merged/considered by other Web-based DSS to assist the inter-enterprise or intra-enterprise decision-making process. To achieve this objective, Web@IDSS exploits the functionality of the *information and knowledge integration module* of the logic-based framework located at @IRRI layer. This module helps the Web@IDSS to transform the reasoning chain into a standard format i.e. AIF and publish it over the enterprise's intranet or the Internet. This process involves the following two steps:

   - Modeling of reasoning chains as an AIF argument network

     The elements of a reasoning chain i.e. arguments and the relationships between them, are modelled as an AIF argument network.

   - Semantic annotation and serialization of a reasoning chain

     The modeling of a reasoning chain as an AIF argument network is realized by annotating the elements of a reasoning chain with the concepts and relationships defined in the AIF core ontology [1]. The

[1] Details of the AIF core ontology are provided later in this chapter

semantic annotation is an automated process and once completed, the annotated reasoning chain is in OWL/RDF format and is published over the enterprise's intranet or the Internet.

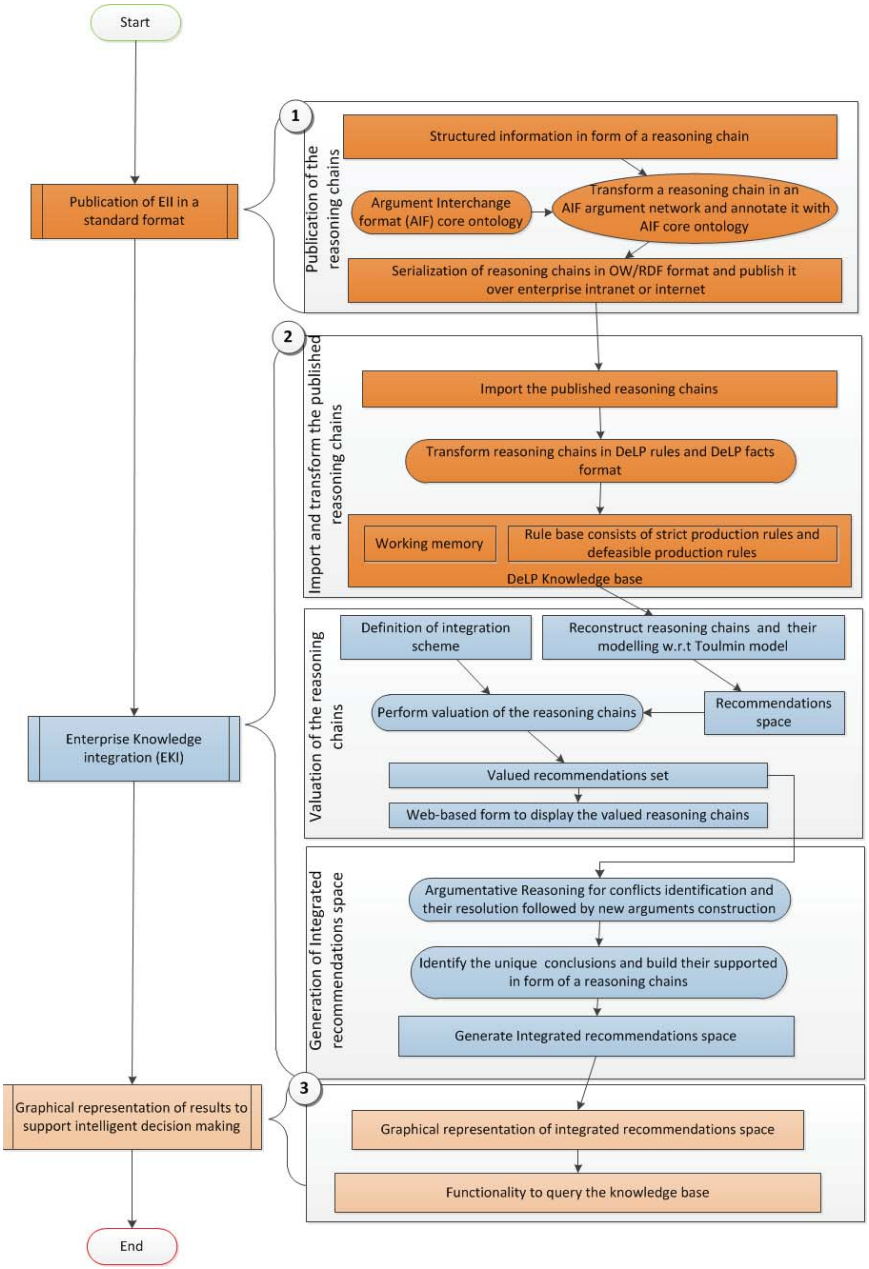


Figure 6.5: Flowchart illustrating steps performed by Web@KIDSS for enterprise knowledge integration

2. Enterprise knowledge integration

Once the reasoning chains have been published, Web@KIDSS needs to import and integrate them, generate a graphical representation of the integrated

reasoning chains to assist the decision maker in the decision making process of the inter-enterprise or intra-enterprise. To achieve this objective, Web@KIDSS exploits the functionalities of the *information and knowledge integration module* and the *argumentative reasoning module* of the logic-based framework located at @IRRI layer. This process involves the following three steps:

- Import and transform the published reasoning chains

  This steps involves importing the published reasoning chains followed by their transformation into DeLP format (i.e. DeLP rules and DeLP facts) and saves them in the knowledge base.

- Valuation of the reasoning chains

  This starts with the reconstruction of reasoning chains from the knowledge base and their modeling in the form of an argument using the Toulmin model for the argument's structure. The next step is to define and apply an integration scheme on the modelled reasoning chains. This is to identify and consider only those reasoning chains for the next step that adhere to the decision maker's required criteria. This process is called the 'valuation of the reasoning chains'. The reasoning chains that pass the valuation process are called *valued reasoning chains* and are saved in the *valued recommendations set*. The valuation of the reasoning chains involves the following steps:

  - Re-construct the reasoning chains. As discussed in Section 5.5.1, this involves the compilation of the production rules (DeLP rules) in the rule base as a Rete network and the DeLP facts in the working memory are passed through the Rete network. This results in the construction of arguments. The arguments are then linked in the form of reasoning chains. The reasoning chains are then modelled as an argument by using the Toulmin model for the argument's structure.

  - Define an integration scheme that specifies the decision-maker's criteria in the form of pre-requisites which the reasoning chain must satisfy in order for it to be considered for further processing.

  - Apply the integration scheme over the reasoning chains that resulted in the creation of the valued recommendations set.

  - Display the valued reasoning chains to the decision maker.

- Generation of the integrated recommendations space

  Once the valued recommendations set has been generated, the next step is to perform argumentative reasoning to identify and resolve conflicts between them, and identify the unique conclusions supported by the valued reasoning chains followed by their integration. Such integration of the valued reasoning chains results in the creation of the *integrated recommendations space*, involving the following steps:

  - Perform argumentative reasoning which involves the identification of conflicts between arguments belonging to different valued reasoning chains in a valued recommendations set. Once the conflicts have been identified, the automated resolution of conflicts between arguments takes place by computing either static or dynamic defeat. Once the conflicts between the arguments in the valued recommendations set have been resolved, the construction of new arguments takes place by combining the premises of those arguments that support the same claim.
  - Identify the unique conclusion supported by underlying valued reasoning chains.
  - Build the reasoning chains (as defined in Section 5.6.1), each of which support a unique conclusion. Such integration of information is called an integrated recommendations space.

3. Graphical representation of results to support intelligent decision making

   Once the integrated recommendations space has been created, Web@KIDSS exploits the functionality of the *information and knowledge integration module* of the logic-based framework located at @IRRI layer to provide a graphical representation of the integrated recommendations space and assist the decision maker to make the final decision. This process involves the following steps:

   - Graphical representation of the integrated recommendations space

     Once the integrated recommendations space has been created, Web@KIDSS provides the decision maker with a graphical representation to assist him in the intra-enterprise or inter-enterprise decision-making process. Such an integrated recommendation space represents the different viewpoints in the underlying information and the support for each.

- Functionality to query the knowledge base

  The Web@KIDSS provides an interface to query the knowledge base if the decision maker needs an explanation of the results returned by the system.

In the next sections, each of these steps will be discussed in detail.

6.4 Publication of enterprise integrated information (EII) in a standard format

As discussed in Chapter 5, a Web@IDSS represents, reasons and integrates potentially incomplete and\or contradictory information that exists both within the enterprise and/or in other enterprises to assist the decision maker in the decision-making process. For EKI, a reasoning chain produced by a Web@IDSS needs to be shared with other Web-based DSS which may be either within the enterprise and/or in other enterprises. The current representation of reasoning chains is Web@IDSS specific and it cannot be consumed directly by other Semantic Web applications. The proposed framework addresses this drawback with the help of the *information and knowledge integration module* of the logic-based framework located at @IRRI layer. This module helps the Web@IDSS transform and publish the reasoning chain in AIF format. Figure 6.6 describes the following steps involved in the transformation of a reasoning chain in AIF format and its publication over an enterprise's intranet or the Internet:

(a) Modeling of a reasoning chain as an AIF argument network

AIF is an effort to provide a standard representation of a set of arguments and the relationships between them to ensure the argument network is understandable by different applications. As discussed in Section 5.6.1, a reasoning chain is composed of a set of arguments and the relationships between them. In order to model the reasoning chain as an AIF argument network, an argumentative production system is defined as an AIF network of arguments in Section 6.3.1.2, where the elements of a reasoning chain are mapped to the elements of an AIF .

(b) Semantic annotation and serialization of a reasoning chain

Once the reasoning chain has been modelled as an AIF argument network, the next step is the realization (implementation) of a reasoning

chain as an AIF argument network. Using the mapping defined in Section 6.3.1.2, the reasoning chain is annotated with the ArgDF ontology that provides AIF reification [2] in OWL/RDF format. As a result of semantic annotation, the resulting reasoning chain is serialized in OWL/RDF format and published over the enterprise's intranet or the Internet.

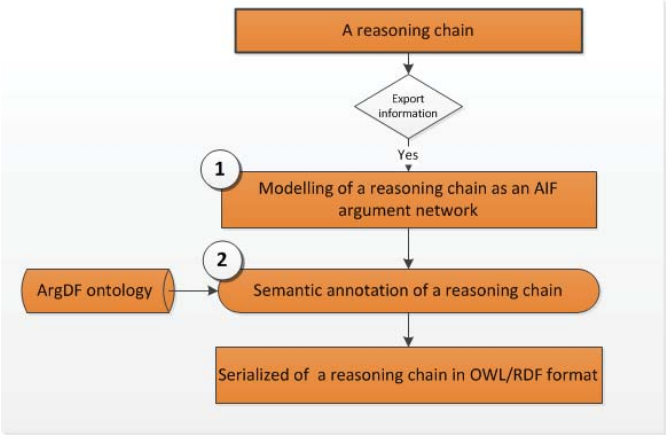


Figure 6.6: Flowchart illustrating steps performed by Web@KIDSS for publication of the reasoning chains

In the next-subsections, these two steps are discussed in detail.

6.4.1 Modeling of a reasoning chain as an AIF argument network

The Argument Interchange Format (AIF) is an international effort to develop a representational mechanism for exchanging argument resources between research groups, tools, and domains, using a semantically rich language (Chesnevar et al., 2006a; Iyad Rahwan, 2009; Rahwan et al., 2007b). AIF was developed as a commonly agreed upon core ontology i.e. AIF core ontology, that specifies the basic concepts used to express arguments and the relationship between arguments. The AIF core ontology, as depicted in Figure 6.7, is composed of the following two ontologies:

- Upper Ontology

[2] AIF reification refers to the use of a concrete language to represent an AIF argument network

The Upper Ontology defines the basic building blocks of AIF argument graphs, and the types of nodes and edges. There are two types of argument nodes:

- information nodes (I-Nodes) which capture information in the form of a premise, conclusion, exception or presumption, and
- scheme nodes (S-nodes) which provide the relationship between two I-Nodes and are further classified as:
  - rule application nodes (RA-Node) which correspond to inferences from premises to claims;
  - conflict nodes (CA-Node) which correspond to conflicts between two nodes;
  - preference application nodes (PA-node) which correspond to preference ordering between contradictory nodes.

- Forms Ontology

  The Forms Ontology allows for the conceptual definition of the elements of AIF graphs, such as premises, inference schemes and exceptions. Inference schemes are similar to the rules of inference in logic such as a deductive or defeasible inference.

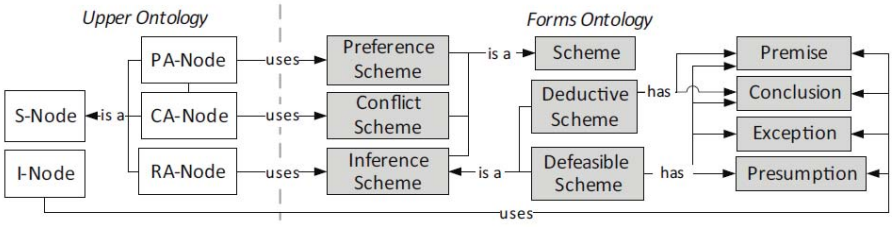


Figure 6.7: The Upper and Forms ontology of the AIF ontology(Bex et al. (2010))

Based on the definition of the AIF argument network (defined in Section 6.3.1.1), an argumentative production system is defined as a network argument (defined in Section 6.3.1.2) where the elements of a reasoning chain are mapped to the elements of the AIF argument network as follows:

- A strict argument consists of a set of premises and a conclusion. The premises and conclusion are linked with the help of a strict inference. During mapping, each premise and a conclusion is represented as an I-Node and the strict inference is represented as an S-Node using the deductive scheme.
  $\mathcal{N}_i^I$,.....,$\mathcal{N}_j^I \overset{Uses(RA,deductiveScheme)}{\longrightarrow} \mathcal{N}^I$

- A strict defeasible argument also consists of a set of premises and a conclusion. The premises and conclusion are linked with the help of a defeasible inference. During mapping, each premise and conclusion is represented as an I-Node and the defeasible inference is represented as an S-Node using the defeasible scheme.
  $\mathcal{N}_m^I$,.....,$\mathcal{N}_n^I \overset{Uses(RA,defeasibleScheme)}{\dashrightarrow} \mathcal{N}_o$

The binary relations between arguments in a reasoning chain are mapped to an AIF argument network as follows:

- The counter-argument relation involves two arguments which are in conflict with each other. The claims of the argument and its counter-argument are mapped as an I-Node and a CA-Node is used to represent the relationship between them.
  $\mathcal{N}_i^I \overset{Uses(CA-Node)}{\dashrightarrow} \sim \mathcal{N}_j^I$ such that $\mathcal{N}_i^I$ is a counter-argument $\mathcal{N}_j^I$

- Static defeat and dynamic defeat are two types of defeats that are used by an argumentative production system to resolve conflicts between an argument and its counter-argument that results in the establishment of preferences between them. During mapping, this relationship is represented as a PA-Node between the claims of arguments that are represented as an I-Node.
  $\mathcal{N}_i^I \overset{Uses(PA-Node)}{\dashrightarrow} \mathcal{N}_j^I$ such that $\mathcal{N}_i^I$ is has priority over $\mathcal{N}_j^I$

- For representation of the sub-argument relationship in AIF format, a blank-node is added into the argument network i.e. $\mathcal{N}_i^I \overset{Uses(Blank-Node)}{\dashrightarrow} \mathcal{N}_j^I$ such that $\mathcal{N}_i^I$ (claim of an argument) is a sub-argument of $\mathcal{N}_j^I$ (premise of an argument).

- The I-Node i.e. $\mathcal{N}_i^I$ with no successor and with predecessor nodes is called the 'result' of the reasoning chain. The remaining nodes are known as 'support' for the result.

To explain the modeling of a reasoning chain with an example, consider the case study discussed in Section 6.2 where the recommendation from the IT department about choosing the relocation service provider XYZ are shown in illustration 6.1.

$$IT=\left\{\begin{array}{l}[rc1.a.it.d1]client(it), happy(it,xyz), relocationService(xyz) \dashrightarrow reuseService(it,xyz)\\ [rc1.x.d2]client(it), relocationService(xyz), reuseService(it,xyz) \dashrightarrow giveDiscount(it);\\ [rc1.x.s1]giveDicount(it), advancmentPayment(it) \rightarrow normalDiscount(it)\\ [rc1.a.it.d3]ontimeDelivery(xyz) \dashrightarrow efficient(xyz).\\ [rc1.a.it.d4]not\ dmanageProduct(xyz) \dashrightarrow safeDelivery(xyz).\\ [rc1.x.d3]efficient(xyz), safeDelivery(xyz) \dashrightarrow reliableService(xyz)\\ [rc1.a.it.d5]largeTruck(xyz), reuseService(it,xyz), reliableService(xyz),\\ normalDiscount(xyz) \dashrightarrow goodRelocationService(xyz)\\ [rc1.a.it.d6]language(english), languageProblem(xyz,english) \dashrightarrow\sim clearCriteria(xyz)\\ [rc1.a.it.d7]demandCash(xyz), demandTip(xyz) \dashrightarrow\sim convienent(xyz)\\ [rc1.a.it.d8]goodRelocationService(xyz), not\ convienent(xyz), not\ clearCriteria(xyz)\\ \dashrightarrow recommendService(xyz)\end{array}\right\} illustration(6.1)$$

The Figure 6.8 provides the graphical representation of the recommendations in form a reasoning chain constructed from set of arguments shown in illustration 6.1 by using the approach proposed in Section 5.6.1.

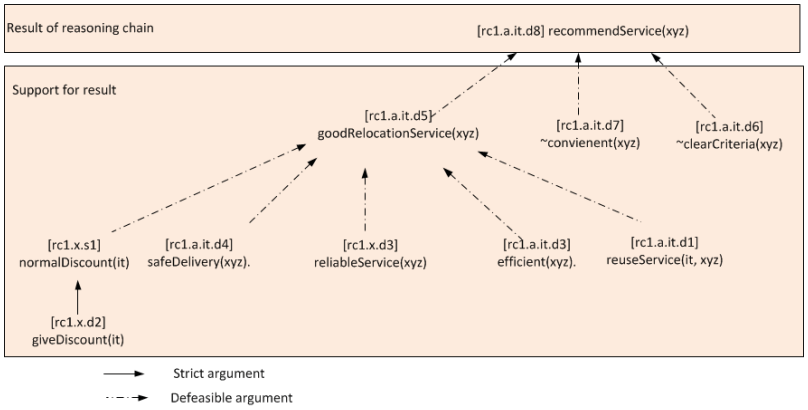


Figure 6.8: Pictorial representation of the recommendation forwarded by IT department

Figure 6.9 depicts the pictorial representation of a reasoning chain modeled as an AIF argument network where the premises and the conclusion are represented as I-Nodes and defeasible/strict inference is represented as RA-nodes that use defeasible/strict modus ponens to reach a conclusion. The directed arrows are simply to emulate the edge from an S-Node to a I-Node and bold-text I-Nodes are used represent the claim of an argument for better readability. Similarly, an I-Node such as $[rc1.a.it.d8]recommendService(xyz)$ that has no successor and has predecessor nodes represents the 'result' of a reasoning chain and the remaining nodes represent the 'support' for the result. The AIF argument does not provide a node to represent the

sub-argument relationship, therefore, to capture the sub-argument relationship, the sub-argument i.e. $[rc1.x.s1]normalDiscount(xyz)$ is linked to the argument i.e.$[rc1.a.it.d5]goodrelationService(xyz)$, with a blank node.

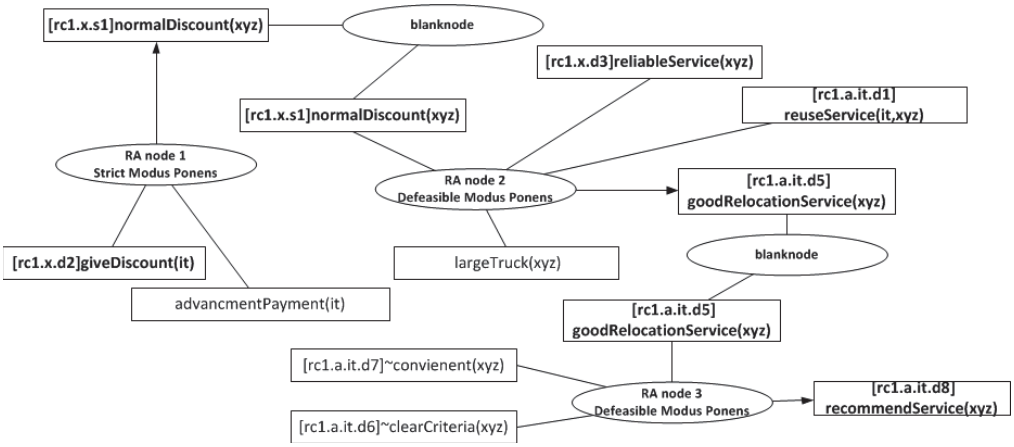


Figure 6.9: Pictorial representation a reasoning chain as an AIF argument network

6.4.2 Semantic annotation and serialization of a reasoning chain

Once the reasoning chain has been modelled as an AIF argument network, the next step is the realization (implementation) of a reasoning chain as an AIF argument network. To achieve this objective, the reasoning chain is annotated with an ArgDF ontology that provides AIF reification [3]. In Section 6.3.1.2, the mapping between the elements of a reasoning chain and the elements of an AIF argument network is defined. Using this mapping, the concepts and relationships defined in the ArgDF ontology are used to annotate the reasoning chain and the resulting reasoning chain is serialized in OWL/RDF format.

To explain the semantic annotation and serialization of a reasoning chain with an example, consider a reasoning chain that comes from the arguments shown in Figure 6.8. The semantic annotation is an automated process and the resulting reasoning chain is saved in OWL/RDF format. Figure 6.10 depicts the serialization of a reasoning chain built from the arguments shown in Figure 6.8.

[3]AIF reification refers to the use of a concrete language to represent an AIF argument network

```
@prefix RC: <http://www.abc.org/it/RC.owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl2xml: <http://www.w3.org/2006/12/owl2-xml#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix : <http://www.w3.org/2006/12/owl2-xml#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix kb: <http://protege.stanford.edu/kb#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://www.w3.org/2002/07/owl#> .
[ rdf:type owl:Ontology ;
  owl:imports
<file:/D:/data/Downloads/ArgDFProtegeOntology/ArgDFProtegeOnto
logy/ArgOnt.rdfs>
] .
###  http://www.abc.org/it/RC.owl#DefeasibleInference
RC:DefeasibleInference rdf:type owl:Class ;rdfs:subClassOf
kb:RuleScheme .
###  http://www.abc.org/it/RC.owl#recommendations
RC:recommendations rdf:type owl:Class ; rdfs:subClassOf
owl:Thing .
###  http://protege.stanford.edu/kb#hasPremise
kb:hasPremise rdf:type owl:NamedIndividual .
###  http://www.abc.org/it/RC.owl#RAnode1
RC:RAnode1 rdf:type kb:RA-Node ,  owl:NamedIndividual ;
           kb:hasPremiseDescription RC:advancePayment(it) ,
                                    RC:giveDiscount(it) ;
           kb:hasConclusionDescription RC:normalDiscount(it) ;
          kb:edgeFromSNode RC:normalDiscount(it) .
###  http://www.abc.org/it/RC.owl#advancePayment(it)
RC:advancePayment(it) rdf:type kb:I-Node , owl:NamedIndividual
;
                      kb:hasPremiseDescription kb:hasPremise .
###  http://www.abc.org/it/RC.owl#giveDiscount(it)
RC:giveDiscount(it) rdf:type kb:I-Node , owl:NamedIndividual ;
                    kb:edgeFromINode RC:RAnode1 .
###  http://www.abc.org/it/RC.owl#normalDiscount(it)
RC:normalDiscount(it) rdf:type kb:I-Node , owl:NamedIndividual
```

Figure 6.10: Serialization of AIF compliant reasoning chain in turtle format

6.5 Enterprise knowledge integration (EKI)

Once the reasoning chains have been published, the next step is EKI i.e. the integration of published reasoning chains and the provision of a graphical representation of integration information to assist the decision maker for the intra-enterprise or inter-enterprise decision-making process. To achieve this objective, Web@KIDSS exploits the functionalities of the *information and knowledge integration module* and the *argumentative reasoning module* of the logic-based framework located at @IRRI layer. This module helps the Web@KIDSS to import and transform the published reasoning chains, perform hybrid reasoning over them and integrate them in a format that can assist the decision maker in an enterprise-wide decision making process. Figure 6.11 presents the flowchart of enterprise knowledge integration. Enterprise knowledge integration involves the following three steps:

- Import and transform the published reasoning chains

During this step, the published reasoning chains in AIF format are imported by Web@KIDSS,transformed into DeLP format (i.e. DeLP rules and DelP facts) and saved in the knowledge base. During transformation of AIF argument network nodes to the elements of a reasoning chain, the transformation rules considered are as follows:

- I-Nodes are transformed to premises and the conclusion of an argument.
- RA-Nodes are used to determine the types of argument. The RA-Nodes that use strict modus ponens are realized as strict arguments and RA-Nodes that use defeasible modus poenens are realized as defeasible arguments.
- CA-nodes do not need any transformation because Web@KIDSS can identify the contradictory arguments using the argumentative reasoning module.
- PA nodes are transformed to the preferences relationship between contradictory arguments.

• Valuation of the reasoning chains

During this process, the following steps are performed:

- Re-construction of the reasoning chains. As discussed in Section 5.5.1, this involves compilation of the production rules (DeLP rules in the rule base) as a Rete network and facts (DeLP facts in the working memory) are passed through the Rete network. This results in the construction of arguments. The arguments are then linked in the form of reasoning chains. The reasoning chains are then modelled using the Toulmin model for an argument's structure.
- Define an integration scheme that specifies the decision maker's criteria in the form of pre-requisites for a reasoning chain to satisfy in order for it to be considered for further processing.
- Application of an integration scheme over the reasoning chains. The reasoning chains that pass the valuation are called *valued reasoning chains*. The collection of valued reasoning chains is called the *valued recommendations set*.

– Display the valued reasoning chains to the decision maker.

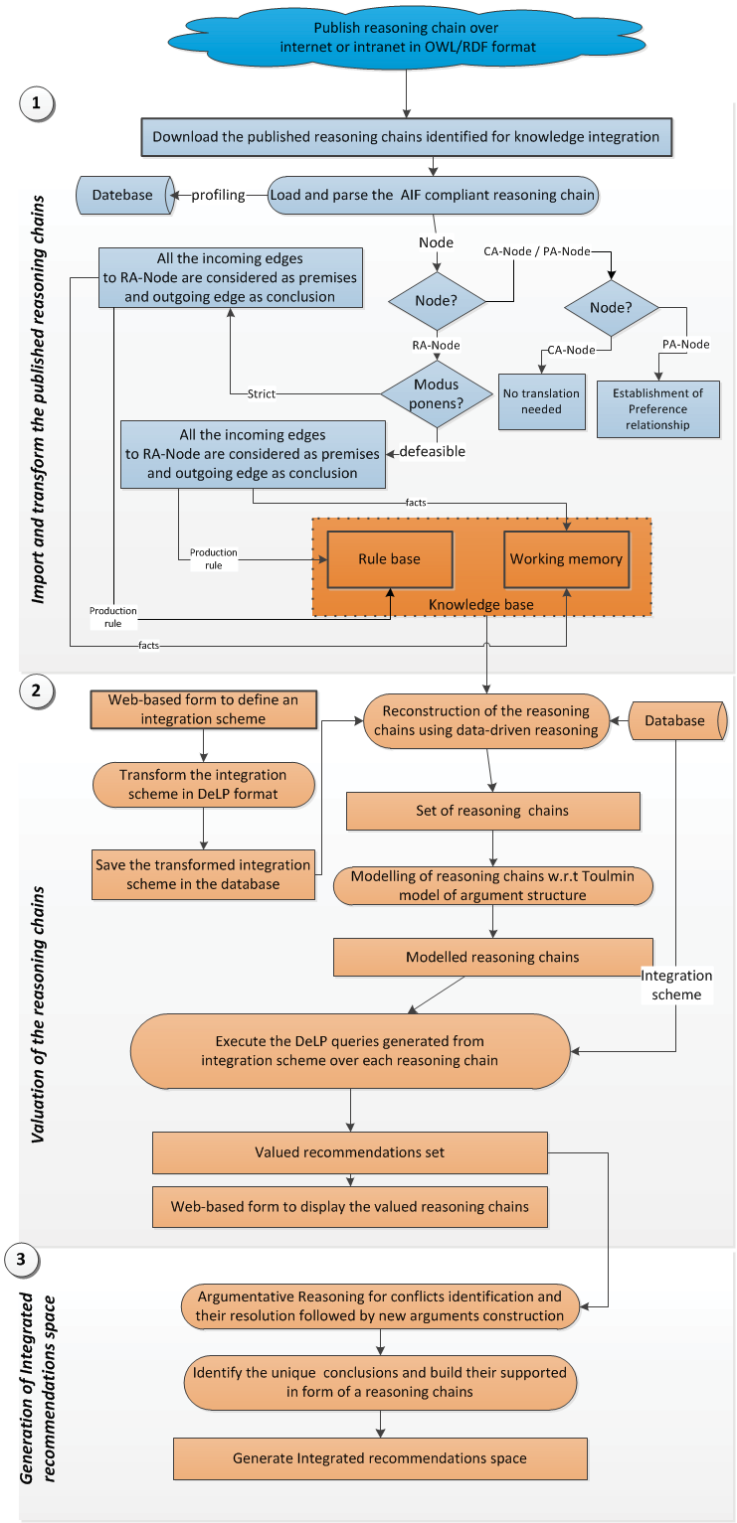


Figure 6.11: Flowchart illustrating steps performed by Web@KIDSS for knowledge integration

- Generation of integrated recommendations space

  Once the valuation of the reasoning chains is accomplished and the valued recommendations set is produced, the next step is to integrate the reasoning chains in the valued recommendations set to form an integrated recommendations space. To achieve this objective, the following steps are performed:

  - Perform argumentative reasoning which involves the identification of conflicts between arguments belonging to different valued reasoning chains in a valued recommendations set. Once conflicts have been identified, the automated resolution of conflicts between arguments occurs, with the help of computing either static or dynamic defeat. Once the conflicts have been resolved between the arguments in the valued recommendations set, the construction of new arguments takes place by combining the premises of these arguments that support the same claim.
  - Identify the unique conclusion supported by underlying valued reasoning chains.
  - Build integrated reasoning chains, each of which support a unique conclusion. Such integration of information is called the *integrated recommendation space*.

In the following sub-sections, I will explain each of these steps in detail.

6.5.1 Import and transform the published reasoning chains

The reasoning chains published on the enterprise's intranet or the Internet by different Web@IDSS in AIF format are imported by the Web@KIDSS. It understands and consumes these reasoning chains in AIF format (serialized in OWL/RDF format) and translates them to DeLP constructs as follows:

- The information nodes are translated as either the premise of an argument or claim, whereas the scheme nodes are used to build the types of arguments and the relationship between arguments. For example, if there is an RA-node (defeasible or strict inference), the predecessor of the scheme nodes will be the premise and the successor of the RA-node and will be the claim of the argument.

- Similarly, CA-nodes and PA-nodes are translated into counter-arguments and defeat the relationship between the arguments, respectively. The blank-nodes are translated as the sub-argument relationship between arguments.

To explain with an example, consider the AIF representation of reasoning chains shown in Figure 6.9. The steps involved in the translation of AIF elements of a reasoning chain to DeLP format by Web@KIDSS are as follows:

- Strict inference

  If the RA-Nodes use strict modus ponens (represented in Figure 6.12), all the incoming edges to the RA-Node are considered premises and the successor node is considered as the claim of a strict argument.

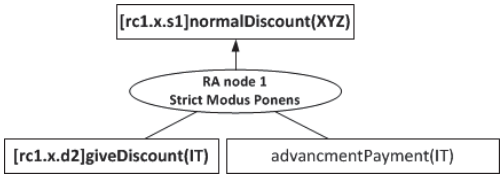


Figure 6.12: AIF representation of a strict argument

- Defeasible inference

  If the RA-Nodes use defeasible modus ponens (as represented in Figure 6.13), all the incoming edges to the RA-Node are considered premises and the successor node is considered the claim of the defeasible argument.

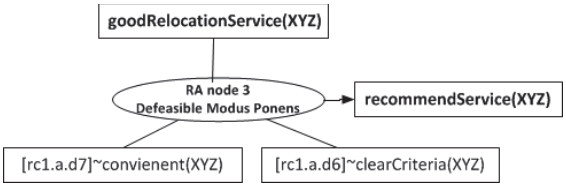


Figure 6.13: AIF representation of a defeasible argument

- CA-node

  No translation for the CA-node (as represented in Figure 6.14) as the proposed Web@KIDSS has a built-in mechanism to identify contradictory arguments.

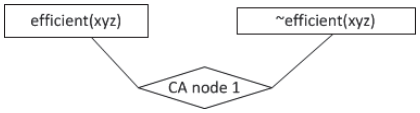


Figure 6.14: AIF representation of a CA-Node

- PA-node

  If the PA-nodes (as represented in Figure 6.15) exit between an argument and its counter-argument, then the argument having an incoming edges from the PA-Node has low priority than its counter-argument.

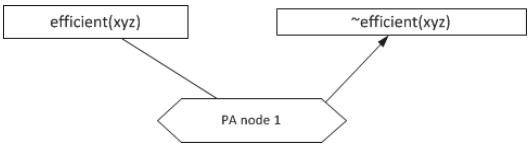


Figure 6.15: AIF representation of PA node

After translation of a reasoning chain, the arguments are transformed to production rules (DeLP rules) and saved in the knowledge base. During the process, new variables and DeLP facts are generated. Ground predicates such as $shopper(david)$, are transformed to a predicate with attribute variables such as $shopper(X)$. Such transformation of a reasoning chain allows the hybrid reasoning engine to save them in the knowledge base and perform hybrid reasoning over it. A similar procedure is performed for all of the premises of a production rule. Additionally, the premises of the argument (except those that represent incomplete information and start with 'not') are saved as DeLP facts in the working memory. Figure 6.16 represents the pictorial representation of the procedure that transforms an argument into production rules and saves the resulting DeLP facts and DeLP rules in the working memory and rule base, respectively.

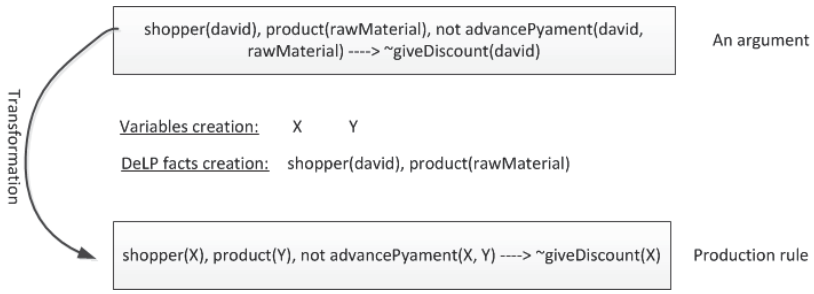


Figure 6.16: Pictorial representation of the transformation of an argument to a production rule

To explain the importation and transformation of reasoning chains, consider the case study discussed Section 6.2, where enterprise ABC considers the reasoning chains published by its departments. The collection of these reasoning chains is called recommendations space. The recommendation space for enterprise ABC is depicted in Figure 6.17 and can be mathematically represented as follows:

$$\Theta = \left\{ [rc1]\lambda_{(d7,recommend)}, [rc2]\lambda_{(d4,\sim recommend)}, [rc3]\lambda_{(d6,\sim recommend)} \right\} \quad \ldots \text{Equation (6.10)}$$

where

- $[rc1]\lambda_{(d7,recommend)}$ represents the recommendation from the IT department identified as 'rc1' in the form of a reasoning chain as shown in illustration 6.1.
- Similarly, $[rc2]\lambda_{(d6,\sim recommend)}$ is a recommendation from the Mark department identified as 'rc2' as shown in illustration 6.2, and $[rc3]\lambda_{(d4,\sim recommend)}$ is a recommendation from the HR department identified as 'rc3' as shown in illustration 6.3.

$$\text{Mark}=\left\{\begin{array}{l} [rc2.a.mk.d1] not\ happy(marketing, xyz), relocationService(xyz) \dashrightarrow \\ \sim reuseService(marketing, xyz) \\ [rc2.x.d1] relocationService(xyz), client(marketing), useService(xyz) \dashrightarrow \\ \sim giveDiscount(marketing) \\ [rc2.a.mk.d3] ontimeDelivery(xyz), largeTruck(xyz), \dashrightarrow efficient(xyz) \\ [rc2.a.mk.d5] dmanageProduct(xyz) \dashrightarrow \sim safeDeliver(xyz) \\ [rc2.a.mk.d4] not\ efficent(xyz), not\ reuseService(xyz), not\ giveDiscount(marketing), \\ not\ safeDeliver(xyz) \dashrightarrow \sim goodRelocationService(xyz) \\ [rc2.a.mk.d6] not\ goodRelocationService(xyz) \dashrightarrow \sim recommendService(xyz) \end{array}\right\} illustration(6.2)$$

$$\text{HR}=\left\{\begin{array}{l} [rc3.a.hr.d1] language(english), languageProblem(xyz, english) \dashrightarrow \sim clearCriteria(xyz) \\ [rc3.x.d2] client(hr), relocationService(xyz), reuseService(hr, xyz) \dashrightarrow giveDiscount(hr) \\ [rc3.a.hr.d2] not\ ontimeDelivery(xyz) \dashrightarrow \sim efficient(xyz) \\ [rc3.a.hr.d3] not\ efficient(xyz), not\ giveDiscount(xyz) \dashrightarrow \sim goodRelocationService(xyz). \\ [rc3.a.hr.d4] not\ goodRelocationService(xyz), not\ clearCriteria(xyz), \dashrightarrow \\ \sim recommendService(xyz) \end{array}\right\} illustration(6.3)$$

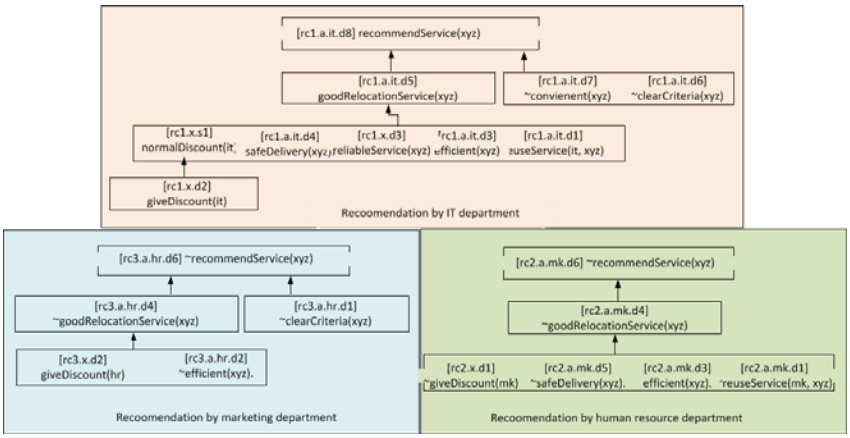


Figure 6.17: Pictorial representation of the recommendations space for an enterprise ABC

6.5.2 Valuation of the reasoning chains

The valuation of a reasoning chain consists of the following steps:

1. Modeling of reasoning chains w.r.t the Toulmin model of an argument's structure

   The Web@KIDSS first models the reasoning chain by identifying its basic elements as determined by Toulmin (2003). Such modeling of a reasoning chain helps to obtain a better understanding of the reasoning process and the importance of each element of a reasoning chain in the entire process. A reasoning chain is modelled as follows:

   - Back-up evidence: The initial working memory describes the current situation, from which the argumentative reasoner starts its derivation activity. In a reasoning chain, these nodes have no incoming edge (no predecessor nodes) and only an outer edge, or successor nodes, are considered back-up evidence.
   - Claim: The conclusion/result of the reasoning chain corresponds to the claim.
   - Warrant: The support for the result of a reasoning chain is called a 'warrant'. It is a set of arguments linked to form a reasoning chain link as back-up evidence for a claim.

   Such modeling of a reasoning chain has significant relevance for correctly modeling a practical argumentation activity and helps to categorize the various ways by which arguments can be analysed and defeated; therefore, the following strategies could have significant value as identified by Baroni et al. (1998).

- If conflict exists between a critical question and data, the entire conclusion drawn from them is undermined.
- It could help to point out flaws in the reasoning chain that relate data to the conclusion.
- If conflict exists between the claim and critical question, the decision maker has to see the warrant and data in order to defeat the claim.

To explain the modeling of a reasoning chain by following the Toulmin model, consider the case study discussed in Section 6.2 where the IT department forwards its recommendation to the business manager. Figure 6.18 provides the pictorial representation of a reasoning chain model using the Toulmin model for the argument structure.

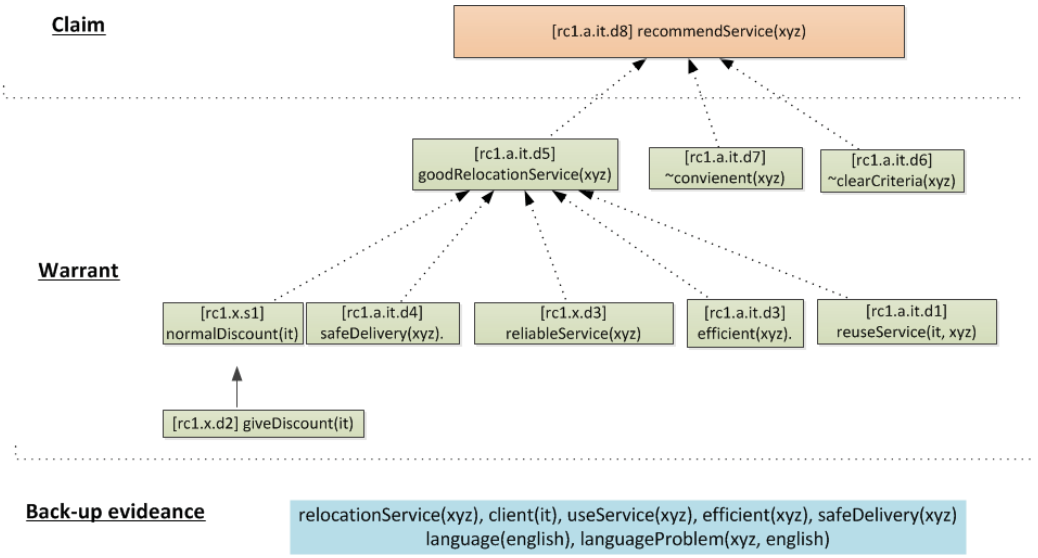


Figure 6.18: Pictorial representation of a modelled reasoning chain using Toulmin model

2. Definition and application of an integration scheme on the reasoning chains

Once the modeling of the reasoning chains is completed, the next step is to define and apply the integration scheme. The integration scheme, derived from the concept of the argumentation scheme, corresponds to our daily life pattern of reasoning. The application scope of parameters, defined in the integration scheme, ranges from the valuation of reasoning chains and their integration during the decision-making process. In the proposed framework, the DeLP language is used to create an integration scheme using the following steps:

- Name the integration scheme.

- Define the set of premises.
- Define the set of critical questions. The critical questions are queries to be executed on a reasoning chain. The critical questions are further categorised as follows:
  - Set of assumptions
  - Set of exceptions
- Set conflict handling variant i.e. conflict blocking either true or false. The scope of conflict handling can be defined at the valuation of reasoning chains or their integration or at both levels.
  - During the valuation of a reasoning chain, if any conflicts exist between a critical question and the premise, in the case of the conflict blocking variant being true, the reasoning chain is not considered suitable for knowledge integration and vice versa.
  - During enterprise knowledge integration, if a conflict exists between two arguments from different reasoning chains, in the case of the conflict blocking variant being true, these arguments are not considered in the final decision-making process and vice versa.

  Once the Integration scheme has been defined, the next step is to apply it on the reasoning chains. This process involves the generation of DeLP queries from the integration schemes and their execution on the selected reasoning chain. During this process, if any conflict exists either between the data and premise, or between the critical question and the warrant, the Web@KIDSS stores these results and depending upon the conflict blocking variable value, the reasoning chain will be considered for the next phase. The Web@KIDSS also displays the results to the decision maker.

Algorithm 6.1 provides the working of the valuation process over a valued recommendations set. It takes into account a set of AIF compliant reasoning chains

and sets their valuation flag to false.

```
   Data: Recommendation space {rc1,rc2,rc3}
   Result: Integrated arguments network
 1 Array rc []= {rc1, rc2, rc3}; int i=0;
 2 supplierIngtegrationScheme si;
 3 Decision maker initialize si;
 4 foreach rc.length do
 5 |   rc[i].valued=false;
 6 |   i++;
 7 end
 8 foreach rchain in rc do
 9 |   boolean a = Call to Valuation of reasoning chain(rchain,si);
10 |   if a is true then
11 |   |   rc.valued = true;
12 |   else
13 |   |   rc.remove(rchain);
14 |   end
15 end
```

Algorithm 6.1: Process of valuation over valued recommendations set

Then, it applies the integration scheme to each of the reasoning chains one-by-one by calling Algorithm 6.2 for the valuation of each reasoning chain. During the valuation of a reasoning chain, the decision maker's queries defined in the form an integration scheme are executed on the reasoning chain. If the execution of the query on the reasoning chain returns false (i.e. the reasoning chain does not adhere to the decision-maker's criteria), this is captured as a conflict between the reasoning chain and the integration scheme. Once the valuation of the reasoning chain is completed, if a conflict exists between the reasoning chain and the integration scheme and the variant flag is set to false (i.e. the reasoning chains is not considered for further processing), the reasoning chain is removed from the valued recommendations set.

To explain the integration scheme with help of an example, consider the case study discussed in Section 6.2 where the business manager has a set of recommendations in the form of reasoning chains and he wants to consider only those reasoning chains for enterprise knowledge integration which satisfy certain specific criteria. For example, the business manager specifies the criteria that the recommendation must be for relocation service provider XYZ. Therefore, only recommendations for XYZ are considered for the final decision-making process, as shown in Figure 6.19 .

```
   Data: a reasoning chains(rc) and integration scheme(is)
   Result: boolean
 1 foreach premise in is do
 2 |   if result = execute(premise) on rc then
 3 |   |   if result== false then
 4 |   |   |   addconflict(premise, rc);
 5 |   |   end
 6 |   end
 7 end
 8 foreach cq in is do
 9 |   if result = execute(cq) on is then
10 |   |   if result== false then
11 |   |   |   addconflict(cq, rc);
12 |   |   end
13 |   end
14 end
15 if addconflict[] !=null or is.conflictBlocking==true then
16 |   return true;
17 else
18 |   return false;
19 end
```

Algorithm 6.2: Valuation of a reasoning chain

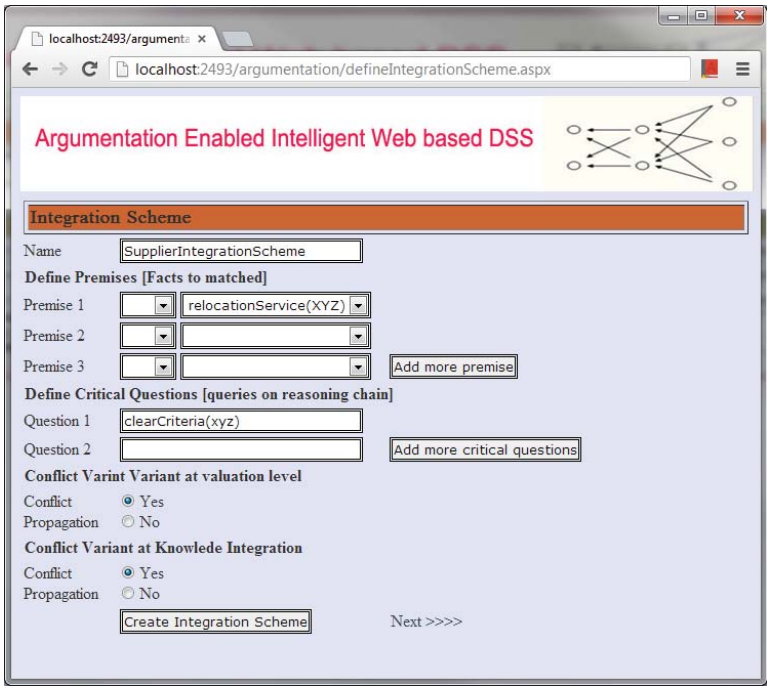


Figure 6.19: Web-based from of Web@KIDSS to define integration scheme

To achieve this objective, they define the features in the integration scheme as follows:

- Premise

  The business manager wants to define a criteria that the recommendation is against relocation service provider XYZ. This is done by defining an execute function that takes a predicate as input as follows: $executeQuery(relocationService(xyz))$. The premise is a query over the knowledge base that contains the reasoning chain from the IT department. If the query returns yes, then this demonstrates that the backup evidence contains information that the relocation service provider is XYZ.

- Critical questions

  The business manager wants to know whether or not in the underlying reasoning chain XYZ is good at formalising the clients' criteria and how it is supported by the reasoning chain (i.e. warrant). This is accomplished by defining an execute function that takes a predicate as input such as: $executeQuery(relocationService(xyz))$.

- Variant

  If the business manager wants to consider only those reasoning chains which passed the tests defined in the integration scheme and qualify for EKI, he sets the boolean variable to true. Otherwise, he can set it to false in order to include the reasoning chain even if it does not qualify for EKI. Figure 6.20 shows a Web-based form depicting the valued reasoning chain initially forwarded by the IT department of enterprise ABC.

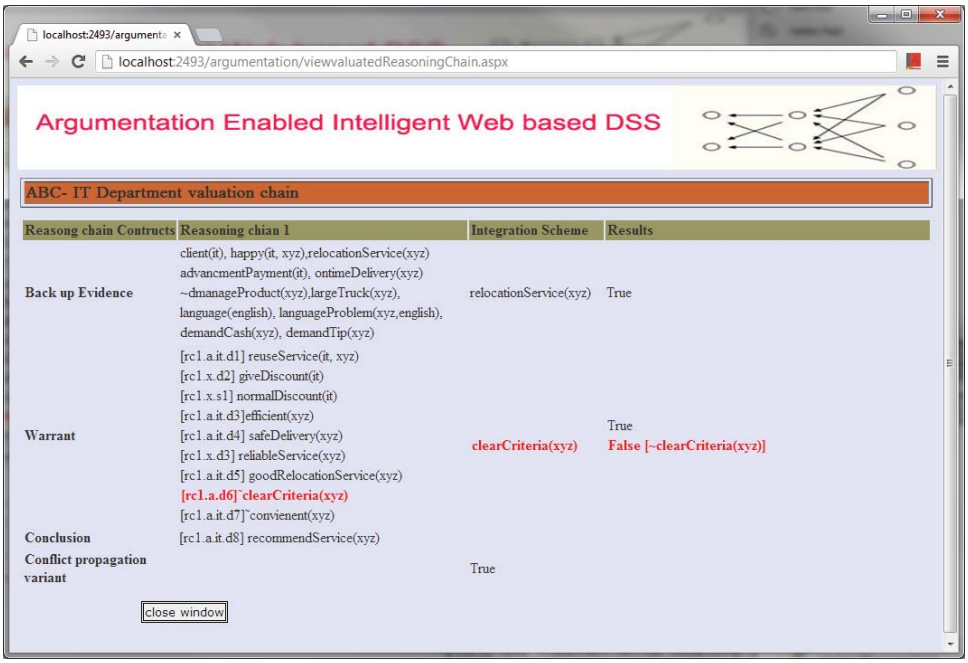


Figure 6.20: Web-based form of Web@KIDSS that shows the valuation of a reasoning chain

6.5.3 Generation of integrated recommendations space

Once the valued recommendation set has been generated, the next step is to perform argumentative reasoning to identify and resolve conflicts between them, identify the unique conclusions supported by the valued reasoning chains followed by their integration. Such integration of valued reasoning chains is called *integrated recommendations space* and involves the following steps:

1. Argumentative reasoning between reasoning chains

   During this process the following steps are performed:

   - The identification of conflicts between arguments belonging to different valued reasoning chains in a valued recommendations set. The conflict operator ($\oslash$) is a binary operator defined in Section 6.3.1.10, which when applied on the valued reasoning chains (e.g. rc[i] $\oslash$ rc[j]) returns a set of arguments that are in conflict.

   - Once the contradictory arguments have been identified, the Generalize Specificity conflict resolution strategy is used by computing either static or dynamic defeat to resolve conflicts between arguments. In the case of blocking arguments (where the dialectical trees of both the arguments and

their counter-arguments are undefeated), then Web@KIDSS needs human intervention to resolve the conflict between them. Further discussion on this is given in Chapter 7.

- After conflict resolution, the construction of new arguments is started. If two arguments from a valued recommendations set have the same claim, the premises of these arguments are combined to produce a new argument. The focus operator ($\otimes$) is a binary operator defined in Section 6.3.1.7, which when applied to valued reasoning chains (e.g. e.g. rc[i] $\otimes$ rc[j]), returns the set of arguments that share the same claim. The merge operator ($\boxplus$) is a binary operator defined in Section 6.3.1.8, and when applied to a set of arguments, results in the construction of a new argument that replaces the old arguments which support the same claim.

2. Identification of the unique conclusion supported by underlying reasoning chains

   Once the argumentative reasoning over the valued recommendations set is completed, the next step is to identify the unique conclusions from it. The unique operator ($\odot$) is a binary operator defined in Section 6.3.1.9, which when applied to valued reasoning chains (e.g. rc[i] $\odot$ rc[j]) returns the set of arguments that support the unique claim.

3. Building integrated reasoning chains

   Once the unique conclusions have been identified, the last step is to build the reasoning chains. The methodology proposed for the construction of reasoning chains in Section 5.6.1 is used for the construction of the integrated recommendation space.

Algorithm 6.3 shows the process of generating the integrated recommendation space. It first loops through a set of reasoning chains and compares the results of a reasoning chain with the results of the remaining reasoning chains; if the results match, then these reasoning chains are integrated. Four kinds of operators are used during this integration process. With the help of focus ($\otimes$) and merge ($\boxplus$) operators , the new arguments are constructed and then loaded into a valued recommendation set. With the help of a unique operator ($\odot$), unique arguments from both reasoning chains are loaded into an valued recommendation set. With the help of the conflict operator ($\oslash$), contradictory arguments are taken into account for conflict resolution. If the conflict blocking flag for knowledge integration is false, then Web@KIDSS tries to resolve conflicts with the help of static or dynamic defeat. Otherwise, Web@KIDSS provides an interface for the decision maker to establish the preference between the

contradictory arguments. Finally, it invokes Algorithm 5.4 (defined in Chapter 5) in order to build a reasoning chain from the arguments in the valued recommendations set. The important thing to note here is that conflicts may exist in a valued recommendation set if the conflict blocking flag is true.

```
   Data: Valued recommendation set, claimflag
   Result: Array of combined reasoning chain
 1 Array commonClaims [ ];
 2 Array contradictoryArguments [ ];
 3 for i=0;i <rc.length(); i++ do
 4     for j=i;j <rc.length(); j++ do
 5         if flag==sameClaims then
 6             condition=rc[i].result ==rc[j].result
 7         else
 8             condition=rc[i].result !=rc[j].result
 9         end
10         if condition==true then
11             commonClaims = rc[i] ⊗ rc[j];
12             argumentsSet= rc[i] ⊙ rc[j];
13             contradictoryArguments = rc[i] ⊘ rc[j];
14             foreach c in commonClaims do
15                 argRc1 = getArgument(c, rc1);
16                 argRc2 = getArguments(c, rc2);
17                 r = argRc1 ⊞ argRc2;
18                 argumentsSet=r;
19                 i++;
20             end
21             foreach arg in contradictoryArguments do
22                 argRc1 = getArgument(arg, rc1);
23                 argRc2 = getArguments(arg, rc2);
24                 if conflictblocking==true then
25                     argumentsSet = argRc1;
26                     agumentsSet = argRc2;
27                     if static or dynamic defeat does not exists between argRc1, argRc2 then
28                         userPreference=getUserPreference(argRc1,argRc2);
29                     end
30                     if userPreference(argRc1,argRc2) then
31                     preferenceSet = argRc1 > argRc2;
32                 end
33             end
34         end
35         rc [i]=BuildupReasoningChain(argumentsSet);
36     end
37 end
38 return rc;
```

Algorithm 6.3: Integrated recommendation space

To explain enterprise knowledge integration with an example, consider the case study discussed in Section 6.2 where each department needs to formulate and forward its recommendations about relocation service provider XYZ to the business manager. During this process, each department, with the help of Web@IDSS , produces recommendations in the form of a reasoning chain. Consider

the recommendation forwarded by the manager of the IT department shown in Figure 6.6 where he recommends that although the relocation service provider is not convenient and is not good at formalising criteria, he assumes it to be a good relocation service provider and recommends it for relocation purposes (i.e. $[rc1.a.it.d8]goodRelocationService(xyz), not\ convienent(xyz), not\ clearCriteria(xyz)$ $\dashrightarrow recommendService(xyz)$). However, other departments have a different opinion as shown in Figure 6.17. It is important to note here that the recommendations produced by each department contain valuable information about relocation service supplier XYZ which could help the business manager make the final decision i.e. whether or not to select relocation service provider XYZ for the relocation of the enterprise. Figure 6.21 shows the pictorial representation of the integrated recommendation space. The double-circled arguments are newly constructed arguments during argumentative reasoning for enterprise knowledge integration.

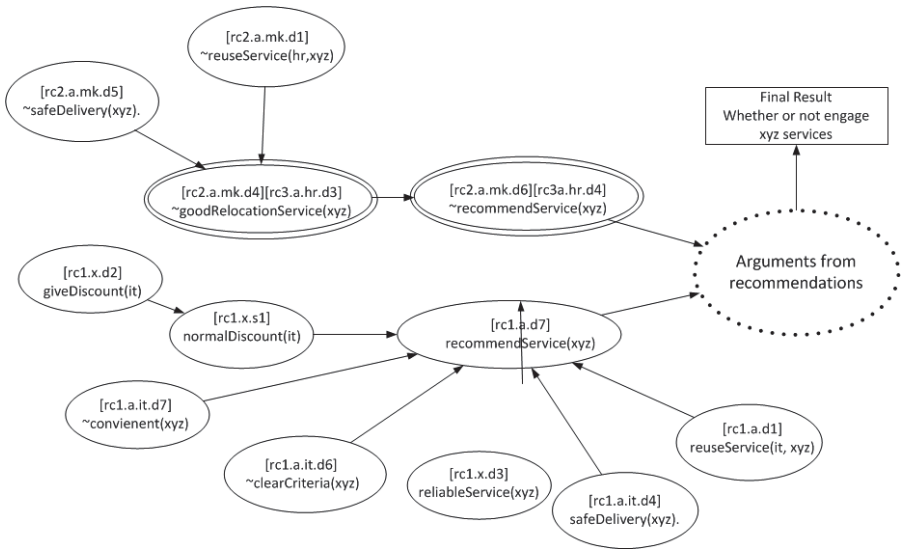


Figure 6.21: Pictorial representation of integrated recommendations space

6.6 Graphical representation of results to support intelligent decision making

The last functionality performed by the *information and knowledge integration module* of the logic-based framework is the graphical representation of the integrated recommendations space and to provide query support to answer the questions of the decision maker and assist them in the decision-making process. This process involves the following steps:

1. Graphical representation of the integrated recommendation space

   Once the generation of the integrated recommendation is completed, the next step is its graphical representation for the decision maker in order to assist him in the decision-making process. To explain with the help of an example, consider Figure 6.22 which represents the graphical representation of the integrated recommendations space depicted in Figure 6.21. The important features of graphical representation of a reasoning chain are as follows:

   - The reasoning chain is represented as a tree. Each reasoning chain supports a unique conclusion.
   - An argument is represented in short form e.g. $[s1]normalDiscount(david)$ where [s1] is the label of the argument and $normalDiscount(david)$ is the claim of the argument.
   - The *arguments* are depicted with an rectangle shape, defeasible inference is depicted with a dotted arrow and strict inference with a straight arrow.

   Such graphical representation helps the business manager of enterprise ABC to understand the whole reasoning process which can result in two recommendations: either recommend XYZ or not. He can identify the reasons for the recommendations as follows:

   (a) Recommend Service provider XYZ

   The manager of the IT department recommends service provider XYZ for the relocation of enterprise ABC. His recommendation is based on the following information:

   - XYZ considers an enterprise ABC eligible for a discount. In light of the current available information for decision making, he will offer a normal discount to enterprise ABC.
   - Although XYZ may be inconvenient and not able to capture the enterprise's criteria, the supplier is reliable and will likely provide safe delivery of the enterprise's goods.
   - XYZ has been used previously by the IT department and the manager is happy with their service and wants to reuse them for the relocation of the department.

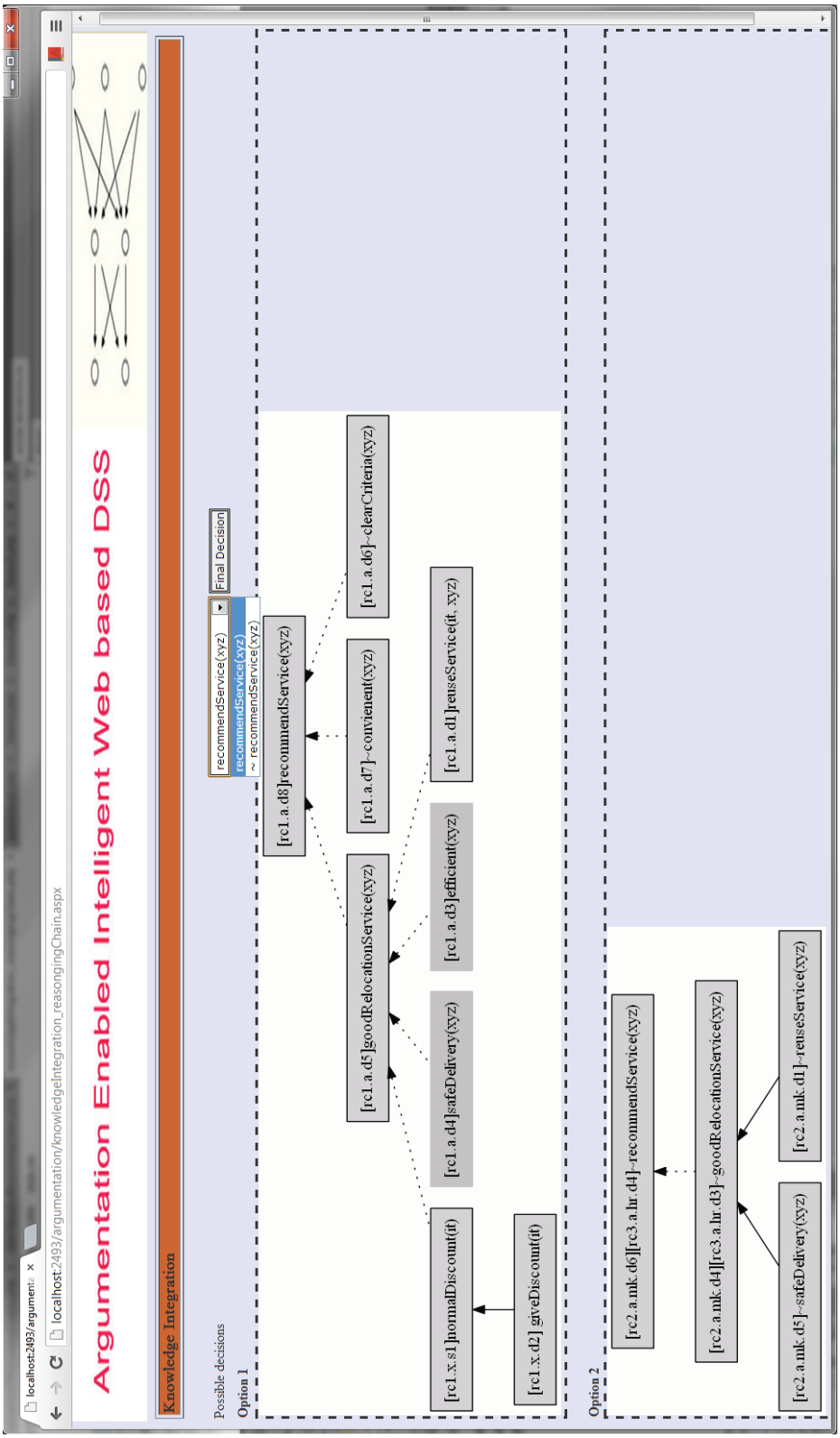


Figure 6.22: Web-based form of Web@KIDSS presenting integrated knowledge to assist the decision maker in decision making process

(b) Not recommend service provider XYZ

The managers of the HR and Marketing departments do not recommend XYZ for the relocation of the departments of enterprise ABC. Their recommendations are based on the following information:

- XYZ has been used for relocation services before and the marketing department was not happy with their service.
- XYZ may not provide safe delivery.
- Both departments consider XYZ to be a bad relocation service provider.

2. Query the knowledge base

Once the integrated recommendation space has been generated and displayed to the decision maker in a graphical format, he may query the knowledge base to obtain an explanation of the results. In Section 6.3.1.13, the definition of a query is provided. The execution of a query on the knowledge base may result in one of the following conclusions:

- If the answer is 'yes',, the result will be an undefeated dialectical tree. Mathematically, it is represented as follows:

$$\Sigma_U(\mathcal{A}, h) = executeQuery(q) \quad \text{Equation (24)}$$

- If the answer is 'no',, the result will be a defeated dialectical tree. Mathematically, it is represented as follows:

$$\Sigma_D(\mathcal{A}, h) = executeQuery(q) \quad \text{Equation (25)}$$

- If the answer is 'undecided',, the result will be a blocked dialectical tree. Mathematically, it is represented as follows:

$$\Sigma_B(\mathcal{A}, h) = executeQuery(q) \quad \text{Equation (26)}$$

- Unknown, if the predicate in the query is not in the language of the program. Mathematically, it is presented as follows:

$unknown = executeQuery(q)$ ..... Equation (27)

To explain the query on the knowledge base with help of an example, consider that the decision maker wants to know whether XYZ is a good relocation service provider. To accomplish his objective, the query $goodRelocationService(xyz)$ is executed on the knowledge base and results in a defeated dialectical tree. This is because the argument that states that XYZ is a good relocation service provider has been defeated by the set of arguments that state that XYZ is not a good relocation service provider. The decision maker uses this representation which considers all the recommendations from the different stakeholders and resolves the conflicts between them to assist him in taking an informed decision.

6.7 Conclusion

In this chapter, a solution for EKI was presented in order to assist the decision maker in enterprise-wide decision making. It was pointed out that the Web@IDSS (discussed in Chapter 5) addressed the issues of information integration to assist in the decision-making process, but does not address the issue of sharing and integrating information for the intra-enterprise or inter-enterprise decision-making process. To overcome this problem, the Web@IDSS was extended to make its results shareable in AIF format. Additionally, a framework for argumentation-enabled Web-based IDSS for enterprise knowledge integration (Web@KIDSS) was proposed. The Web@KIDSS import transforms standard reasoning chains to DeLP format, evaluates them against the decision maker's defined criteria defined as an integration scheme followed by their integration using argumentative reasoning. The major contributions of this chapter are as follows:

1. The extension of Web@IDSS to share its reasoning results in a standard AIF format.

2. The formalization of syntax and semantics for enterprise knowledge integration in an enterprise .

3. The proposal of a framework for representing, reasoning and integrating potentially incomplete and/or contradictory reasoning chains to support the intra-enterprise or inter-enterprise decision-making process.

4. The graphical representation of the integrated results and the provision of query support for decision makers.

Chapter 7 - Process Map Discovery from Business Policies: A Knowledge Representation approach with Argumentative Reasoning (KR@PMD)

7.1 Introduction

In Chapter 5, an Argumentation-enabled Web-based IDSS (Web@IDSS) was proposed to help decision makers consider the *structured information*, which exists within the enterprise and/or in other enterprises, to represent, reason over it, resolve conflicts between this information and the existing enterprise information using the *Generalize specificity-based conflict resolution strategy* and integrate this to assist in the decision-making process. In this chapter, the functionality of Web@IDSS is extended to take into account the *unstructured information* which exists within the enterprise and/or in other enterprises, to represent and reason over it, to enable this information to be considered in decision making and provide solutions for Enterprise Information Integration (discussed in Chapter 5) and Enterprise Knowledge Integration (discussed in Chapter 6).

As mentioned in Section 4.6.3, the process of considering unstructured information using the proposed logic-based framework by taking into account the business polices of an enterprise or two or more collaborating enterprises was explained. Operational business processes that are derived from business policies consist of business processes and business rules that define how an enterprise carries out its operations. It may be possible that the business policies of such enterprises may be incomplete and/or contradictory, leading to inconsistencies between them and operational business

processes. This needs to be resolved in order to ensure either successful collaboration of enterprises or that their working environment is in accordance with legal regulations or government policies. To overcome this problem, there is need to generate a business process map from unstructured business policies that can be used for the validation of operational business processes or the realization of new business processes at an operational level in order to ensure that their working environment is in accordance with legal regulations or government policies.

To address this problem, in this chapter, a **K**nowledge **R**epresentation and argumentative reasoning-based approach for business **P**rocess **M**ap **D**iscovery (KR@PMD) is proposed. The proposed framework represents and reasons over unstructured information (i.e. business policies of an enterprise or collaborating enterprises), providing different *argumentation-driven conflict resolution strategies* to identify and resolve conflicts, followed by the integration and graphical representation of the information in a format that may assist the decision maker in the intra-enterprise or inter-enterprise decision-making process. This will advance research in Web@IDSS as depicted in Figure 7.1

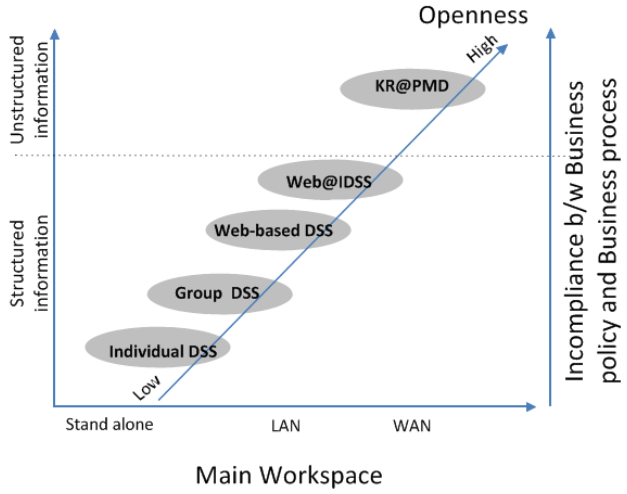


Figure 7.1: Evolution towards Web-based IDSS that can discovers process map from unstructured business policies

The remainder of this chapter is organized as follows: Section 7.2 provides an introduction to the context where unstructured information, represented as business policies, brings challenges for its representation, reasoning and integration for intra-enterprise (EII) or inter-enterprise decision making (EKI). Section 7.3 outlines a real-life case study to explain the problem. In Section 7.4, an overview of the proposed framework for process map discovery from business policies is provided. From Sections 7.5 to 7.7, each component of the proposed framework is explained in detail and how it

provides a solution to the problem highlighted in the case study is discussed. Section 7.8 concludes the chapter.

7.2 Unstructured business policies and challenges for the enterprises

Business policies play a pivotal role in an enterprise and are defined as high level directives that control, guide, and define constraints and procedures, thus shaping how an enterprise determines its course of action (Markovic et al., 2009). Using business policies, business processes and rules are derived according to how an enterprise carries out its operations, as shown in Figure 7.2. Two important factors that are essential at this stage to ensure the successful completion of business activities are:

1. that the derived operational business processes consistently represent the business policies, and

2. that there is no ambiguity or contradictory information in the business policies that may result in conflict in the derived operational business processes. This is particularly important when the information comes from multiple sources either within the enterprise and/or from different enterprises.

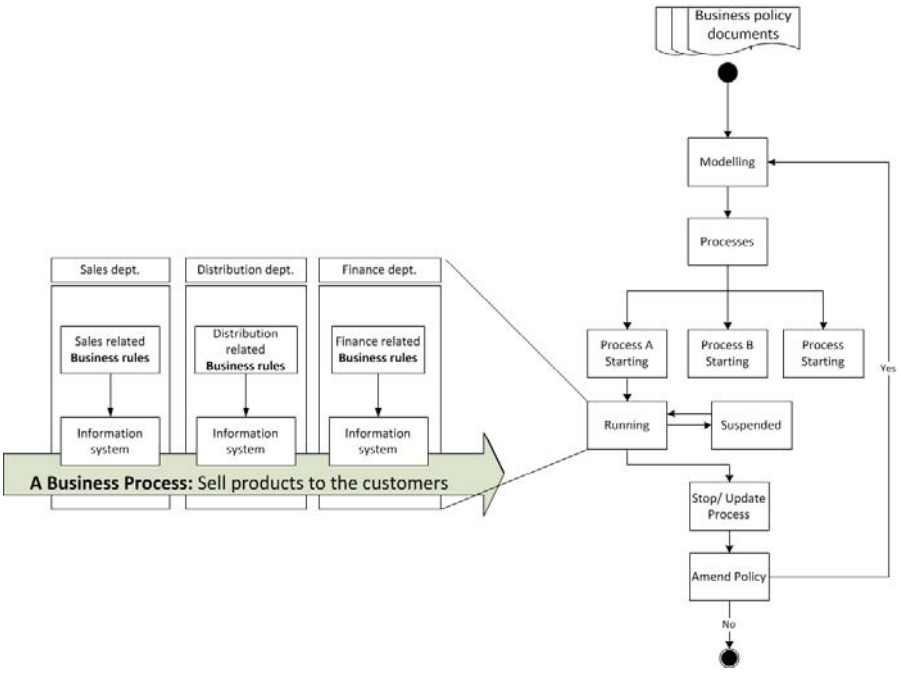


Figure 7.2: Business policy life cycle in an enterprise

The research literature shows that derived operational business processes often do not comply with enterprise business policies (Wang et al., 2009; Aalst, 2009), and as a result, enterprises invest huge amounts of money to ensure their business policies comply with the requirements of various regulating authorities (Sadiq and Governatori, 2009; Turetken et al., 2011; Meyer et al., 2011; Rajsiri et al., 2010). The mismatch between business policies and operational business processes may be for reasons related to both business process modeling and the implementation of business process phases. For example:

- During process modeling from the business policy phase, the current process modeling languages, i.e. Business process modeling notation (BPMN) and Event-Driven process chains (EPCs), lack proper semantics which often leads to debates on how to interpret business process models (Lohmann et al., 2009). Such situations may result in misunderstanding an enterprise's business processes in relation to its different functions and cause potential internal control deficiencies.
- During the implementation of the process model phase, business processes in modern enterprises often operate in a dynamic environment in which business processes are modified on a continuous basis to achieve business goals. This situation may result in some deviation or exception from the ideal business process execution defined in the business policies. Current process modeling languages are not able to capture the exceptions in business processes that appear at the operational level (Klein and Dellarocas, 2000). Additionally, research has shown that in most enterprises, the link between the last phase of process modeling shown in Figure 7.2 i.e. Amend Policy, and the first phase, i.e. Modeling, is not pursued actively (Wang et al., 2009; Aalst, 2009; Liu and Ong, 1999). As a result, the exceptions or updates in business processes are not reflected in business policies.

To overcome this problem, process mapping techniques (Madison, 2005) have been used in the literature. Process mapping techniques provide a visual representation of the business process and assist in creating a common understanding of business processes. They also assist in the identification of visible issues and provide an opportunity for business process improvement. However, being traditional in nature, these process mapping techniques are resource-intensive and time-consuming (Reijers et al., 2003). Other attempts have been made in the literature to '*redesign*' existing business processes by applying formal methods and theories, such as linear programming (Aldowaisan and Gaafar, 1999) and computational experiments (Hofacker and Vetschera, 2001). However, none of these attempts use information such as business policies (which could be unstructured) as their inputs.

In order to overcome the abovementioned challenges, there is a need for a system that can consider unstructured business policies and extract a business process map from them. The graphical representation of extracted business process maps may assist the decision maker to use it for the identification of any incompliance between operational business processes and an enterprise's business policies. The need for such system is increasingly becoming a subject of interest for enterprises seeking solutions to possible business mergers or to ensure that they are working in accordance with legal regulations or government policies.

7.3 Case study for problem definition

To explain the problem with an example, consider a case study of a public university in Australia to illustrate business policies for "Travel Bookings for University Staff". For a staff member to make a successful travel booking, different departments located within the university and/or in other universities need to collaborate with each other as defined in the business policies of the university, as depicted in Figure 7.3. The business policies may involve many tasks, data items, resources, constraints and actions, which make the corresponding business process model a significant one. However, it is assumed that these policies contain contradictory information as a result of the following:

- updating of a business policy over a period of time, and
- policy definitions by different departments located within the university and/or in other universities.

For demonstration purposes, a small set of travel policies from "Travel Bookings for University Staff" are selected as follows:

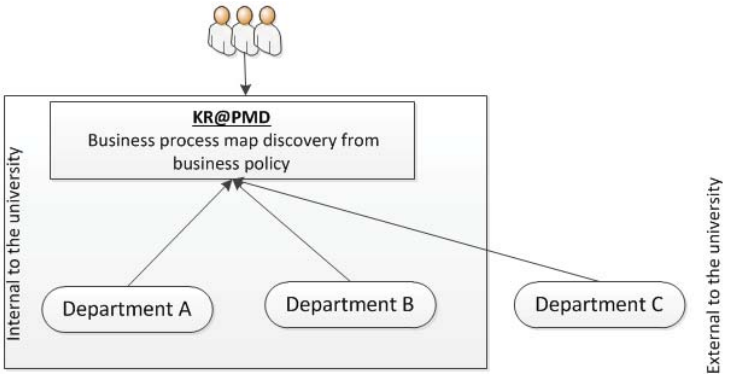


Figure 7.3: Interaction among departments for travel bookings for university staff

- *Process 1 : Authority to approve travel*

    1. Executive managers have the authority to approve travel but may also choose to authorise 'University officers' to approve travel on their behalf.
    2. However, approval of 'Business Class' fares and nomination of 'Level One Travellers' remains with the Executive Manager.

- *Process 2 :Travel booking for staff and associates*

    1. On receipt of the correctly filled travel form, the facilitator usually submits the travel booking form to the finance director for approval.
    2. If the form is not filled out correctly, the facilitator should not submit the form to the director for approval.
    3. On receiving travel approval from the director, the facilitator proceeds to make a booking for the traveller.
    4. For domestic travel, s/he uses the online booking tool and for international travel, s/he must book through a travel consultant.

The abovementioned unstructured business polices make it difficult for the decision maker to ensure that the operational business process for travel bookings for staff follow the business policies defined by the university. To overcome this problem, such an unstructured business policy needs to be considered by Web-based IDSS in order to extract a business process map that may explain the steps involved in a staff member making a travel booking. Such a business process map could be used as a tool to check the compliance of the business policy with existing operational business processes. In order to achieve this objective, the challenges that confront Web-based DSS are as follows:

- the need for a methodology to extract the concepts (i.e. elements of a process) from an unstructured business policy document and represent them in business rules format;
- the need for a reasoning mechanism that can activate/execute the business rules and is capable of providing methodologies for handling conflicts present among the business rules in different contexts, and
- the need for a mechanism to integrate the activated business rules in the form of a business process map and provide its graphical representation for the decision maker.

To achieve the abovementioned challenges, a Web-based DSS is required which has the following functionalities:

1. Specification of the domain knowledge in the form of an ontology.
2. An interface for decision makers to load the business policy document and extract the concepts (i.e. elements of the process) and annotate them with the domain ontology.
3. An interface for decision makers to specify business rules by using the concepts extracted and annotated with the domain ontology in the previous step.
4. A hybrid reasoning engine for the automated activation/execution of business rules and the provision of different conflict resolution strategies for decision maker to identify and resolve the conflicts among the activated business rules.
5. The generation of a business process map and the provision of its graphical representation for decision makers so that it can be used as a validation tool to check and ensure the compliance of operational business processes with business policies.

Assumptions

- a declarative language is used for specifying the business rules, and
- a declarative language has the capability to represent potentially incomplete and/or contradictory information present in the business policy.

To achieve the abovementioned functionalities, in the next section, a framework for business process map discovery from business policies using the knowledge representation approach with argumentative reasoning (KR@PMD) is proposed.

7.4 Proposed framework for KR@PMD

In this section, the solution for Process Map Discovery from business policies using a knowledge representation approach with argumentative reasoning (KR@PMD) is proposed to assist the decision maker of an enterprise or collaborating enterprises in the decision-making process. Figure 7.4 represents the proposed framework which consists of three layers as follows:

1. Information layer

   The information layer represents the unstructured information identified by the decision makers to be considered during the decision-making process. This information comprises different business policy documents in a textual format that outlines the series of steps to be followed.

2. @IRRI layer

   This layer comprises a logic-based framework that enables the applications located at Semantic Web application layer, such as Web-based DSS, to deal with unstructured business policies and generate the graphical representation of the process map to assist the decision maker in the intra-enterprise or inter-enterprise decision-making process. It enables different modules to load a policy document, perform semantic annotation followed by the specification of business rules and facts for reasoning, perform hybrid reasoning to identify and resolve conflicts and provide a graphical representation of a business process map for the decision maker that can assist him to validate the operational business process. The modules are as follows:

   (a) *An information representation module* that provides the following functionalities:

   - A Web-based form to load the business policy document and a domain ontology such as a process ontology [1] that enables the decision maker to extract and annotate information from the business policy with concepts defined in the process ontology and save the annotated information in the form of predicates in the relational database.
   - A Web-based form to specify the business rules and facts by using the annotated predicates saved in the relational database. The specified production rules in DeLP format are saved in the rule base and facts in DeLP format are saved in the working memory.

   (b) *An argumentative reasoning module* that provides the following functionalities:

   - Performs hybrid reasoning over underlying information in the knowledge base. The hybrid reasoning engine performs data-driven reasoning to activate the tasks in a process (i.e. arguments

[1] An introduction to process ontology is given in the next sub-section

construction) and goal-driven reasoning for conflicts identification among the tasks in a process followed by their resolution.

- Provides different argumentation-driven conflicts resolution strategies for the decision makers to resolve conflicts between arguments and their counter-arguments.

(c) *Information and knowledge integration module* that provides the following functionalities:

- Integration of the output of hybrid reasoning in the form of a business process map (i.e. reasoning chain).
- Graphical representation of the business process map generated in the previous step.

3. Semantic Web applications layer

This layer consists of KR@PMD that exploits the @IRRI layer and the information layer to achieve its objectives.

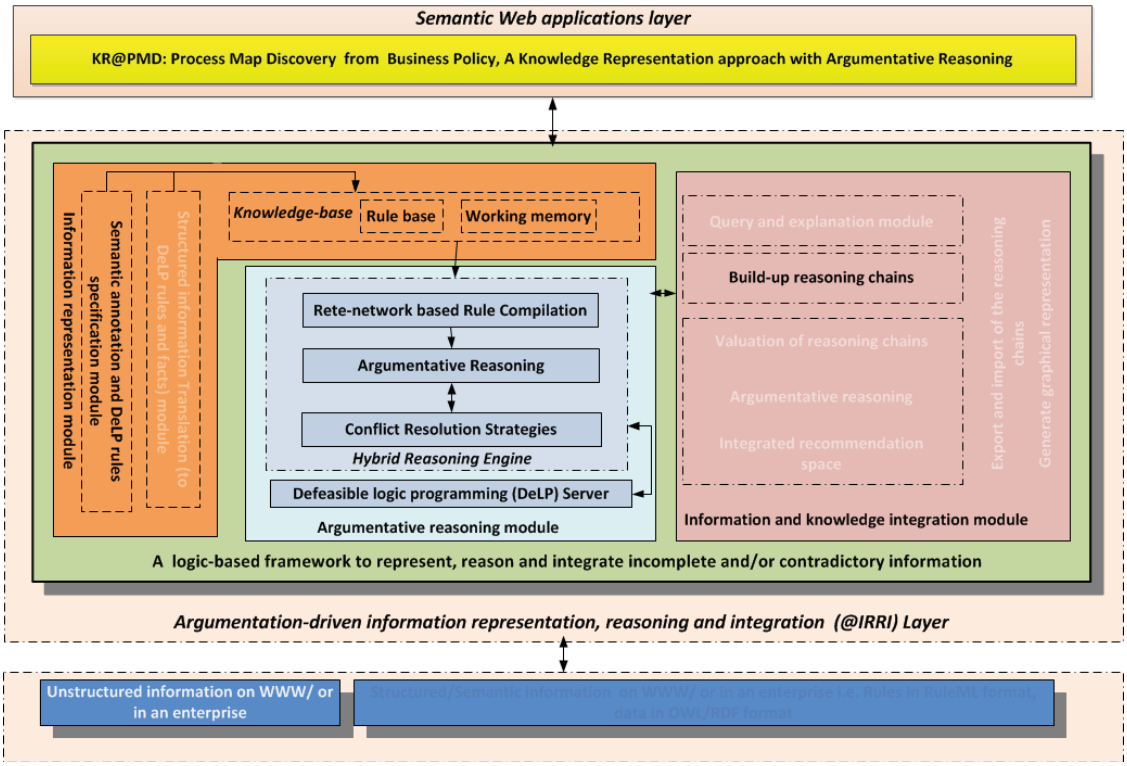


Figure 7.4: Proposed framework with highlighted components exploited by KR@PMD

Before explaining the working of the proposed framework, in the next sub-section, the process ontology which is used to annotate the unstructured business policy document in the proposed framework for KR@PMD is introduced.

7.4.1 Process ontology

A business policy document outlines the working of the business processes in an enterprise. Usually, an enterprise involved in collaboration either does not capture its business policies formally or it documents them in natural language. These documented business policies are for human consumption only and cannot be directly translated into a machine-processable format. To consider such business policies and make them understandable by Web-based DSS, a *'Process Ontology'* has been designed and developed which provides an explicit, declarative specification of business process concepts, mostly specified in business policy documents. In the proposed framework, the process ontology is used to annotate the unstructured business policies and make them understandable by Web-based DSS. Figure 7.5 shows the pictorial representation of the process ontology.

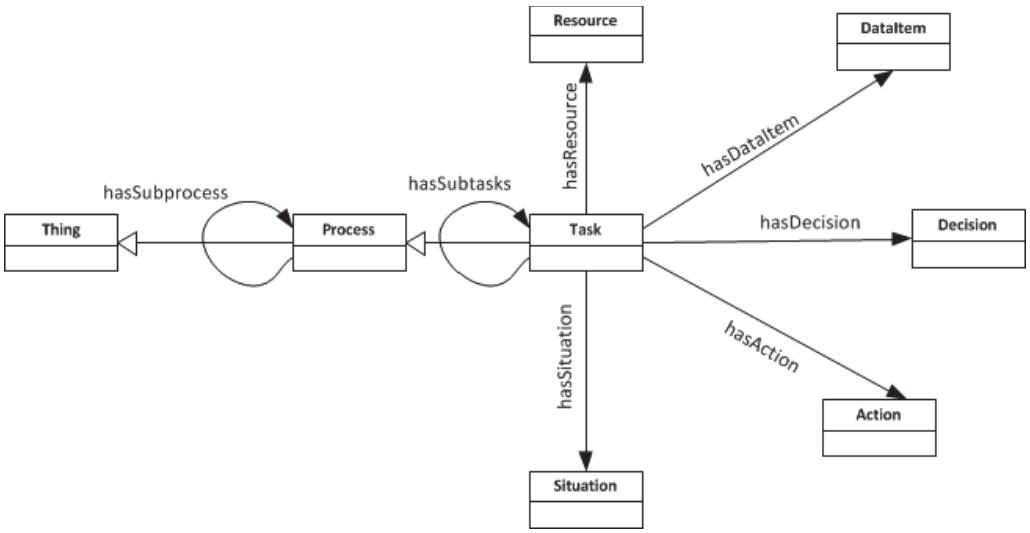


Figure 7.5: Pictorial representation of the process ontology

The process ontology is composed of the following important concepts:

1. process: represents a business activity that may comprise a set of different tasks.
2. task: represents a business activity that may comprise a set of resources, data-items, constraints, situations, decisions and actions.
3. resource: represents an enterprise asset needed to accomplish a certain task, e.g., credit card, manager etc.

4. data-item: represents the data/information required for the execution of a given task e.g., travel form.

5. constraint: represents certain limitations pertaining to the given task, resource or data-item.

6. decision: represents a decision point in a task.

7. situation: represents the presence of a conflict in a task.

8. action: represents the execution or transition of a current procedure or activity to another.

A process ontology is also enriched with different object properties, some of which are as follows:

1. hasSubProcess property

   This property is used to capture a relationship between a process and its sub-processes.

2. hasDependentTask property
   This property is used to capture the relationship between tasks that are dependent on each other for the completion of a process. In such cases, the commencement or completion of a task is dependent upon the commencement or completion of another task. Three different types of hasDependentTask relationships are identified as follows:

   - finishToStart(X,Y)

     Task Y is dependent on task X and task Y will start only when task X is finished. This is a very common type of relationship when the execution of tasks one after another is required.

   - startToStart(X,Y)

     Dependent task Y cannot begin until task X starts. This type of relationship is useful for tasks that do not share information; as a result, they can be executed in parallel.

   - finishToFinish(X,Y)

     Dependent task Y cannot be completed until task X is finished. In this relationship, dependent task Y needs information from task X for its completion.

3. hasResource property
   This property is used to capture the relationship of a task with a resource needed for its execution. Examples of such a relationship are *'task assignment to a person'*, *'owner of a task'*, etc.

4. hasDataItem property
   This property is used to capture the relationship of a task with a data-item it needs for its execution. The possible word representation of such relationships in business policies are '*contain*', '*refer*', '*include*', etc.

5. hasSituation property
   This property is used to capture the relationship of a task with a situation which needs the attention of the manager for its resolution. One example of such a situation is the existence of conflict among the business rules. This relationship is not modelled at design time; rather, it is used by the proposed framework to annotate the business process map where such a situation exists.

6. hasDecision property
   This property is used to capture the relationship of a task with a decision made during its execution.

7.4.2 Working of the proposed framework for KR@PMD

In this section, the working of the proposed framework for business process map discovery from business policies to assist the decision maker in the identification of any incompliance between business policies and operational business processes is discussed. Figure 7.6 depicts the working of the proposed framework. The sequence of steps in the proposed framework are as follows:

1. Semantic annotation of unstructured business policies for business rules specification
   For the extraction and specification of business rules from an unstructured business policy document, the KR@PMD exploits the functionality of the *information representation module* of the logic-based framework located at @IRRI layer. It provides a Web-based form for the decision maker to load the unstructured business policy document, annotate the loaded information with the concepts defined in the process ontology and make it available for the specification of the business rules. The process involves the following two steps:

(a) Semantic annotation of business policies

During this step, a Web-based form is provided for the decision maker to load the policy document, extract the information (i.e., elements of a process) from the business policies and annotate them with the concepts defined in the process ontology. The annotated information is then saved in the relational database.

(b) Specification of business rules and facts

Once the semantic annotation of the business policies is completed, the next step is to use the annotated information saved in the relational database for the specification of business rules and facts and make the information ready for further processing.

For the specification of business rules and facts, a Web-based form is provided that loads the annotated predicates from the relational database (from step 1 (a)) and makes them available for the decision maker for the specification of business rules and facts. The specified business rules are saved as DeLP rules in the rule base and facts are saved as DeLP facts in the working memory.

2. Argumentative production system performing hybrid reasoning

Once the knowledge base has been created i.e., a rule base containing DeLP rules and the working memory containing DeLP facts, KR@PMD exploits the functionality of *argumentative reasoning module* of the logic-based framework located at the @IRRI layer for reasoning, identification and resolution of conflicts for information integration. The process involves the following three steps:

(a) Process activation using data-driven reasoning

In this step, the business rules in DeLP format are activated and fired. An important point to note is that DeLP rules consist of a set of premises and a conclusion, each of which represents an element of a business process. Therefore, the activation of DeLP rules is actually the activation of the element of a business process. In the proposed framework, this objective is achieved by performing data-driven reasoning over the underlying information (i.e. DeLP rules and DeLP facts). This step involves the following two sub-steps:

- Compilation of DeLP rules in the form of a Rete network.

- Perform data-driven reasoning by introducing certain DeLP facts from the working memory to the Rete network.

As a result of data-driven reasoning, the activated business process (i.e. represented as a set of arguments) are saved as the *Activated Process set*. Two types of arguments are constructed during this phase: strict arguments and defeasible arguments.

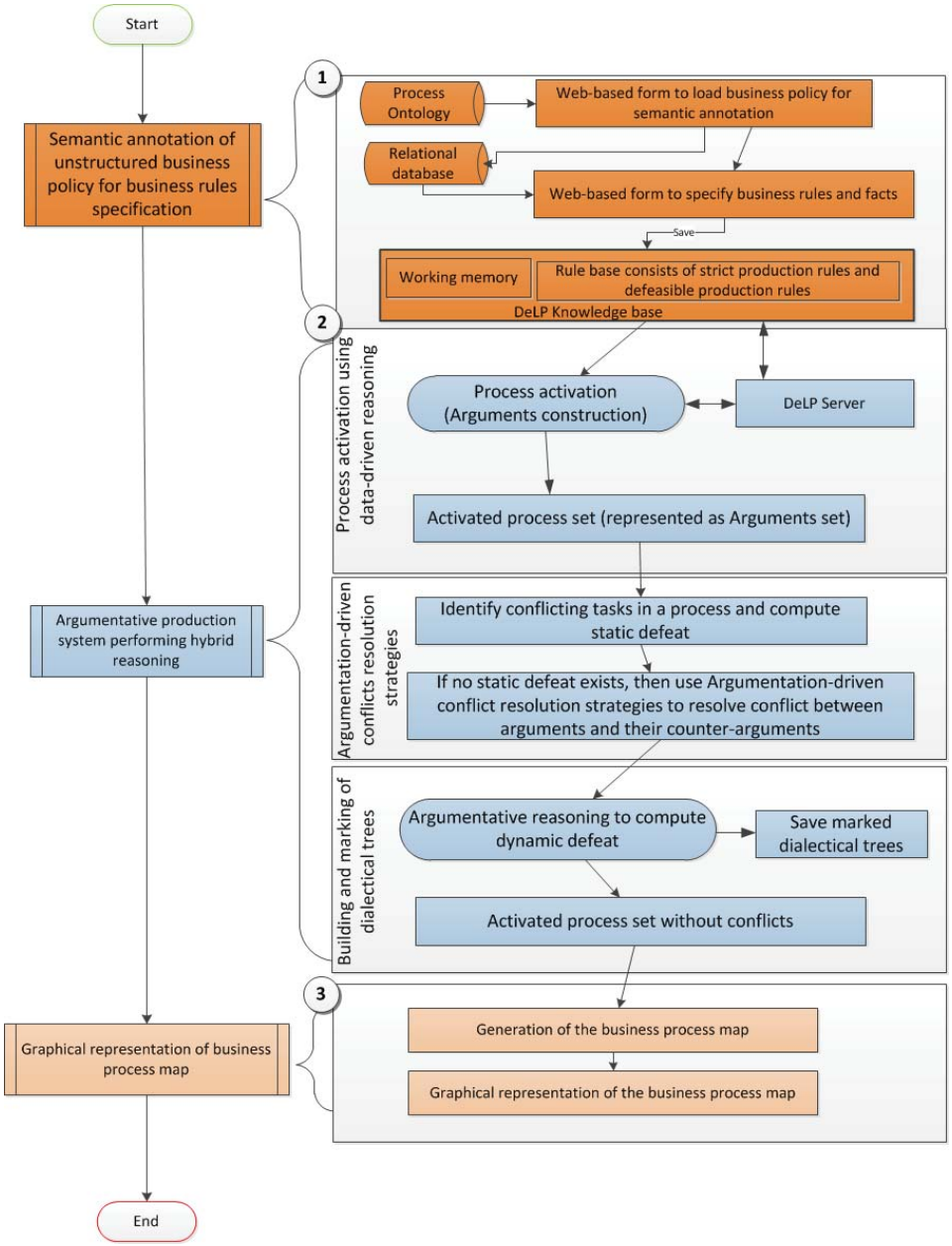


Figure 7.6: Flowchart illustrating sequence of steps performed by KR@PMD

(b) Argumentation-driven conflict resolution strategies

Once the activation of business processes is completed, the next step is the identification and resolution of conflicts present in a business process (i.e. represented as an argument and its counter-argument) by computing either static or dynamic defeat. If static defeat does not exist, then dynamic defeat is computed which involves the following steps:

- resolve the conflict between an argument and its counter-argument by using the selected argumentation-driven conflict resolution strategy by the decision maker, and
- build and mark the dialectical trees for the arguments involved in a conflict.

The defined argumentation-driven conflict resolution strategies that can be selected by the decision maker are as follows:

- Generalize specificity: This is an automated conflict resolution strategy where an argument that is specific defeats its counter-argument which is less specific. The Generalize specificity conflict strategy resolution of the conflicts between arguments is explained in Section 5.5.2.
- Dung's style: This is also an automated conflict resolution strategy where an argument X (part of a reasoning chain Y) is attacked by a counter-argument Z (not part of reasoning chain Y) and X is defeated by Z if there is no other argument (in reasoning chain Y) that attacks and defeats the counter-argument Z.
- Fuzzy preferences: This is a semi-automated conflict resolution strategy that takes input from the decision makers about their preference between an argument and its counter-argument in fuzzy terms such as definitely preferred, slightly preferred or no preferences etc. and uses them to establish a priority between an argument and its counter-argument.
- Voting: This is a semi-automated conflict resolution strategy that considers input from a number of decision makers who are either in favour or against an argument involved in the conflict. The argument that has more votes in favour defeats its counter-argument that is less favoured.

Further explanation about Dung's style, Voting and Fuzzy preferences is given in Sections 7.6.2.2, 7.6.2.3 and 7.6.2.4, respectively.

(c) Building and marking of dialectical trees

Once the conflict between an argument and its counter-argument has been resolved using the argumentation-driven conflict resolution strategy, the next step is to construct and mark dialectical trees (as defeated or undefeated), as discussed in Section 5.5.2. The marked dialectical trees are used by the argumentative production system to establish the preference between arguments and their counter-arguments. The marked dialectical tree is then saved for future use, such as to provide an explanation of the output of conflict resolution for the decision maker.

3. Graphical representation of the business process map

Once the hybrid reasoning process is complete, KR@PMD exploits the functionality of the *information and knowledge integration module* of the logic-based framework to integrate the information obtained from hybrid reasoning and display it to the decision maker. This process involves the following steps:

(a) Generation of business process map

During this process, the arguments present in the Activated Process set are linked in the form of a map known as the business process map.

(b) Graphical representation of the business process map

Once the generation of the business process map is complete, the next step is to provide a graphical representation of the business process map for the decision maker.

In the following sections, the working of each step is explained in detail.

7.5 Semantic annotation of unstructured business policies for business rules specification

To consider unstructured business policies for the specification of business rules, KR@PMD exploits the functionality of the *information representation module* of the logic-based framework located at @IRRI layer. This module helps the KR@PMD to

load and annotate the business policy with a process ontology and use the annotated predicates for the specification of DeLP rules and DeLP facts and save them in the knowledge base. Figure 7.7 illustrates the steps performed by KR@PMD for information representation. These steps are as follows:

1. Semantic annotation of business policy

   For the semantic annotation of the business policies, KR@PMD provides a Web-based form for the decision maker to load the business policy document. Once it is loaded, the information is displayed to the decision maker from where he can extract the process elements and annotate the concepts defined in the process ontology. The resulting annotated information in the form of predicates is saved in the relational database.

2. Specification of business rules and facts

   Once the semantic annotation of the business policies is completed, the next step is to use the annotated information saved in the relational database for the specification of DeLP rules and facts.

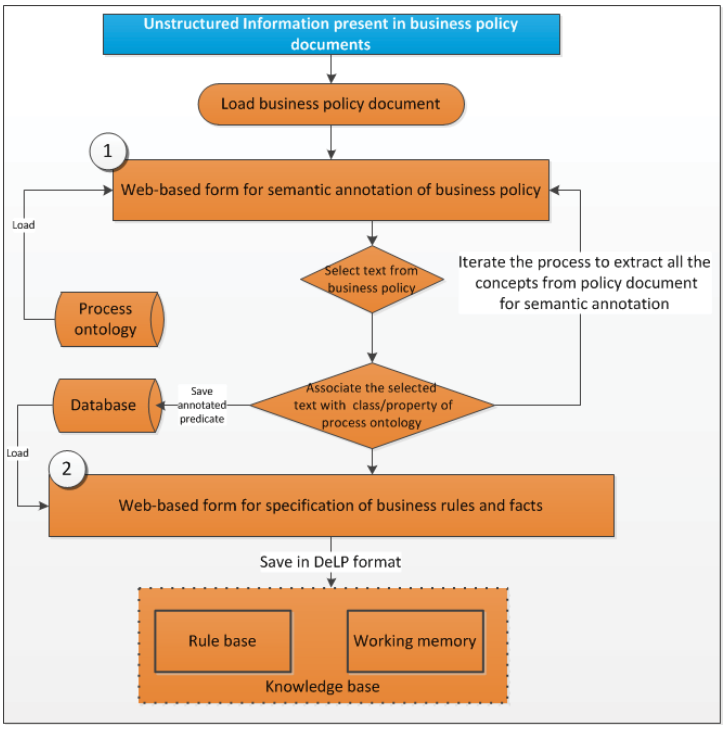


Figure 7.7: Flowchart illustrating steps performed by KR@PMD for semantic annotation and production rules specification

In the next sub-sections, each of these steps is discussed in detail.

7.5.1 Semantic annotation of business policies

This step involves reading the business policy document by KR@PMD stored on the local machine or downloaded from WWW and making it ready for the purpose of semantic annotation. During this process, a Web-based form is provided to load the process ontology and make its concepts available for the annotation of the business policies. The decision maker can extract the process elements and annotate them with the concepts defined in the process ontology. This process is repeated for all the important information in the business policies. The resulting annotated information in the form of predicates is saved in the relational database. It is important to note here that the use of ontology-based business rules improves the shared understanding of business rules and thus their reusability will be enhanced as required in an open environment such as the Web and in enterprises.

To explain the Semantic annotation of business policies, consider the case study discussed in Section 7.3 where the business process are; 1: authority to approve travel and processes; and 2: travel bookings for staff and associates are provided in an unstructured format. Figure 7.8 provides the graphical representation of the semantic annotation of the 'Travel Bookings for University Staff' policy with a process ontology where:

- *traveller* is an instance of the class *Resource* and has a data property *Name*. Similarly, *Submit* is an instance of the *Action* class with the data properties *PersonName* and *FormName*. The data properties for the rest of the concepts are likewise defined in an ontology;
- finishToStart(X,Y): Similarly, Process 2, i.e. *Travel booking for staff*, will start only when Process 1, i.e. *Authority to approve travel*, ends. The information flows from Process 1 to Process 2;
- hasResource property: Similarly, the approval for travel, i.e. *authorize(X,Y)* is a hasResource property in which *executiveManager* X authorizes *universityOffice* Y to approve travel on his/her behalf, and
- information such as '*fill booking form*' and '*correctly filled form'* are examples of the hasDataitems property.

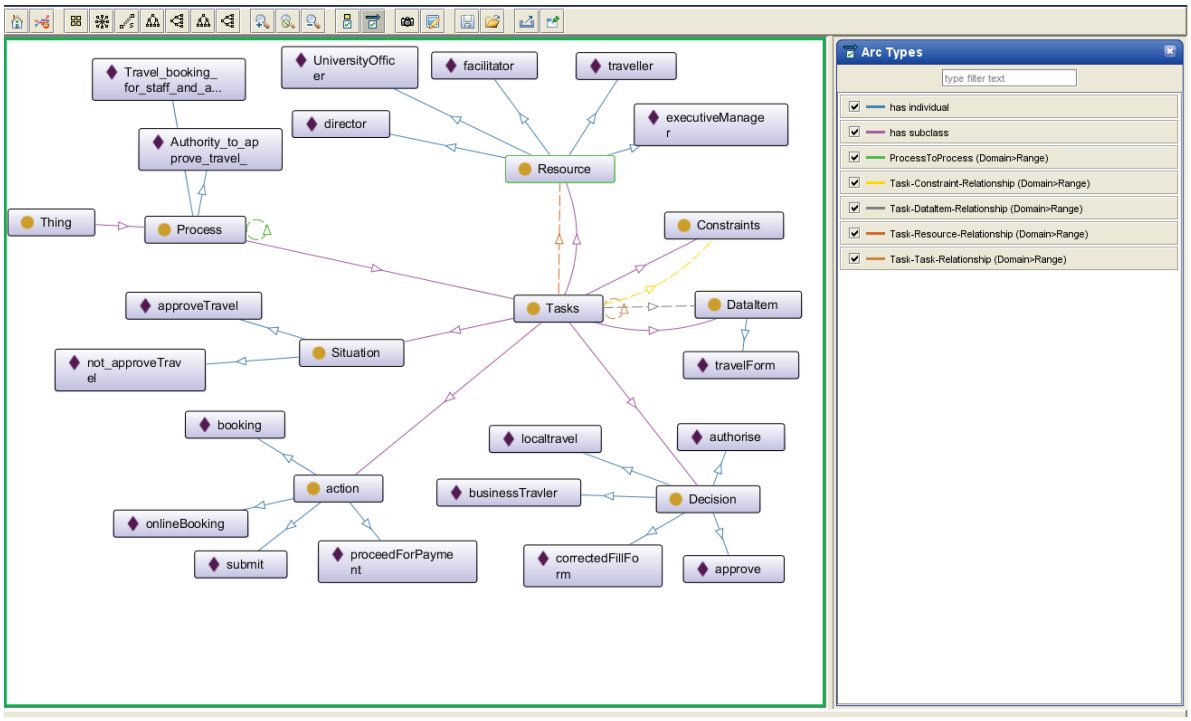


Figure 7.8: Graphical representation of annotation of travel policy with the process ontology

7.5.2 Specification of business rules and facts

Once the business policy has been annotated, the next step is to use these predicates for the specification of business rules into DeLP format. To achieve this objective, KR@PMD provides a Web-based form for the decision maker to specify the business rules by using semantically annotated information and saves them in DeLP format in the rule base. Section 5.3.1 provides details about the syntax and semantics of strict and defeasible DeLP rules. These syntax and semantics are considered during this process for the specification of business rules and facts.

To explain the specification of business rules and facts, Figure 7.9 depicts the Web-based form provided by KR@PMD to the decision maker for the specification of business rules and facts by using annotated information. The decision maker can give a name to the business rule, select its type as either strict or defeasible, select a set of premises and a conclusion. The specified business rule is then saved as the DeLP rule in the rule base. Similarly, he specifies the facts and saves them as DeLP facts in the working memory.

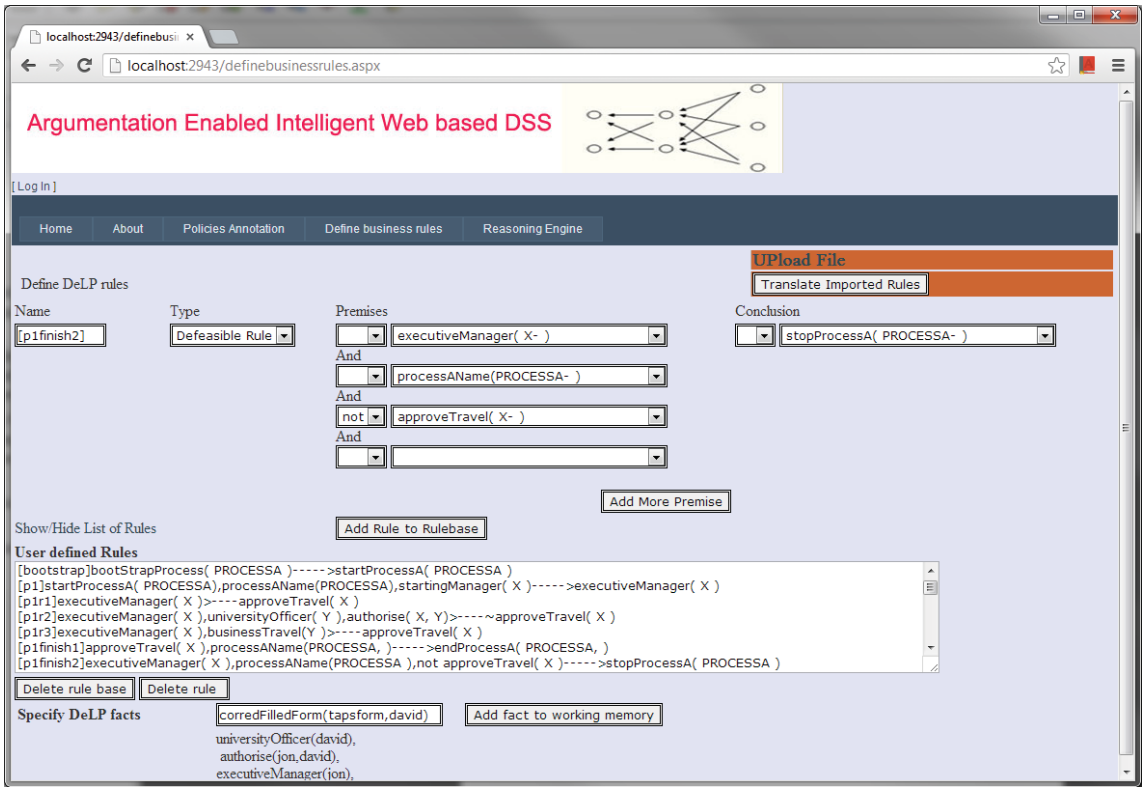


Figure 7.9: A Web-based form of KR@PMD for the specification of business rules and facts

Illustration 7.1 demonstrates the DeLP rules and illustration 7.2 demonstrates the DeLP facts specified for the case study discussed in Section 7.3. From illustration 7.1, the business rules *'p4r4'* and *'p4r7'* are examples of strict business rules and the business rules *'p1r1'* , *'p1r2'* and *'p1r3'* are examples of defeasible business rules. The business rule *'p4start'* represents the realization of *finishToStart (X, Y)* object property in the form of a rule. Similarly, the rest of the object properties i.e. *startToStart(X, Y)* and *finishToFinish(X, Y)* relationships can be realized in rule form, as follows:

$$startToStart(X,Y), start(X) \longrightarrow start(Y)$$

$$finishTofinish(X,Y), end(X) \longrightarrow end(Y)$$

Business rules for Travel Bookings for University Staff

$$\left\{\begin{array}{l}
[bootstrap] bootStrapProcess(PROCESSA) \longrightarrow startProcessA(PROCESSA) \\
[p] startProcessA(PROCESSA), processAName(PROCESSA), \\
startingManager(X) \longrightarrow executiveManager(X) \\
[p1r1] executiveManager(X) \dashrightarrow approveTravel(X) \\
[p1r2] executiveManager(X), authorise(X,Y), universityOfficer(Y), \dashrightarrow\sim approveTravel(X) \\
[p1r3] executiveManager(X), businessTravel(Y) \dashrightarrow approveTravel(X) \\
[p1finish1] approveTravel(X) \longrightarrow endProcessA(PROCESSA) \\
[p1finish2] not\ approveTravel(X) \longrightarrow stopProcessA(PROCESSA) \\
[p4start] finishToStart(PROCESSA, PROCESSB), endProcessA(PROCESSA), \\
processBName(PROCESSB,) \\
\longrightarrow start(PROCESSB) \\
[p4](PROCESSB), processBName(PROCESSB), startingTraveller(X) \longrightarrow traveller(X). \\
[p4r1] traveller(X,), fillForm(TAPS) \dashrightarrow corredFilledForm(X, TAPS) \\
[p4r2] corredFilledForm(X, TAPS), traveller(X), facilitator(Y) \dashrightarrow submit(TAPS, Y) \\
[p4r3] director(DEPT), traveller(X), submit(TAPS, Y), approvedBy(DEPT) \dashrightarrow booking(Y, X). \\
[p4r4] booking(Y, X), localTravel(X) \longrightarrow onlineBooking(Y, X) \\
[p4r5] onlinebooking(Y, X) \dashrightarrow proccedForPayment(Y). \\
[p4finish1] proccedForPayment(Y), processBName(PROCESSB), \\
start(PROCESSB) \longrightarrow end(PROCESSB)
\end{array}\right\} Illustration(7.1)$$

Initial working memory

$$\left\{\begin{array}{l}
universityOfficer(david), authorise(jon, david), executiveManager(jon), \\
, bootStrapProcess(processone), processAName(processone), businessTravel(category), \\
corredFilledForm(tapsform, david), processBName(processfour), \\
finishToStart(processone, processfour), facilitator(david), fillForm(tapsform), \\
startingManager(jon), startingTraveller(naeem),
\end{array}\right\} Illustration(7.2)$$

Once the business rules have been specified, the next step is to execute the business rules. The execution of business rules is handled by the argumentative production system. In the next section, the working of the main components of the argumentative production system is elaborated.

7.6 Argumentative production system performing hybrid reasoning

Once the required information from the business policies has been captured (as DeLP rules and facts) and saved in the knowledge base, the next step is to perform reasoning over it and make it ready for EII. To address this objective, there is a need for a hybrid reasoning methodology that can reason over the captured information and resolve any conflicts that may arise during the reasoning process before business process map generation. An argumentative production system [2] exploits the functionality of the *argumentative reasoning module* of the logic-based framework located at @IRRI layer. Figure 7.10 illustrates the steps performed by the argumentative production system for hybrid reasoning over the captured information. These steps are as follows:

[2] In Section 5.3.1, several important definitions were introduced to help the reader understand the design and working of the argumentative production system.

1. Process activation using data-driven reasoning

   During this step, data-driven reasoning is performed over underlying information (i.e. DeLP rules and DeLP facts) for the activation of business processes (i.e. in form of arguments). This step involves the following sub-steps:

   - Compilation of DeLP rules in the form of a Rete network.
   - Perform data-driven reasoning by introducing certain DeLP facts from the working memory to the Rete network.

   The activated business processes are saved as the *Activated Process set*. Two types of arguments are constructed during this phase: strict arguments and defeasible arguments.

2. Argumentation-driven conflicts resolution strategies

   Once the activation of business processes is complete and the Activated Process set is created, the next step is the identification and resolution of conflicts present in a process (represented as arguments their counter-arguments) by computing either static or dynamic defeat. If static defeat does not exist, then dynamic defeat is computed which involves the following steps:

   - resolve the conflict between an argument and its counter-argument using argumentation-driven conflict resolution strategy, and
   - build and mark dialectical trees for the arguments involved in a conflict.

   The different argumentation-driven conflict resolution strategies are Generalize specificity, Dung's style, Voting and Fuzzy preferences.

3. Building and marking dialectical trees

   Once the conflict has been resolved between an argument and its counter-argument using the argumentation-driven conflict resolution strategy, the next step is to construct and mark dialectical trees.

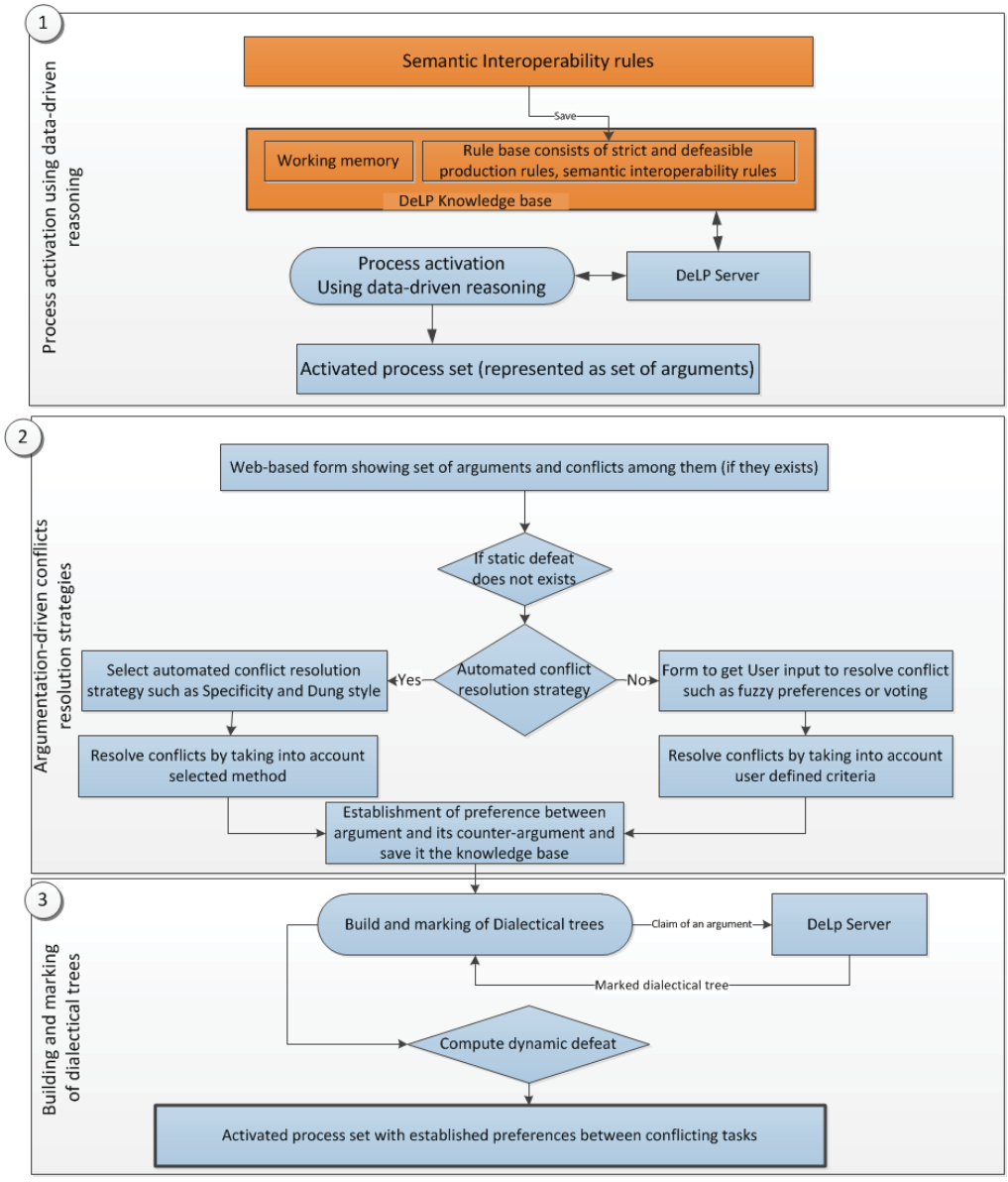


Figure 7.10: Flowchart illustrating steps performed by of KR@PMD during performing hybrid reasoning

In the next sub-sections, each of these steps is discussed in detail.

7.6.1 Process activation using data-driven reasoning

The process activation is a two steps process as follows:

1. Complication of business rules in the form of a Rete network. In Section 5.5.1, the detail of the compilation of business rules as a Rete network is outlined and the extensions made to the Rete network to represent incomplete and/or

contradictory information is also given.

2. The second step is to perform data-driven reasoning over underlying information by passing the facts present in the working memory through the Rete network. This results in the activation of production rules called the Activated Process set. Section 5.5.1 outlines the detailed working of data-driven reasoning.

During data-driven reasoning, the match and execute cycle results in the activation of one-input nodes only if they match the facts coming from the working memory. However, this has a drawback as the business rules specified on top of the ontology e.g. the process ontology discussed in Section 7.4.1 may not be activated due to the absence of matching facts in the working memory even though the working memory contains the facts that semantically match with the one-input node.

To explain the problem with the help of an example, consider two concepts, 'traveller' and 'person', represented in an ontology such that traveller is a subclass of person. Furthermore, consider that the rule-base contains a business rule i.e $[i]traveller(X) \rightarrow giveDicount(X)$ which means that if X is a traveller, then he must be given a discount, and the working memory contains facts such as person(perth) and subClass(traveller, person). Now, if data-driven reasoning is performed, then the business rule i, specified in the rule base, will not be activated because of the absence of the fact 'traveller' in the working memory. In order to address this problem, a set of semantic inter-operability rules, as shown in Table 7.1 are saved in the knowledge base before data-driven reasoning is performed. These rules provide ontology schema translation in order to bring semantic inter-operability among the business rules specified in DeLP format. As a result, during data-driven reasoning, the activation and firing of rule 2 depicted in Table 7.1 i.e., $subClass(traveller, person), person(X) \rightarrow traveller(X)$, adds the new fact i.e. traveller(perth), in the working memory. The addition of this new fact will result in the activation of business rule i.

Rule 1	type(X,C)→C(X)	Class
Rule 2	subClassof(Sc, C), Sc(X)→C(X)	Subclass
Rule 3	objectProperty(X), domain(X, Y), range(X,Z) →X (Y, Z)	Object Property
Rule 4	objectProperty(X), X(Z, V), subProperty(X, Y)→Y(Z, X)	subProperty
Rule 5	dataProperty(X), domain(X, Y), range(X, Z) →X(Y, Z)	Data Property
Rule 6	dataproperty(X), X(Z, V), subProperty(X, Y)→Y(Z, X)	SubProperty

Table 7.1: Ontology schema translation rules in DeLP format

To explain process activation using data-driven reasoning with the help of an example, Illustration 7.3 shows the activated processes generated during data-driven reasoning using DeLP rules (specified in illustration 7.1) and DeLP facts (specified in illustration 7.2).

Activated Process set

$$\left\{\begin{array}{l}
[bootstrap]bootStrapProcess(processone) \longrightarrow startProcessA(processone) \\
[p1]startProcessA(processone), processAName(processone), \\
startingManager(jon) \longrightarrow executiveManager(jon) \\
[p1r1]executiveManager(jon) \dashrightarrow approveTravel(jon) \\
[p1r2]executiveManager(jon), universityOfficer(david), authorise(jon, david) \\
\dashrightarrow\sim approveTravel(jon) \\
[p1r3]executiveManager(jon), businessTravel(david) \dashrightarrow approveTravel(jon) \\
[p1finish1]approveTravel(jon), processAName(processone,) \longrightarrow endProcessA(processone) \\
[p1finish2]executiveManager(jon), processAName(processone), not\ approveTravel(jon,) \\
\longrightarrow stopProcessA(processone) \\
[p4]startProcessB(processfour), processBName(processfour), startingTraveller(naeem,) \\
\longrightarrow traveller(naeem) \\
[p4r1]traveller(naeem), fillForm(tapsform) \dashrightarrow corredFilledForm(naeem, tapsform) \\
[p4start]finishToStart(processone, processfour), endProcessA(processone) \\
processBName(processfour) \longrightarrow startProcessB(processfour) \\
[p4r2]corredFilledForm(naeem, tapsform), traveller(naeem) \\
facilitator(david) \dashrightarrow submitForm(tapsform, david) \\
[p4r3]submitForm(tapsform, david), traveller(naeem), director(smith), \\
approvedBy(smith) \dashrightarrow booking(david, naeem) \\
[p4r4]booking(david, naeem), localTravel(naeem) \longrightarrow onlineBooking(david, naeem) \\
[p4r5]onlineBooking(david, naeem) \dashrightarrow proccedForPayment(naeem) \\
[p4finish1]proccedForPayment(naeem), startProcessB(processfour), \\
\qquad processBName(processfour) \longrightarrow endProcessB(processfour)
\end{array}\right\} Illustration(7.3)$$

It is important to note that the activated process set represents the elements of a process in business rules format. Therefore, when the business rules are activated, it is actually the activation of the elements of a process as depicted in Figure 7.11.

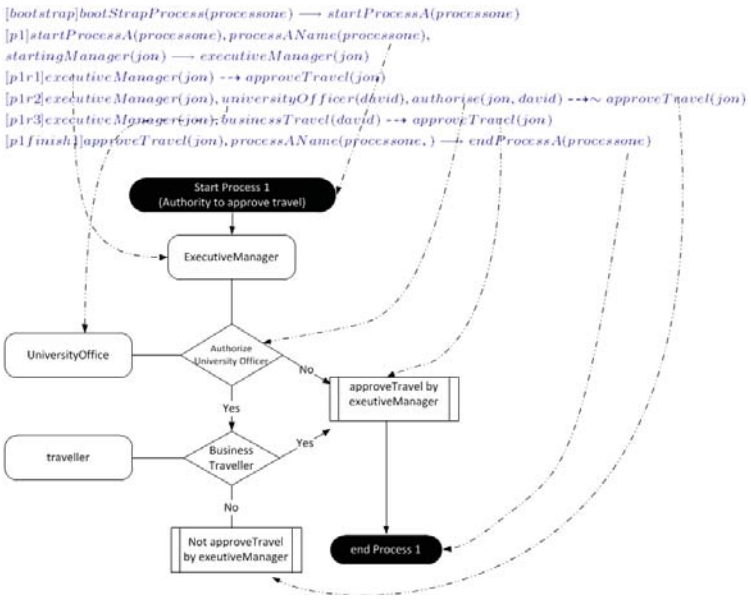


Figure 7.11: Pictorial representation of the mapping of activated business rules in a business process map

Once the activation of the process is complete, the next step is the identification of conflicts between the activated business rules and their resolution. As defined in Section 5.5.2, two different types of priorities exist during the process of argumentation as follows:

1. static priority

   If a strict argument is in conflict with a defeasible argument, the strict argument always has higher priority than the defeasible argument. This is known as static priority.

2. dynamic priority

   Dynamic priority establishment among defeasible arguments involves two steps:

   (a) Identify the conflict and establish a priority between an argument and its counter-arguments using different argumentation-driven conflict resolution strategies.

   (b) Build and mark dialectical tress to obtain the priority status of an argument over its counter-argument by considering the entire knowledge base.

   In the next sub-sections, the two steps involved in the resolution of conflicts between an argument and its counter-argument using dynamic priority are discussed.

7.6.2 Argumentation-driven conflict resolution strategies

To identify conflicts and establish priorities between an argument and its counter-arguments, four different conflict resolution strategies are used. Each conflict resolution strategy takes into account different criteria for the establishment of priority between an argument and its counter-argument and uses it to resolve conflict. The four different conflict resolution strategies are as follows:

7.6.2.1 Generalize specificity

Defeasible logic programming (DeLP) has a built-in mechanism for establishing priority between contradictory business rules, known as Generalize specificity. In Section 5.5.2, details were provided on the working of this strategy.

7.6.2.2 Dung style

Dung's style of argumentation takes a very influential approach to conflict resolution and has been discussed in detail in Section 2.5.1.1. Dung defined the characteristics of the argumentative framework according to the attack relationship between arguments and between sets of arguments.

Using Dung's framework, arguments are categorised into two different sets as follows:

- Conflict free set : A set of arguments S is said to be conflict free if it does not attack itself i.e. there is no argument $A \in S$ such that S attacks A.
- Admissible set : If a set of arguments S is conflict free and if an argument (i.e. a $\in S$) is attacked by another argument (i.e. $b \notin S$), then if there is another argument (i.e. $c \in S$) that attacks the argument b, then S is said to be an admissible set.

To explain the realization of Dung's framework in the proposed framework, first the argumentative production system in Dung's style after the argument construction phase is defined as follows:

$$\mathbb{KBS}=(\mathcal{WM}\prime,\mathcal{R},Args) \quad \text{.....Equation (7.1)}$$

where $\mathcal{WM}\prime$ represents the new state of the working memory after argument construction and $Args$ contains a set of arguments and the relations between them. Two types of relationships exist in $Args$; namely, the counter-argument relationship and the sub-argument relation. The counter-argument relationship is represented as an attack relationship in Dung's framework. Therefore, KBS defined in terms of Dung's framework is as follows:

(a) Conflict free : Given an $Args = (\mathcal{A}, attacks)$. A set $S \subseteq \mathcal{A}$ is conflict-free in $Args$, if, for each $a, b \in S$, $(a, b) \notin attacks$.

(b) Admissible Set : Given an $Args = (\mathcal{A}, attacks)$. A set S$\subseteq \mathcal{A}$ is admissible in $Args$, if

i. S is conflict-free in $Args$, and

ii. $a \in \mathcal{A}$ is defended by S in $Args$, if for each $b \in \mathcal{A}$ with $(b, a) \in attacks$, there exists a $c \in S$, such that $(c, b) \in attacks$.

To explain with the help of an example, consider three arguments, p1r1, p1r2 and p1r3, from illustration 7.3 where p1r1, p1r2 and p1r2, p1r3 are in conflict with each other. If Dung's framework is implemented for conflict resolution, this will result in the following sets:

Conflict free set ={p1r3, p1r1}and {p1r2}

Admissible set= {p1r3, p1r1}

The admissible set in the proposed framework is represented as a reasoning chain, where if an argument (i.e. p1r1) in a reasoning chain is attacked by another argument (i.e. p1r2) which is not part of that reasoning chain, then there exists another argument (i.e. p1r3) within the reasoning chain that helps the argument (i.e.p1r1) to withstand the attack from the argument (i.e. p1r2).

7.6.2.3 Fuzzy preferences

This approach for conflict resolution takes into account the fuzzy preference relation given by the decision makers between an argument and its counter-argument and uses this to determine the priority between them. In order to realize this approach in the proposed framework, the fuzzy preference relations approach defined by Kacprzyk et al. (1992) is followed.

To explain the working of fuzzy preferences, consider a set of decision makers in a group represented as DM=$\{dm_1, dm_2, \ldots dm_n\}$ and a set of arguments involved in a conflict as $Args$=$\{a_1, a_2, \ldots a_n\}$. Each decision maker $dm \, \epsilon \, DM$ gives his/her preferences over $Args$. A fuzzy preference relation Rc, of a decision maker dm_c, by its membership function μ_{Rc} is a cartesian product over $Args$ i.e. $Args \times Args \rightarrow$[0,1] such that

$$\mu_{Rc}(a_i, a_j) = \begin{cases} 1 & \text{if } a_i \text{ is definitely preferred over } a_j \\ d \in (0.5, 1) & \text{if } a_i \text{ is slightly preferred over } a_j \\ 0.5 & \text{no preference} \\ s \in (0, 0.5) & \text{if } a_j \text{is slightly preferred over } a_i \\ 0 & \text{if } a_j \text{ is definitely preferred over } a_i \end{cases} \tag{7.2}$$

Equation 7.2 results in the computation of a preference relation between an argument and its counter-argument individually by each member involved in group decision making. However, to compute the overall strength of a preference between an argument and its counter-argument, a fuzzy linguistic qualifier Q (Zadeh (1983)) i.e. $\mu_{strength}$ is used as a fuzzy set defined in [0,1]. The values obtained by the individual preference of each member for an argument is added together and divided by the number of decision makers involved in the decision-making process, as represented in equation 7.3

$$x = \frac{sum(individuals\ preferrence\ over\ an\ arugment\ and\ its\ counter-argument)}{number\ of\ decision\ makers} \tag{7.3}$$

Once x has been computed for an argument and its counter-argument, then the fuzzy linguistic Q i.e. 'strength', may be given as

$$\mu_{strenth}(x) = \begin{cases} 1 & for\ x \geqslant 0.8 \\ (2x - 0.6) & for\ \ 0.4 < x < 0.7 \\ 0 & for\ x \leqslant 0.3 \end{cases} \tag{7.4}$$

which may be interpreted as follows: if the strength of a preference is at least 0.8 or above, it results in the establishment of the priority of an argument over its counter-argument. If the strength of a preference is 0.3 or below, it results in the priority of a counter-argument over an argument. If the value of preference falls between 0.4 to 0.7, it results in no priority between an argument and its counter-argument.

To explain with an example, consider illustration 7.3 which shows that there are contradictory arguments p1r1,p1r2 and p1r3,p1r2 and there are three decision makers. Matrix (a) represents the preferences of each decision maker in relation to the arguments involved in a conflict. Considering equation 7.2 and arguments (p1r1,p1r2), dm_1 states that argument p1r1 is slightly preferred over argument p1r2 and dm_2 states that no preference exits between arguments p1r1 and p1r2 whereas dm_3 states that argument p1r2 is slightly preferred over argument p1r1.

(a)=		dm_1	dm_2	dm_3
	$(p1r1, p1r2)$	0.6	0.5	0.2
	$(p1r3, p1r2)$	0.2	0.8	0.7

To compute the $\mu_{most}(x)$, equation 7.3 and equation 7.4 is used which results in obtaining matrix b.

$$\text{(b)}= \begin{array}{l|l} (p1r1, p1r2) & 1.3/3 = 0.43 \\ (p1r3, p1r2) & 1.7/3 = 0.56 \end{array}$$

The results in matrix (b) can be interpreted as follows: the strength of attack i.e. 0.43, results in the priority of counter-argument '*p1r2*' over argument '*p1r1*'. Similarly, the strength of attack i.e. 0.56, results in the priority of *'p1r3'*

7.6.2.4 Voting

The last method for conflict resolution is a voting mechanism (Dong et al., 2010) that takes the decision-making problem to a wider set of audiences, along with experts who design the business policies. The definition of voting-based conflict resolution is as follows:

$$\text{Conflict}_{value} = \alpha \times \frac{\Sigma_{i=1}^{n} Vote_{User\ i}}{n} + \beta \times \frac{\Sigma_{j=1}^{m} Expert_{User\ j}}{m} \quad (7.5)$$

where

n is the number of users voting for a given conflict;

m is the number of experts voting for a given conflict;

α is the weight of the votes from normal users;

(β is the weight of the votes from experts, and α+β=1. The votes for the users i.e. $vote_{user}$ and votes for the experts i.e. $vote_{expert}$, are defined for each argument and counter-argument as follows:

(a) The voting result from the audience's preference i.e. $vote_{user}$, varies between

0 Survive

0.5 Undefeated

1 Defeat

(b) The voting result from the policy maker's preference $vote_{expert}$, varies between

0 Survive

0.5 Undefeated

1 Defeat

To explain with an example, consider contradictory arguments p1r1, p1r2 and p1r3, p1r2 from illustration 7.3 and the voting process which involves two users and three experts. Matrix (a) represents the votes by the decision makers against contradictory business rules. Matrix (b) shows their respective mathematical representation. The application of equation 3 on matrix (b) results in matrix (c).

$$\text{(a)}= \begin{array}{c|c|c|c|c} & user1 & user2 & expert1 & expert2 \\ (p1r1, p1r2) & defeat & survive & defeat & survive \\ (p1r3, p1r2) & survive & defeat & defeat & defeat \end{array},$$

$$\text{(b)}= \begin{array}{c|c|c|c|c} & user1 & user2 & expert1 & expert2 \\ (p1r1, p1r2) & 1 & 0 & 1 & 0 \\ (p1r3, p1r2) & 0 & 1 & 1 & 1 \end{array}$$

taking α=0.4 and β=0.6 the calculations would be like

$$\text{(c)}= \begin{vmatrix} 0.4 * (1/2) + 0.6 * (1/2) = 0.20 + 0.12 = 0.32 \\ 0.4 * (1/2) + 0.6 * (2/2) = 0.20 + 0.60 = 0.80 \end{vmatrix}$$

The results in matrix (c) can be interpreted as follows: 0.32 means 32% of members consider that argument '*p1r1*' defeats its counter-argument '*p1r2*'. Therefore, counter-argument '*p1r2*' has priority over an argument '*p1r1*'. Similarly, 80% of members believe that argument '*p1r3*' defeats its counter-argument '*p1r2*' and results in the priority of argument '*p1r3*' over its counterargument '*p1r2*'.

7.6.3 Building and marking of dialectical trees

Once the conflict between an argument and its counter-argument is resolved and the preference between them has been saved in the knowledge base, the next step is to build the dialectical trees to establish dynamic priority between the argument and its counter-arguments. In Section 5.5.2, the methodology for the construction and marking of dialectical trees is discussed in detail. This methodology is used to establish the preference between the arguments and their counter-arguments using the decision makers's selected argumentation-driven conflict resolution strategy.

Considering the example in Section 7.3, any two instantiated DeLP rules are said to be in conflict if they support contradictory claims. In Illustration 7.4, the

business rule *'p1r1'* with claim *approveTravel* is in conflict with business rule *'p1r2'* with claim $\sim$ *approveTravel*. For the resolution of such conflicts, the argumentation-based technique is adopted. This process of argumentation starts when a business rule called an 'argument', i.e. *approveTravel*, contradicts (also known as 'attacked by') another business rule called its 'counter-argument', i.e. $\sim$ *approveTravel*. Counter-arguments are also arguments which, in turn, may be attacked and result in the construction and marking of dialectical trees as shown in Figure 7.12. The defeated arguments are represented by the letter D and undefeated arguments by letter U.

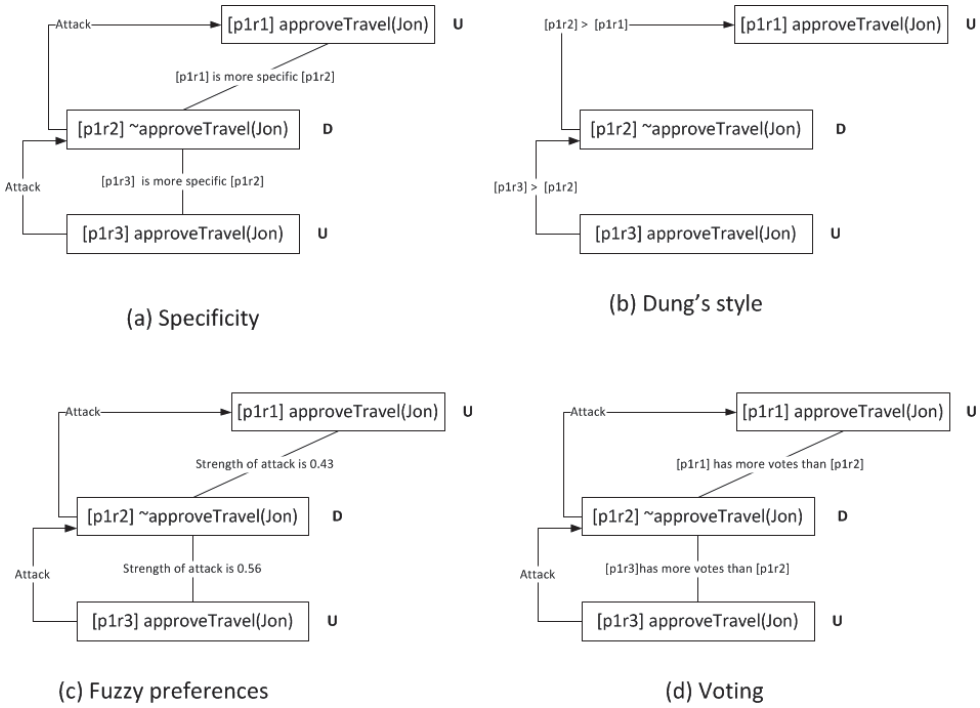


Figure 7.12: Marked dialectical trees considering different argumentation-driven conflict resolution strategies

7.7 Graphical representation of business process maps

Once the hybrid reasoning process is complete, KR@PMD exploits the functionality of *information and knowledge integration module* of the logic-based framework to integrate the information obtained from hybrid reasoning and displays it to the decision maker. This process involves the following steps:

(a) Generation of a business process map

During this step, the arguments present in the Activated Process set are linked in the form of a map (i.e. reasoning chain) known as a business process map. The construction of reasoning chains was explained in Section 5.6.1.

(b) Graphical representation of the business process map

Once the process map has been generated, the next step is to provide its graphical representation to the decision maker. Figures 7.13 and 7.14 provide a graphical flow diagram of business processes 1 and 2 as discussed in Section 7.4. The elements of a task specified as premises of a business rule are depicted in the business process map[3] as follows:

- the decision elements are diamond-shaped and light blue in colour;
- the resource elements are oval-shaped and pink in colour;
- the action elements are rectangular-shaped and blue in colour;
- the data-items are rectangular-shaped and green in colour;
- the situation elements are double octagon-shaped and maroon in colour. These elements are also annotated with the priority information of one over the other, and
- the constraints are rectangular-shaped and red in colour

Similarly, relationships between the different tasks in a process can be easily identified. For example, the '$\longrightarrow$' arrow in a business process map represents a strict business rule, whereas the '$\multimap$' arrow represents defeasible information in a process. The arrow '$\diamond\!\!-\!\!-$ joins the two elements of a business process where the output of the first element becomes the input for the second element in a business process.

[3]The claim of a business rule is represented by a grey, rectangular-shaped box.

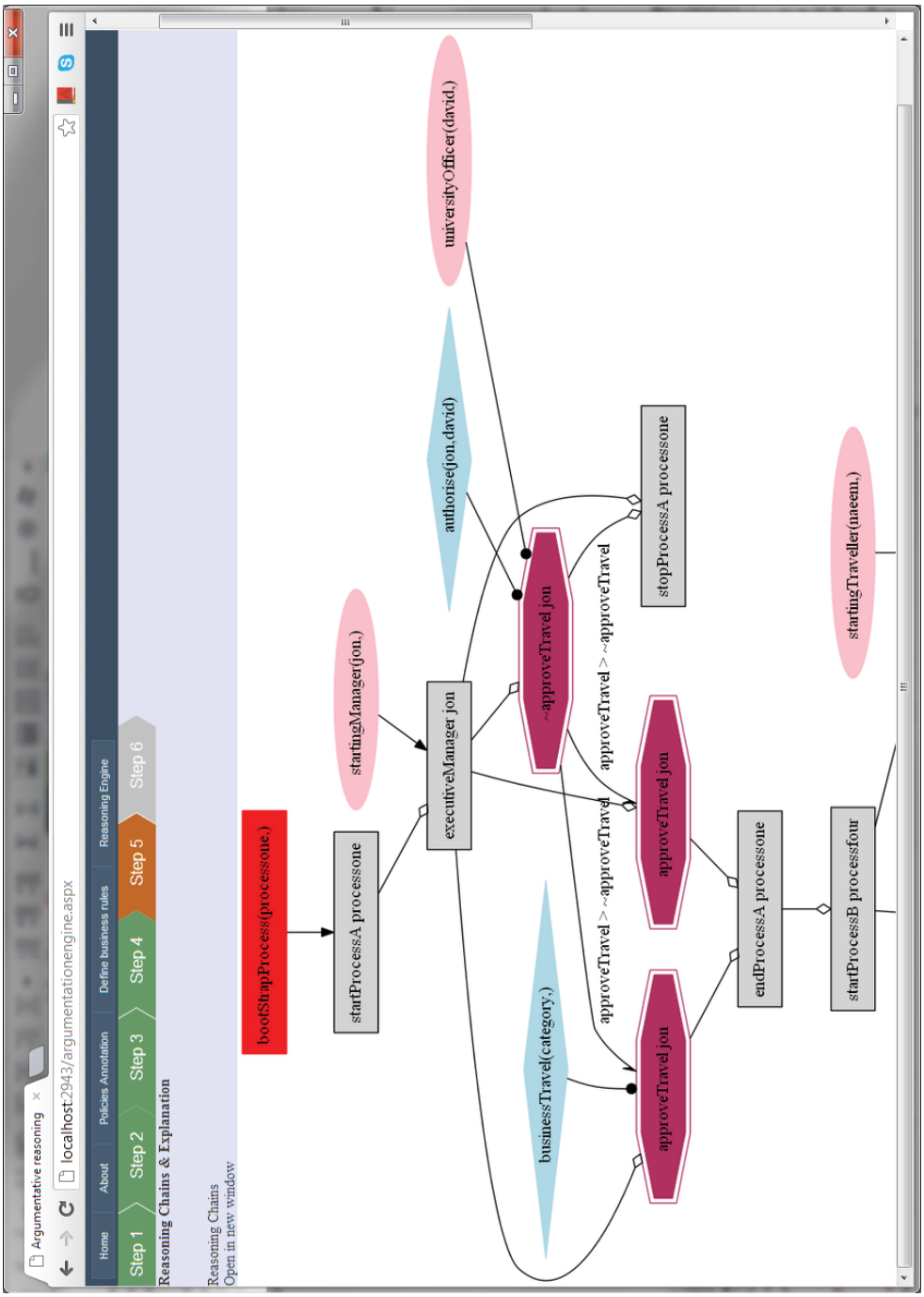


Figure 7.13: Graphical representation of business process map of business process 1 by of KR@PMD

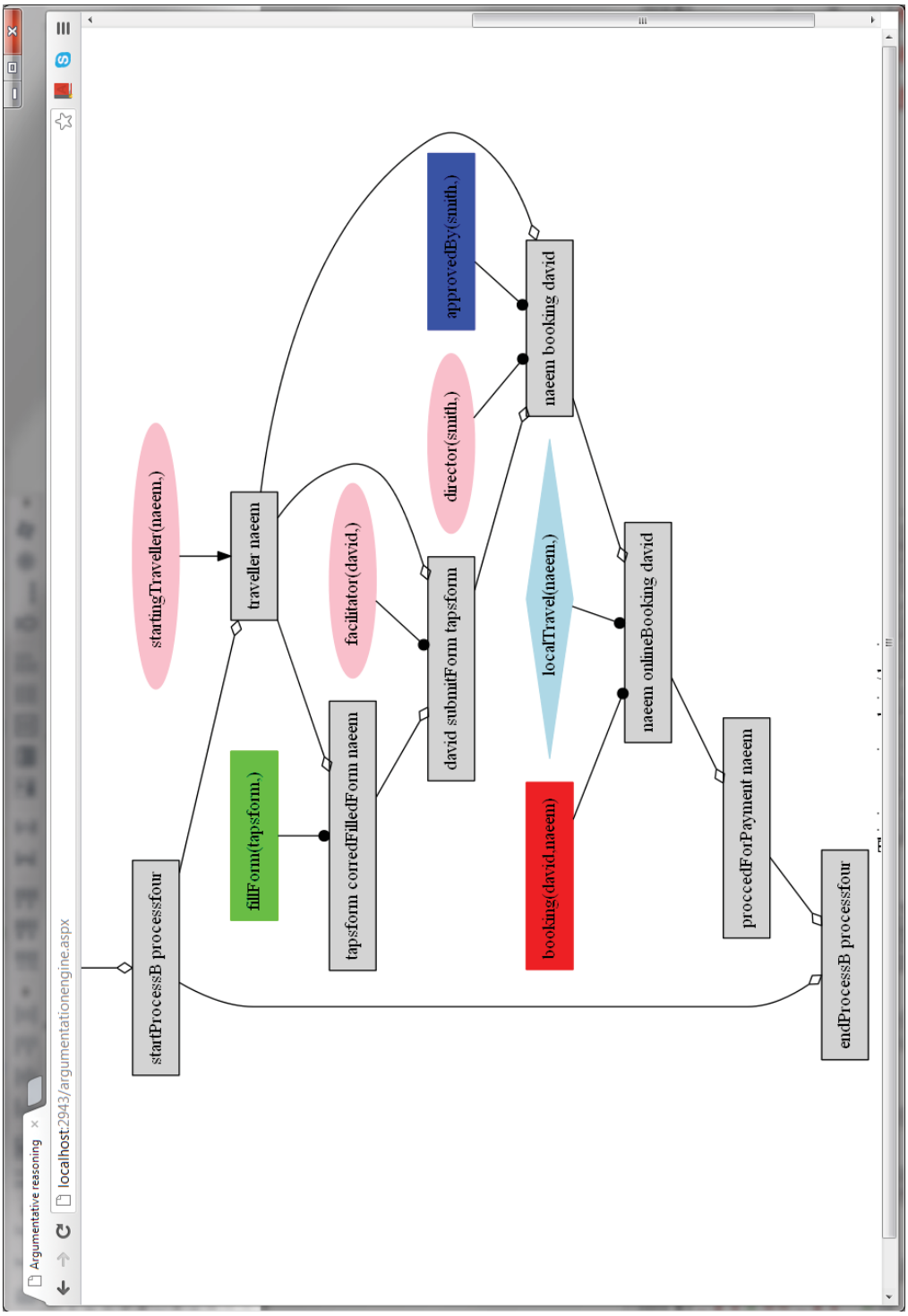


Figure 7.14: Graphical representation of business process map of business process 2 by of KR@PMD

The graphical representation of the business process map generated from unstructured business policies by KR@PMD enables decision makers in the

following ways:

- Identify the important tasks in a business process and how they follow one another as defined by the business policy. They can identify and resolve conflicts that may be present in current business policies so that they can ensure either successful collaboration between departments located within the enterprise and/or in other enterprises or they can ensure they are in accordance with the legal regulations to which the enterprises should comply.

- The business process map can also be used as a validation tool to ensure compliance of operational business processes with business policies. The decision maker can forward a printout of the generated business process map to the departments in order to check the compliance of operational business processes with the enterprise's business policies. The manager of each department can easily identify if a contradictory situation in a business process exists, e.g.,'*Authority to approve travel*' and how it should be resolved. He can further learn that Process 2 can be started only if travel approval is given to 'Jon', otherwise, Process 1 will result in the stopping phase and no further processing will take place.

It is important to note that during data-driven reasoning, it is possible that some business rules (i.e. representing different tasks in a process) may not be activated. As a result, the final business process map may not contain those tasks as part of the process. Therefore, a direct relationship exists between information in the working memory and the business rules in the rule-base. If all the information required by the business processes is introduced into the working memory before data-driven reasoning, more business rules will be activated during data-driven reasoning which results in a more detailed business process map.

7.8 Conclusion

The need for tools to check the compliance of operational business processes with the enterprise's business policies has increasingly become a subject of interest for businesses. In this chapter, this need was addressed using techniques from the field of knowledge representation. A conceptual framework for process discovery

from business policies was proposed by considering their defeasible nature. A Web-based DSS was developed that takes business policies as the starting point and, through a series of steps, generates a graphical business process map. This graphical map enables an operational manager to check the compliance of operational business processes with business policies. The availability of such tools will help enterprises to develop their business policies in compliance with various regulatory authorities and will assist in seamless and successful business mergers.

The contributions of this chapter are as follows:

(a) A semantic model-based process ontology based on process modeling concepts.

(b) The extension of a business rule language for business process modeling.

(c) Different argumentation-driven conflict resolution strategies.

(d) Generation and graphical representation of process maps in order to use them as validation tools to check the compliance of business policies with operational business processes.

Chapter 8 - Validation and Evaluation of GF@SWA

8.1 Introduction

In the previous chapters, the working of the generic defeasible logic programming-based framework (GF@SWA) to represent, reason and integrate incomplete and/or contradictory information (both structured and unstructured) in different Semantic Web applications for EII and EKI was discussed. In this chapter, the effectiveness of GF@SWA is demonstrated by developing three different Semantic Web applications i.e. Web-based IDSSs, using GF@SWA to provide decision support in different scenarios. The developed Web-based IDSSs provide proof of concept as explained in Section 3.6 on research methods. It constitutes the test stage of the system development approach as an IS research methodology, namely, functionality validation and evaluation, using a prototype system realization.

This chapter is organized as follows: Section 8.2 provides the general description of the tool and technologies used for the development of Web-based IDSSs. Section 8.3 lists the objectives for the development of the proposed GF@SWA. Section 8.4 provides the static structure and dynamic behaviour of the proposed GF@SWA. Section 8.5 provides the functionality validation and demonstrates the working of GF@SWA. Section 8.6 concludes the chapter.

8.2 General description of the tools

The development of GF@SWA and Semantic Web applications using GF@SWA i.e. Web-based IDSSs, to address the requirements of different case studies is carried out with help of different tools and technologies as follows:

- Microsoft Visual Studio 2010 [1]

  This is a powerful Integrated Development Environment (IDE) that supports the development of Web-based applications in different programming languages. For the development of GF@SWA , the C sharp (C#) programming language is used. For the development of Web-based IDSSs using GF@SWA, Asp.net using C sharp language is used.

- NRuler [2]

  This is a fast production system library based on the RETE algorithm, written in C sharp. This library is extended for the development of the hybrid reasoning engine.

- QuickGraph [3]

  This provides generic directed/undirected graph data structures and algorithms for .NET. It also supports Graphviz [4] to render the graphs. It is used to generate the graphical representation of the reasoning results produced by the Web-based IDSS.

- DeLP Server

  This is an implementation of Defeasible logic programming (DeLP)(Garcia et al., 2007). It is used as a back-end server for the development of the hybrid reasoning engine.

- MySQL [5]

  This is the world's most popular open source relational database. MySQL server is used on the back-end of the Web-based IDSS to save the information in order to retrieve it later on.

- Protege [6]

  This is a free, open source ontology editor and knowledge-based framework. It is used for the design and development of the process ontology, as discussed in Section 7.4.1.

[1] http://www.microsoft.com/visualstudio/en-us
[2] http://nruler.codeplex.com/
[3] http://quikgraph.codeplex.com/
[4] http://www.graphviz.org/
[5] http://www.mysql.com/
[6] http://protege.stanford.edu/

8.3 Objectives for the development of GF@SWA

The objectives behind the development of GF@SWA to support monological argumentation in Semantic Web applications are as follows:

- To provide a set of classes that can be extended by Semantic Web applications to represent/translate incomplete and/or contradictory information, both structured and unstructured, and save it in the knowledge base.
- To provide a set of classes that can be extended by Semantic Web applications to perform hybrid reasoning over underlying information saved in the knowledge base. It should be able to construct a set of arguments from underlying information and provide a set of strategies to resolve the conflicts present among the arguments.
- To provide a set of classes that can be extended by Semantic Web applications to integrate the output of a hybrid reasoning engine i.e. information integration, and generate the graphical representation of the reasoning process and results.
- To provide a set of classes that can be extended by Semantic Web applications to integrate the output of different hybrid reasoning engines i.e. knowledge integration, published over the enterprise's intranet or Internet and generate the graphical representation of integrated knowledge.

In the next section, the important characteristics of the proposed framework which are exposed to Semantic Web applications are explained which helps them to achieve the abovementioned objectives.

8.4 Characteristics of the proposed GF@SWA

In this section, the basic characteristics and components of the proposed GF@SWA are discussed with the help of Unified Modeling Language (UML) by using the following diagrams:

- Structure diagrams

  Structure diagrams [7] define the static structure of a model which comprises different elements (packages, classes, objects and relationships among them) to create conceptual diagrams that represent concepts from the real world and the relationships between them.

[7] http://www.sparxsystems.com.au/resources/uml2_tutorial/

- Behavior diagrams

  Behavior diagrams [8] depict the variety of interaction and instantaneous states within a system to achieve a certain task over a certain period of time and observing the effects of an operation or event, including its results.

In the next sub-section, these diagrams are used to explain the characteristics of the proposed GF@SWA.

8.4.1 Structure diagrams

In this section, the static structure of proposed framework is explained with the help of Package diagrams. Figure 8.1 shows the package diagram of GF@SWA, which comprises different packages as follows:

1. The Communication package comprises a set of classes to query the back-end services such as the MySQL server and DeLP server. This package is created from scratch to achieve its required functionalities.

2. The Information representation package comprises a set of classes to represent/translate incomplete and/or contradictory information (both structured and unstructured) and save it in the knowledge base. This package is created from scratch to achieve its required functionalities.

3. Hybrid reasoning engine package comprises a set of classes to perform hybrid reasoning over information saved in the knowledge base. Figure 8.2 provides a set of sub-packages that come in the hybrid reasoning engine package. This package is created by extending the NRuler.

4. Information and knowledge integration package comprises a set of classes to perform information and knowledge integration. This package is created from scratch, however, it exploits the Quick Sharp library to generate the graphical representation of a reasoning chain.

[8] http://www.sparxsystems.com.au/resources/uml2_tutorial/

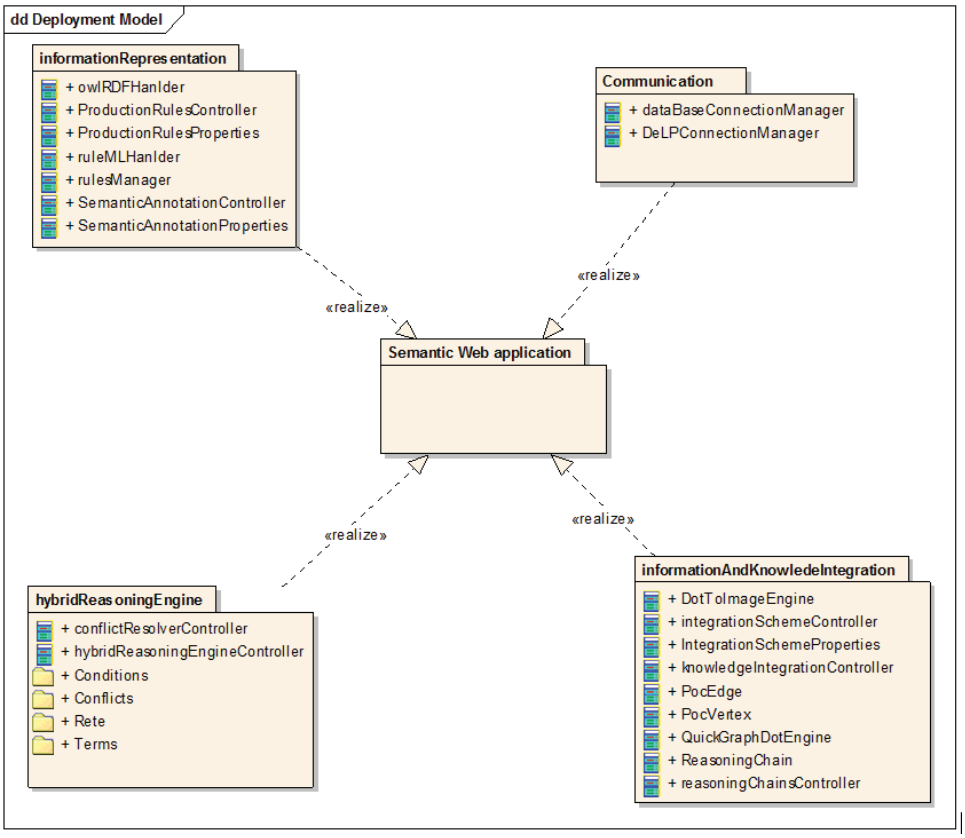


Figure 8.1: Package diagram of GF@SWA

In next sub-sections, each of these packages is discussed in detail.

8.4.1.1 Communication package

This package provides a set of classes to establish communication (i.e. read and write) between the GF@SWA and the back end servers such as the MySQL and DeLP server. The important classes in this package are:

- dataBaseConnectionManager provides a set of functions to GF@SWA in order to communicate with the MySQL server. The important functions it provides are read, write and delete information from the MySQL server.
- DeLPConnectionManager provides a set of functions for GF@SWA to communicate with the DeLP server. The important functions it provides are connect, disconnect, send query, receive query results and load the knowledge base.

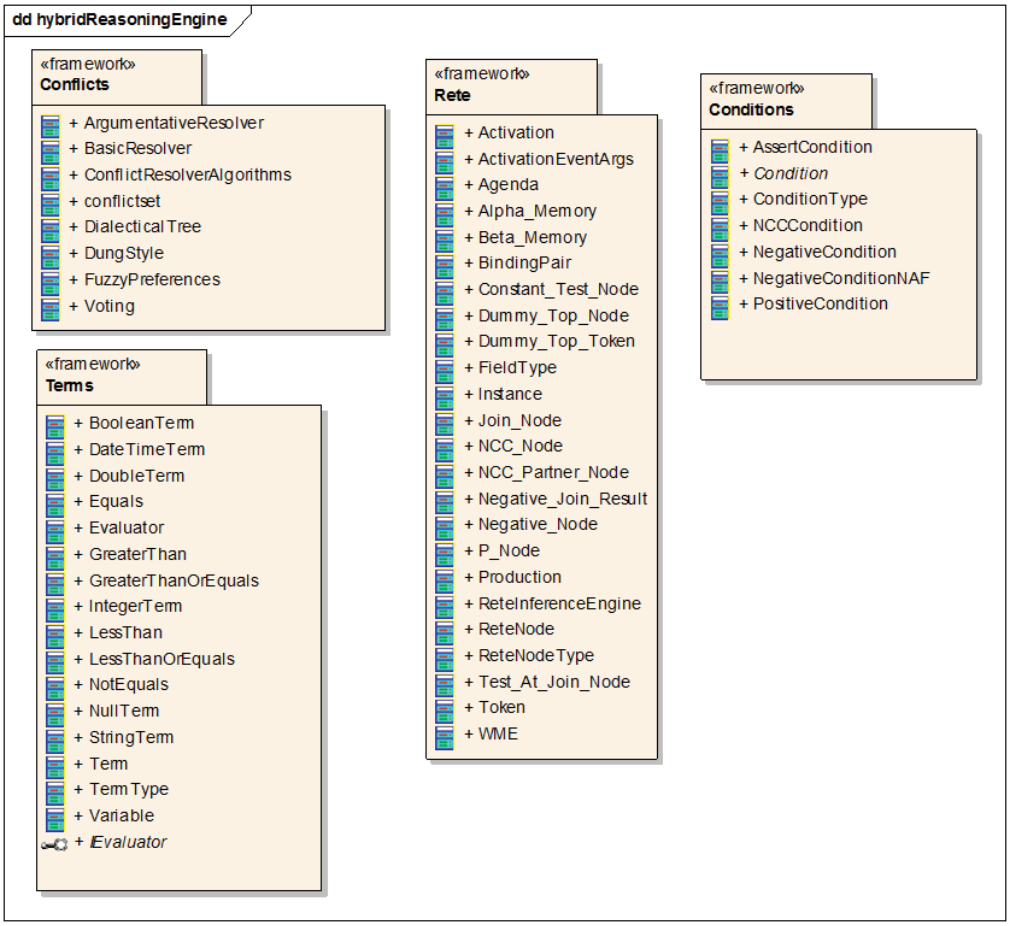


Figure 8.2: Hybrid Reasoning engine sub-packages

8.4.1.2 Information representation package

This package provides a set of classes to represent information (i.e. structured as well as unstructured) that is potentially incomplete and/or contradictory and saves it in the knowledge base. The important classes in this package are:

- The Semantic annotation controller (SA-CO) is responsible for handling the interactions of a Semantic Web application with GF@SWA for the semantic annotation of unstructured information and considers it in the decision-making process. It helps a Semantic Web application perform the following tasks:

  – download the decision maker's specified documents containing unstructured information and load the information in a Web-based form of Semantic Web application;

  – download the decision maker's specified domain ontology, read the concepts

from the ontology and display them on the Web-based form of the Semantic Web application, and

– save the information (i.e. annotated predicates) specified by the decision makers through the Web-based form of the Semantic Web application in the Semantic Annotation Properties Object (SA-PO). SA-CO iterates through information saved in SA-PO, validates the information and saves it in the MySQL server.

- The production rules controller (PR-CO) is responsible for handling the interactions of the Semantic Web application with GF@SWA for the specification of production rules. It helps Semantic Web applications perform the following tasks:

  – load the annotated predicates saved in the MySQL sever on the Web-based form of the Semantic Web application, and

  – save the information (i.e. production rules) specified by the decision makers through the Web-based form of the Semantic Web application in the Production Rules Properties Object (PR-PO). PR-CO iterates through information saved in PR-PO, validates the information and saves it in the knowledge base.

8.4.1.3 Hybrid reasoning engine package

This package provides a set of classes to perform hybrid reasoning over information saved in the knowledge base. The important classes in this package are:

- NegativeConditionNAF is a class located in the *Conditions* sub-package. It is used to represent incomplete information as a predicate in a production rule.

- The AssertCondition is a class located in the *Conditions* sub-package. It is used to represent both contradictory and non-contradictory predicates. A contradictory predicate contains '∼' at the start and the system reads this symbol to identify the contradictory predicates during reasoning process.

- Agenda is a class located in the *Rete* sub-package. It is responsible for the construction of the Rete network and forward chain reasoning. Its functionality has been extended for the construction of arguments during forward chain reasoning.

- ArgumentativeResolver is a class located in the *Conflicts* sub-package. It is responsible for argumentation-driven conflict resolution between arguments and their counter-arguments. It uses the following classes to achieve its functionality:

  - The ConflictResolverAlogirthm class provides a programming interface to attach different reasoning algorithms with the hybrid reasoning engines.

  - Fuzzy preferences, Dung's style and Voting are implementations of different conflict resolution algorithms that are used by the ArgumentativeResolver to resolve conflicts between arguments and their counter-arguments

  - DialecticalTree is class used to save the conflict resolution process information generated by the hybrid reasoning engine during goal-driven reasoning.

- The hybrid reasoning engine controller (HRE-CO) is responsible for handling the interactions of Semantic Web applications with GF@SWA to perform hybrid reasoning. It helps Semantic Web applications perform the following tasks:

  - load the facts and production rules in the Agenda object;

  - interact with the Agenda object to perform forward chain reasoning for the construction of the arguments;

  - interact with ConflictResolverController (defined next) to perform goal-driven reasoning and resolve conflicts using the decision maker's specified conflict resolution strategy, and

  - interact with ReasoningChainController (defined in the next sub-section) for the construction and graphical representation of the reasoning chains.

- The conflict resolver controller (CR-CO) is responsible for handling the interactions of HRE-CO with the DeLP Sever. It helps the Semantic Web applications in the following tasks:

  - provide different conflict resolution strategies for the decision maker and apply the selected strategy to resolve conflicts between arguments, and

  - interact with the DeLP server for the construction of dialectical trees during goal-driven reasoning.

8.4.1.4 Information and knowledge integration package

This package provides a set of classes to consider the output of hybrid reasoning engine/s and perform information and knowledge integration. The important classes in this package are as follows:

- The reasoning chains controller (RC-CO) is responsible for interacting with HRE-CO for information integration. It helps Semantic Web applications perform the following tasks:
  - link the arguments forwarded by HRE-CO in the form of a reasoning chain, and
  - generate a graphical representation of the reasoning chain.
- The integration scheme controller (IS-CO) is responsible for handling the interactions between a Semantic Web application and the HRE-CO for the valuation of reasoning chains. It helps Semantic Web applications perform the following tasks:
  - save the argumentation scheme information specified by the decision maker via a Web-based form in the Integration Scheme Properties Object (IS-PO);
  - perform the valuation of reasoning chains with the help of RC-CO, and
  - return the valuation results of each reasoning chain to the Semantic Web application.
- The knowledge integration controller (KI-CO) is responsible for handling the interactions between the Semantic Web application and HRE-CO for the generation of integrated recommendations space i.e. knowledge integration. It helps Semantic Web applications perform the following tasks:
  - retrieve the valued reasoning chains with the help of RC-CO and forward them to HRE-CO for identification and the resolution of conflicts among the arguments;
  - construct new arguments from existing ones, and
  - generate a graphical representation of the integrated recommendations space with the help of RC-CO and return the results to the Semantic Web application.

8.4.2 Behaviour diagrams

In this section, Sequence diagrams are used to depict the work flow or sequence of steps involved in an activity over a period of time using messages passed between elements. These messages correspond to Class operations and behavior in the software model. Sequence diagrams are used to define the behaviour of GF@SWA in the following activities:

1. Semantic annotation of unstructured information.
2. Production rules specification.
3. Hybrid reasoning and the generation of reasoning chains i.e. information integration.
4. Integration of output of different hybrid reasoning engines and the generation of reasoning chains i.e. knowledge integration.

In the following sub-sections, the sequence diagram of each activity is discussed in detail.

8.4.2.1 Sequence diagram for semantic annotation of unstructured information

Figure 8.3 shows the sequence of steps performed for the semantic annotation of unstructured information by a Semantic Web application. The steps are as follows:

1. The Semantic Web application displays a Web-based form to the decision maker where he can enter the URL/paths for a document that contains unstructured information and for a domain ontology.
2. The Web-based form takes the information specified by the decision maker in the previous step and forwards it to the Semantic Annotation Controller Object (SA-CO). The SA-CO reads the specified files i.e. a document and an ontology, and loads the information on the Web-based form.
3. The decision maker extracts the information/concepts from the loaded information and annotates them with the concepts defined in the domain ontology. The Web-based form saves the annotated predicate in the Semantic Annotation Properties Object (SA-PO). This is an iterative process controlled by the decision maker.

4. The Semantic Web application forwards the SA-PO to SA-CO to save the annotated predicates.

5. The SA-CO reads the annotated predicates from the *List* defined in SA-PO and saves the predicates in the MySQL server.

6. The SA-CO displays the message 'information saved successfully' on the Web-based form.

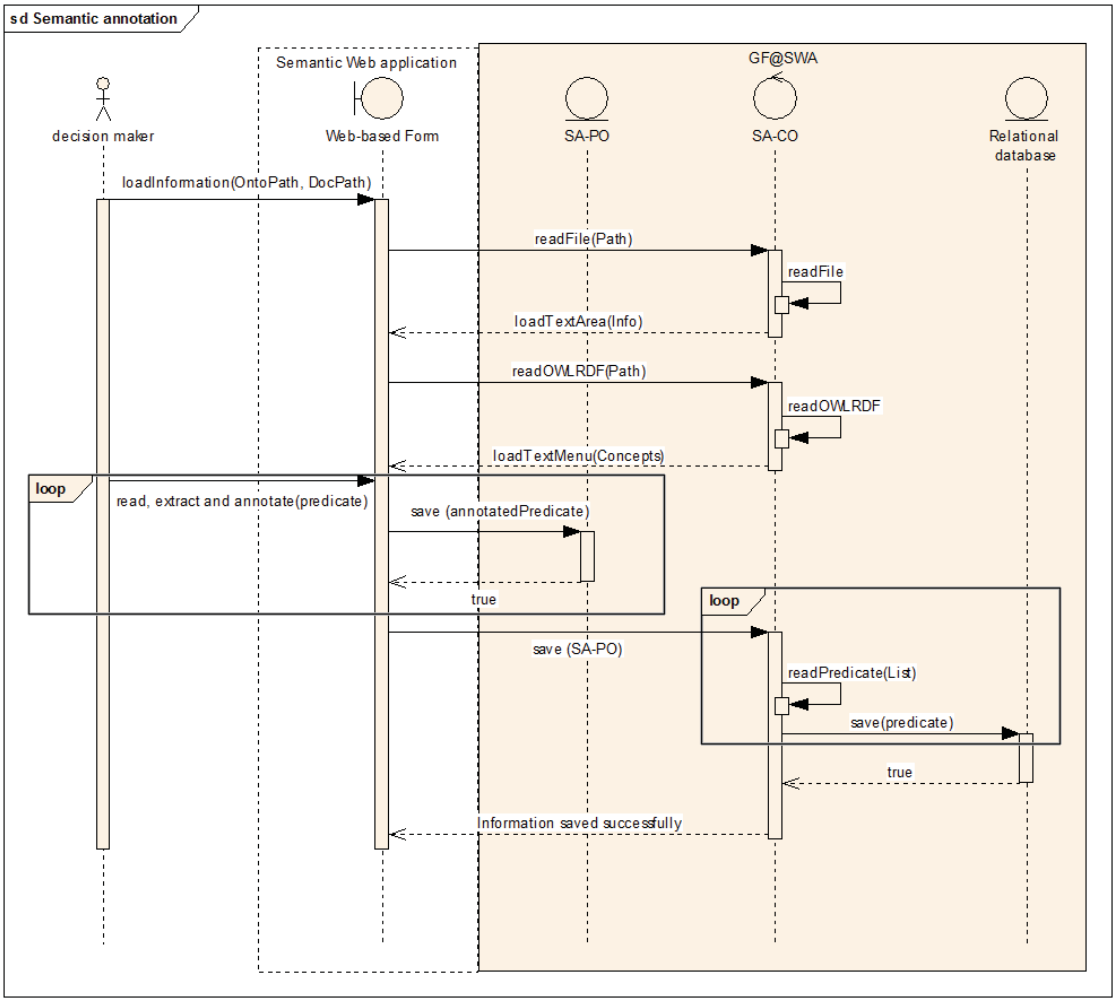


Figure 8.3: Sequence diagram for the semantic annotation of unstructured information

8.4.2.2 Sequence diagram for production rules specification

The GF@SWA provides two ways for the specification of production rules as follows:

1. An automated process performed by the RuleML translator that translates the

production rules specified in RuleML format to DeLP format and saves them in the knowledge base.

2. A non-automated process where the specification of production rules is carried out from scratch by a decision maker.

The first method of specifying production rules is an automated process performed by a single class, therefore its sequence diagram is not represented. Figure 8.4 shows the sequence diagram of the second method i.e. the specification of production rules by a decision maker. The steps performed during this process are as follows:

1. The decision maker interacts with the Semantic Web application to open the production rules specification form.

2. The Semantic Web application loads and displays the Web-based form for the decision maker. During form loading, the Semantic Web application forwards the request to PR-CO to load the annotated predicates. PR-CO reads the annotated predicates from the MySQL server and loads the information on the Web-based form.

3. The decision maker selects/fills in the production rule specifications (i.e. select premises, rule type and a conclusion) and the Web-based form saves this information in PR-PO. This is an iterative process, controlled by the decision maker.

4. Once the decision maker finishes with the specification of production rules, the Web-based form forwards the PR-PO to PR-CO to save the production rules in the knowledge base. The PR-CO reads the production rules from PR-PO, and checks if the production rules *label* already exist in the knowledge base. If the query returns false, then the production rule is saved in the knowledge base. Otherwise, the decision maker is informed of the problem and is asked to rectify it.

5. Once all the production rules are saved in the knowledge base, PR-CO returns a task completion message to the Semantic Web application and displays it on the Web-based form for the decision maker.

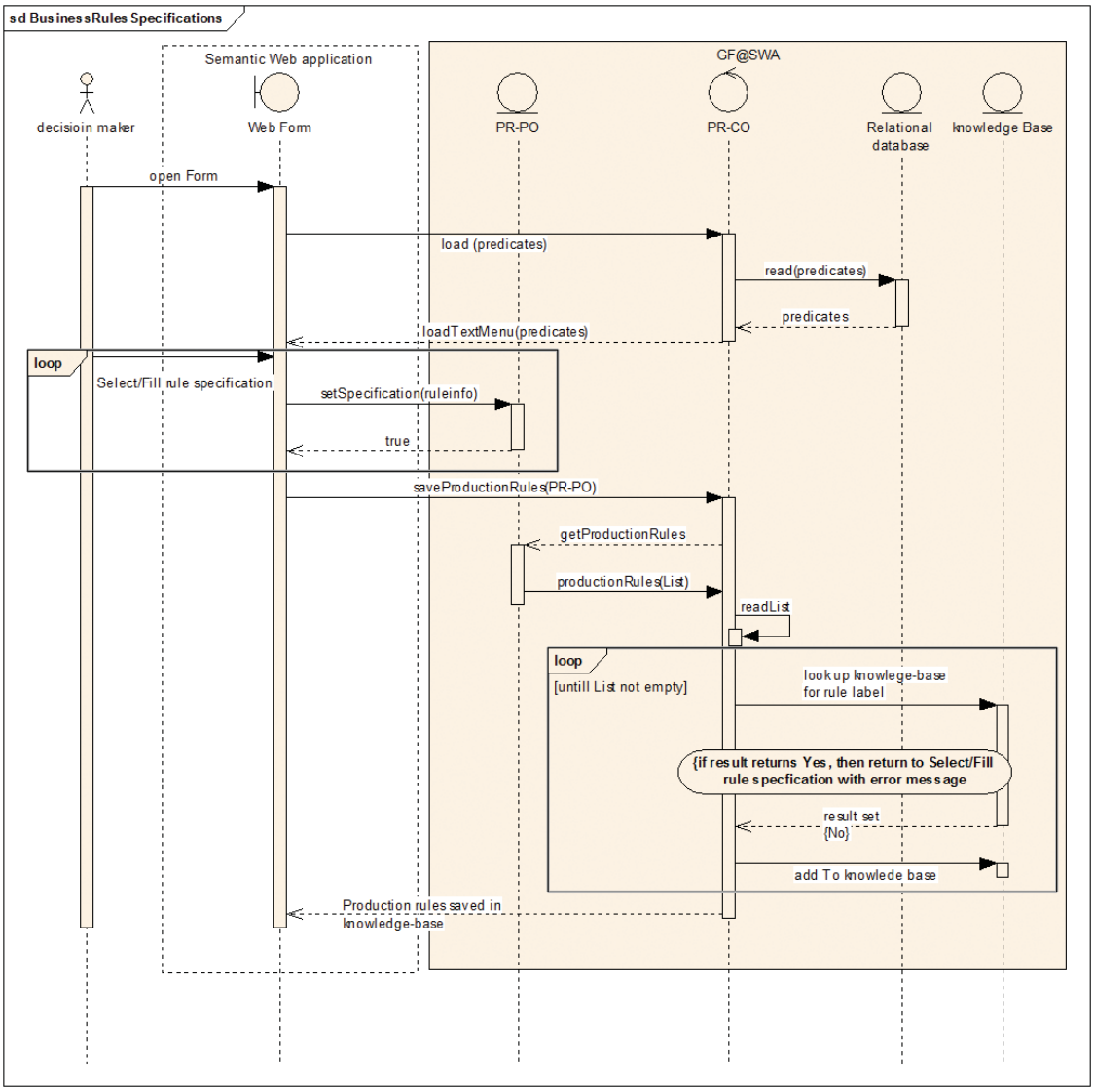


Figure 8.4: Sequence diagram for production rules specification

8.4.2.3 Sequence diagram for hybrid reasoning and the generation of graphical reasoning chains

Figure 8.5 shows the steps performed during hybrid reasoning and the generation of graphical reasoning chains. The sequence of steps involved in this process are:

1. Once all the required information has been specified and saved in the knowledge base, the hybrid reasoning engine is ready to perform hybrid reasoning.

2. The Web-based form of the Semantic Web application delegates the task of hybrid reasoning to the Hybrid Reasoning Engine Controller Object (HRE-CO). The HRE-CO loads the facts and production rules into the Agenda object.

3. Once the information from the working memory and rule base have been loaded into the Agenda object, the Semantic Web application delegates the request to HRE-CO to start reasoning.

4. The HRE-CO forwards the request to the Agenda object to start forward chain reasoning for argument construction. As a result, it returns a set of arguments and a conflict set to HRE-CO.

5. The HRE-CO returns the arguments and conflict set to the Web form of the Semantic Web application. Using the Web-based form, the decision maker selects a conflict resolution strategy to be used in order to resolve the conflicts between arguments and their counter-arguments.

6. The HRE-CO applies the strategy over the set of arguments involved in the conflicts during goal-driven reasoning. If the selected strategy needs some input from the decision maker, the HRE-CO takes the input via the Web form and considers this during the execution of the selected strategy.

7. Once the conflicts have been resolved, HRE-CO forwards the resulting arguments to the Reasoning Chains Controller Object (RC-CO) to generate the reasoning chains and returns their graphical representation to HRE-CO which then displays them back on the Web-based form for the decision maker.

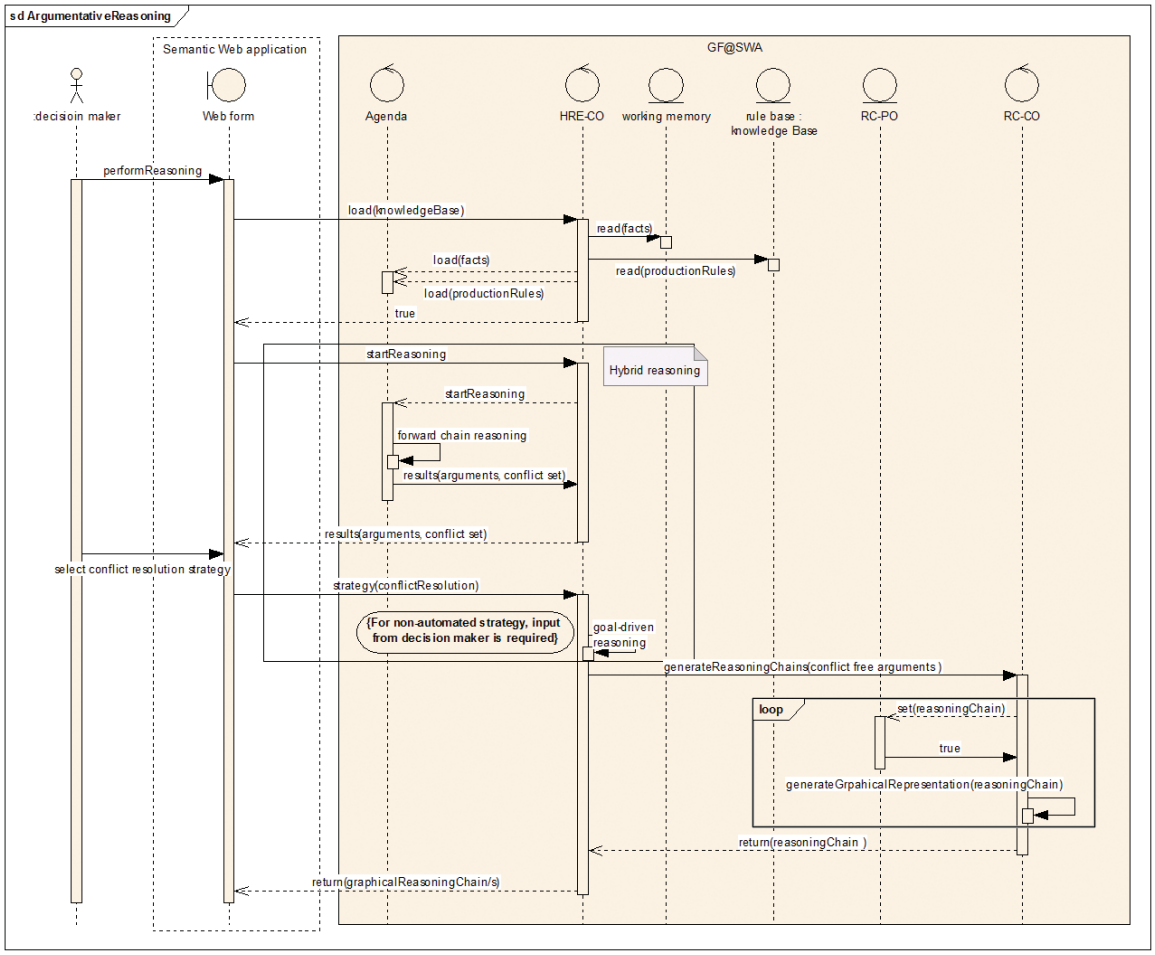


Figure 8.5: Sequence diagram for hybrid reasoning and the generation of graphical reasoning chains

Figure 8.6 shows the sequence diagram of the steps performed by HRE-CO during hybrid reasoning. The steps are:

1. Once the production rules and facts have been loaded in the Agenda object, the HRE-CO forwards the request to the Agenda object to start forward chain reasoning.

2. The Agenda object first builds the Rete network and then performs forward chain reasoning for the construction of arguments. The arguments constructed are then saved in the Arguments Set object. Once forward chain reasoning is complete, the arguments set is returned to HRE-CO.

3. To resolve conflicts between arguments using goal-driven reasoning, HRE-CO forward the Argument Set object to the Conflict Resolver Control Object (CR-CO). CR-CO retrieves the conflict resolution strategy selected by the

decision maker from the Conflict Resolver Properties Object (CR-PO) and applies it for the resolution of conflicts. During goal-driven reasoning, the marked dialectical trees are saved in the Dialectical Tree Properties Object (DT-PO). On the completion of goal-driven reasoning, the conflict-free arguments set is returned to HRE-CO.

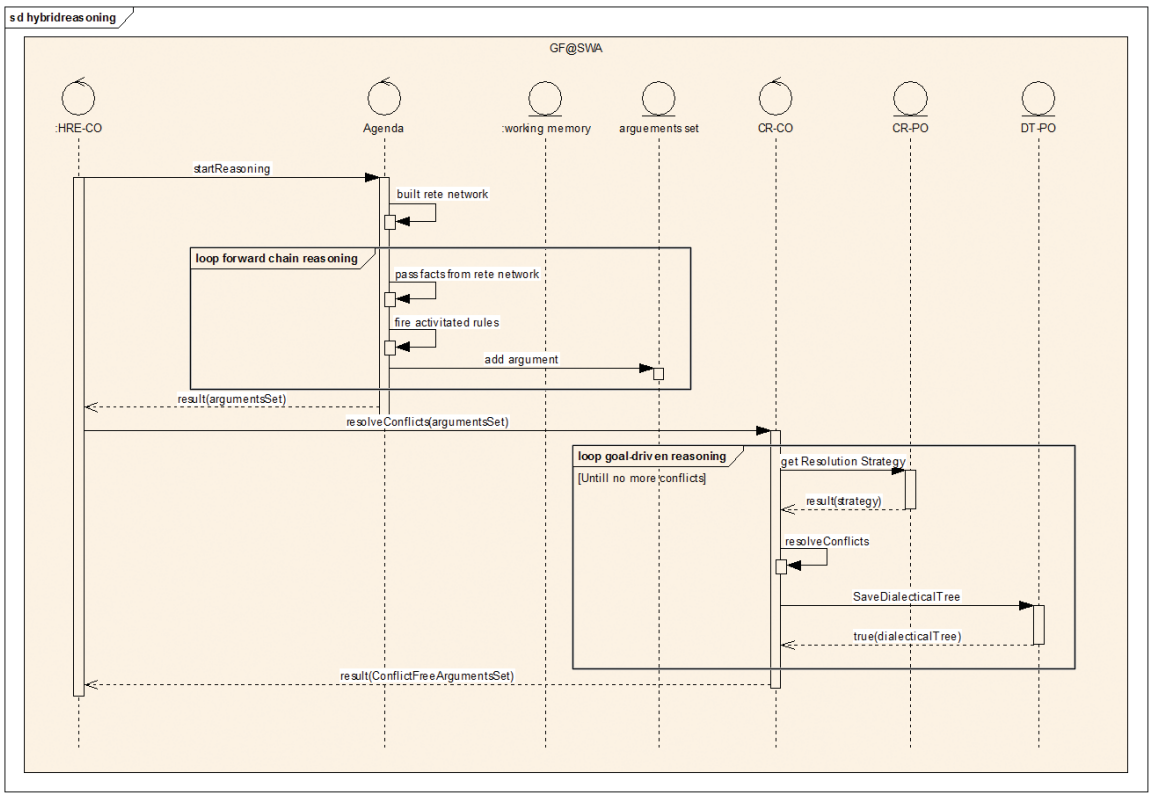


Figure 8.6: Sequence diagram represents the steps performed by hybrid reasoning engine

8.4.2.4 Sequence diagram for knowledge integration

Figure 8.7 shows the sequence diagram for knowledge integration which involves the following series of steps:

1. The Semantic Web application displays a Web-based form to the decision maker to define an integration scheme. The decision maker selects/fills in the integration scheme information. The Web form saves the integration scheme information in the Integration Scheme Properties Object(IS-PO).

2. Once the information is saved in IS-PO, the Web-based form forwards the request to Integration Scheme Controller Object (IS-CO) for the valuation of the reasoning chains. The IS-CO pulls the reasoning chains with the help of

RC-CO and applies the integration scheme on each reasoning chain and saves the valued reasoning chain. This is an iterative process which continues until all the reasoning chains are processed.

3. Once the valuation of the reasoning chains is complete, the decision maker generates the integrated recommendation space. During this process, the Web-based form forwards the request to the Knowledge Integration Data Control Object (KI-CO). KI-CO pulls the reasoning chains with the help of RC-CO and with the help of HRE-CO, identifies and resolves the conflicts among the reasoning chains. Then it generates new arguments and forwards the argument set to RC-CO to build the integrated recommendation space.

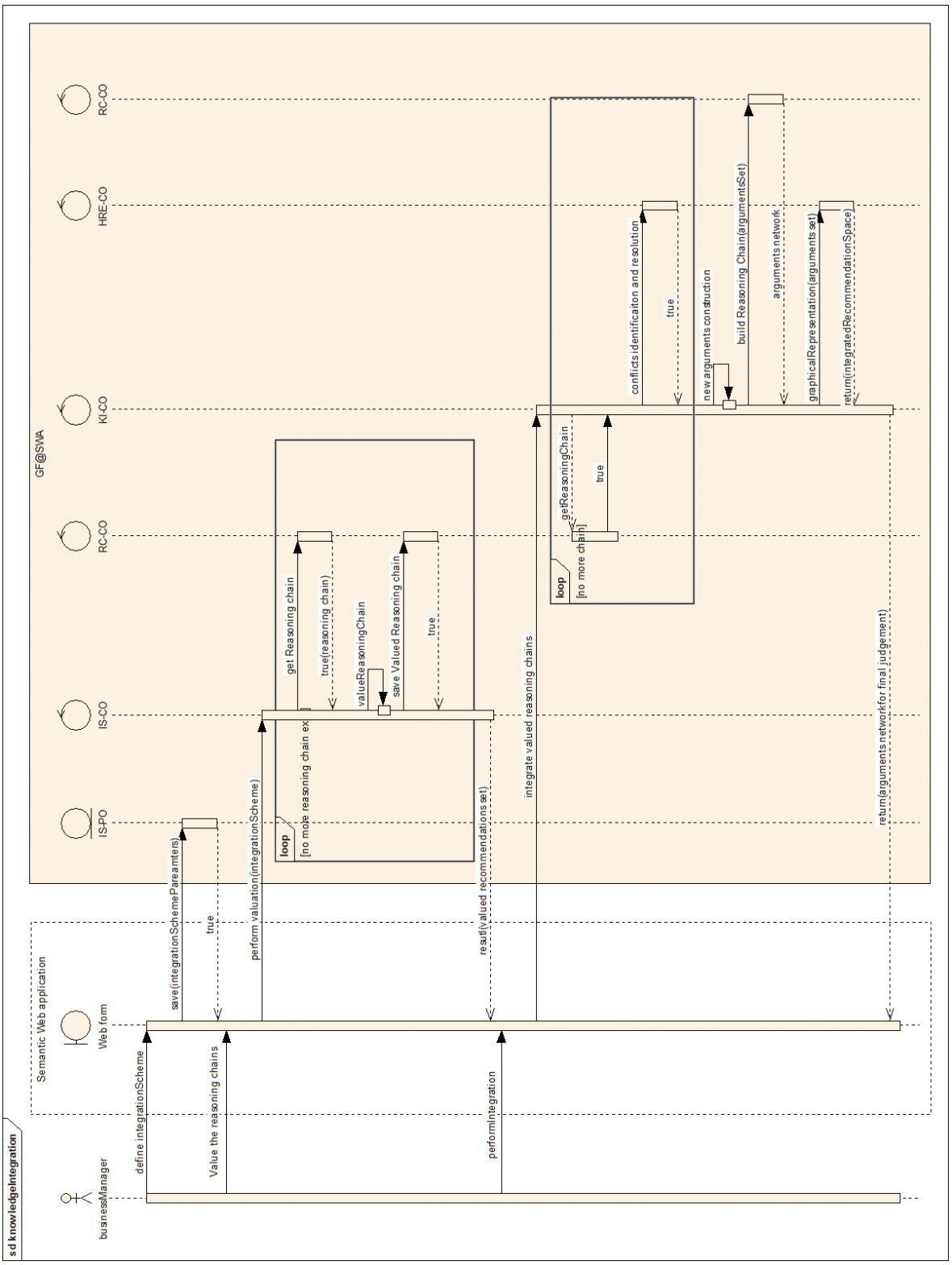


Figure 8.7: Sequence diagram knowledge integration

8.5 Functionality validation and feature evaluation of GF@SWA

Software functionality validation (Andriole, 1986) results in confirmation by examination and the provision of objective evidence that software specifications conform to the decision maker's needs, and that the particular requirements implemented through the software can be consistently fulfilled. Software evaluation (Anderson, 1989) involves the assessment of the software by comparing its characteristics with existing software in that domain. During software feature-based evaluation, each piece of software is assessed against the features defined in a matrix, known as the feature matrix. Such evaluation of software helps to categorise the existing software and identify the one that addresses the needs of the decision maker in the best possible manner.

In order to validate and evaluate the proposed GF@SWA for monological argumentation support in Semantic Web applications, the following steps were performed:

- Three Semantic Web applications were developed using GF@SWA. The validation of the functionality of each application provides the validation of the GF@SWA. The developed applications are:
  - Web@IDSS : Section 8.5.1 provides the validation of Web@IDSS by considering the functional requirements identified in the case study discussed in Section 5.2.
  - Web@KIDSS: Section 8.5.3 provides the validation of Web@IDSS by considering the functional requirements identified in the case study discussed in Section 6.2.
  - KR@PMD: Section 8.5.5 provides the validation of Web@IDSS by considering the functional requirements identified in the case study discussed in Section 7.3.
- Feature evaluation was performed on each developed Semantic Web application identified above with the existing contemporary applications. Such evaluation of each application in turn provides the evaluation of GF@SWA.

In the next sub-sections, the functionality validation and features evaluation of each Semantic Web application developed using GF@SWA is discussed.

8.5.1 Functionality validation of Web@IDSS

Section 5.3.2 provides the sequence of steps performed by Web@IDSS to consider the incomplete and/or contradictory information which exists within an enterprise or in other enterprises and assists the decision maker in the decision making process. In the next sub-section, the aims to be met during of the validation of the functionalities provided by Web@IDSS are listed.

8.5.1.1 Aims for the development of Web@IDSS

In this section, the aims for the development of Web@IDSS, as previously identified in Section 5.2, are listed as follows:

1. a Web-based form to download the structured information representing the public policies of an enterprise published over the enterprise's intranet or on the Internet;
2. capability to download feedback or reviews from other customers about the products and services of the enterprise;
3. a Web-based form to define the business requirements in the form of production rules, and
4. ability to perform reasoning over underlying information which may be incomplete and/or contradictory, automatically resolve conflicts and provide a graphical representation of the reasoning process and the reasoning result in order to make them easily understandable by non-technical persons.

In order to validate the functionality of Web@IDSS, in the next subsection, the working of Web@IDSS is described with an example of the case study discussed in Section 5.2 to achieve the aims discussed above.

8.5.1.2 Working of Web@IDSS

Figure 8.8 depicts a Web-based form of Web@IDSS displayed for the decision maker to import the structured information representing public business rules or polices of an enterprise defined in RuleML format over the Web. Appendix A.1 presents the public policies of an enteprise BigW specified in RuleML format. Using the Web-based form, the decision maker i.e Mr. David, downloads the business rules or policies of the enterprise/s he has identified for consideration in the decision making process.

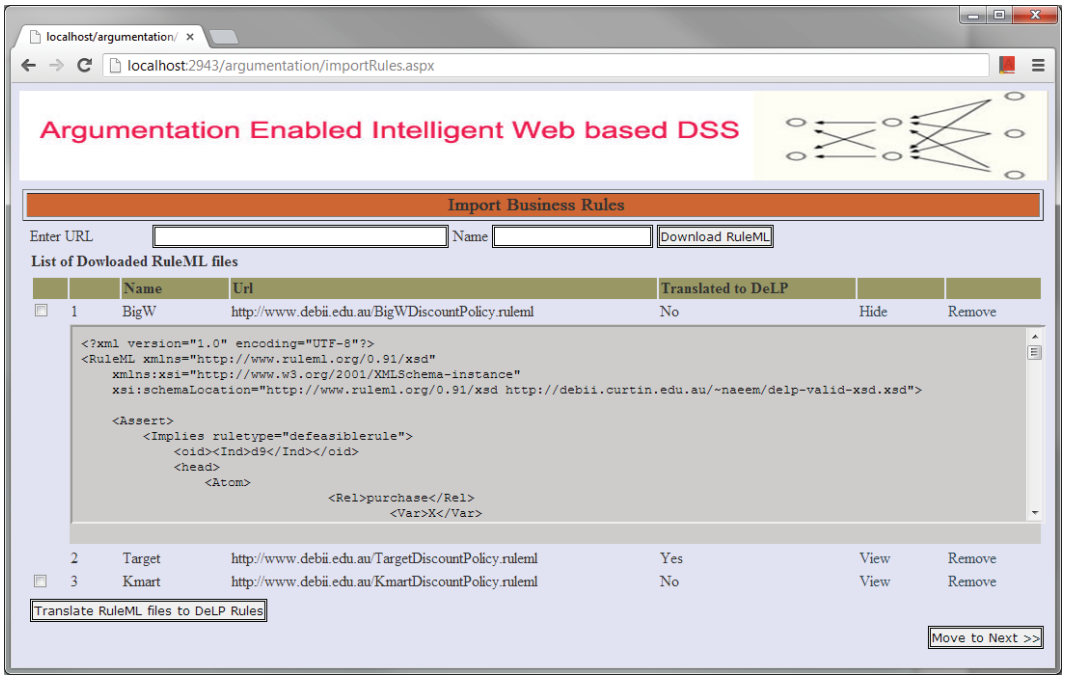


Figure 8.8: Web-based form of Web@IDSS to download RuleML files

Once the decision maker has finished downloading the RuleML file/s, Web@IDSS needs to translate RuleML files to DeLP format in order to consider them during hybrid reasoning. Figure 8.9 depicts the Web-based form where the decision maker can select the RuleML files he wants to consider during the decision-making process. Web@IDSS translates the selected files and saves the translated information in form of production rules in the rule base. The production rules are used by the hybrid reasoning engine during the decision-making process.

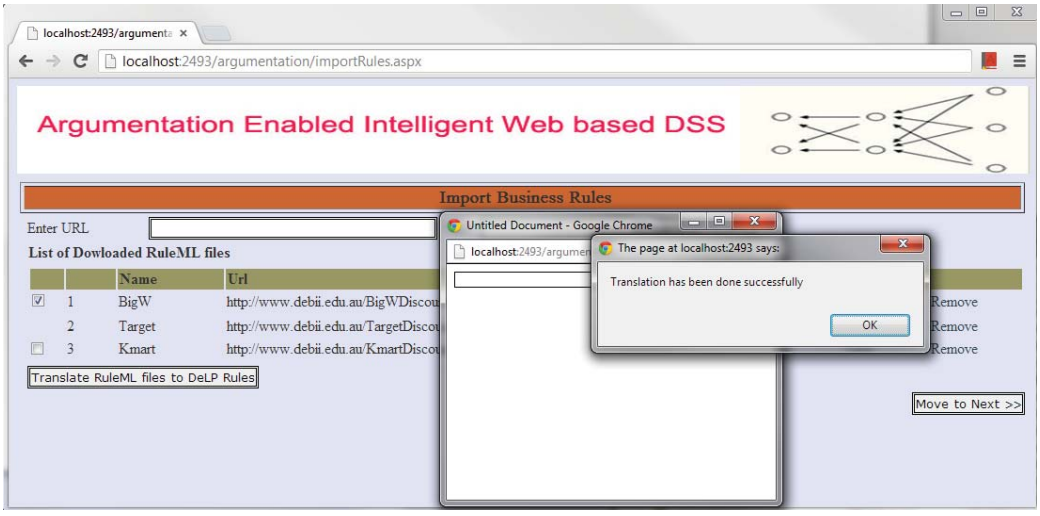


Figure 8.9: Web-based form of Web@IDSS for translation of business rules from RuleML to DeLP format

To consider facts during the decision-making process, Web@IDSS provides a Web-based form as depicted in Figure 8.10 that allows the decision maker to download the feedback information about products and services, serialized in OWL/RDF format and translate it to DeLP facts and save them in the working memory. Appendix A.2 presents feedback information on the raw material provided by BigW in OWL/RDF format. The translated information is used for the activation of production rules saved in the knowledge base by the hybrid reasoning engine during the decision-making process.

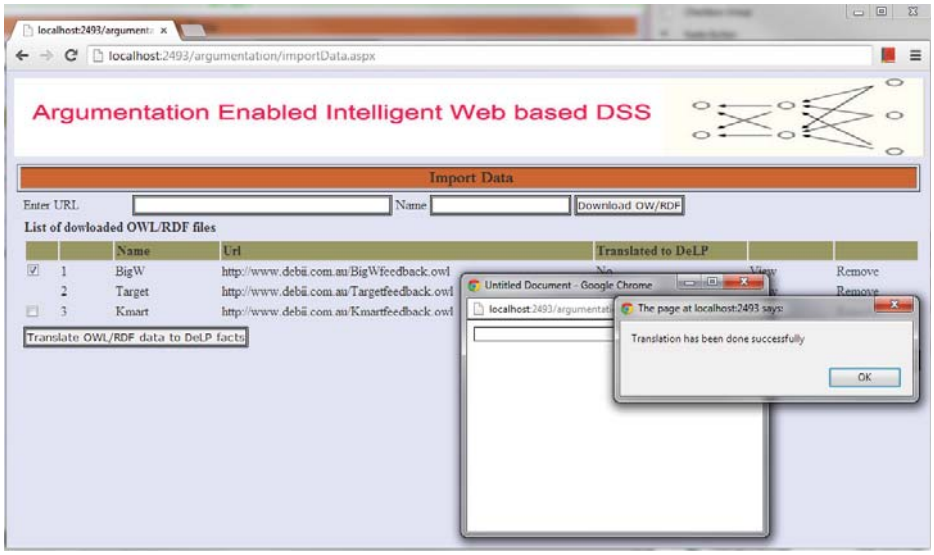


Figure 8.10: Web-based form of Web@IDSS for translation of feedback specified in OWL/RDF format to DeLP format

Once the decision maker is finished downloading and translating the information identified for the decision-making process, he defines his own business requirements that needs to be considered by the Semantic Web application during the decision-making process, using the Web-based form as depicted in Figure 8.11. The decision maker creates a production rule by assigning it a name, inference type, set of premises and a conclusion and saves it in the rule base. He can also view the list of rules created and can edit and delete them. The decision maker can define certain facts to be saved in the working memory. The production rules and facts defined by the decision maker are considered by the hybrid reasoning engine during the decision-making process along with information already saved in the knowledge base i.e. translated business policies of BigW and feedback about its products and services.

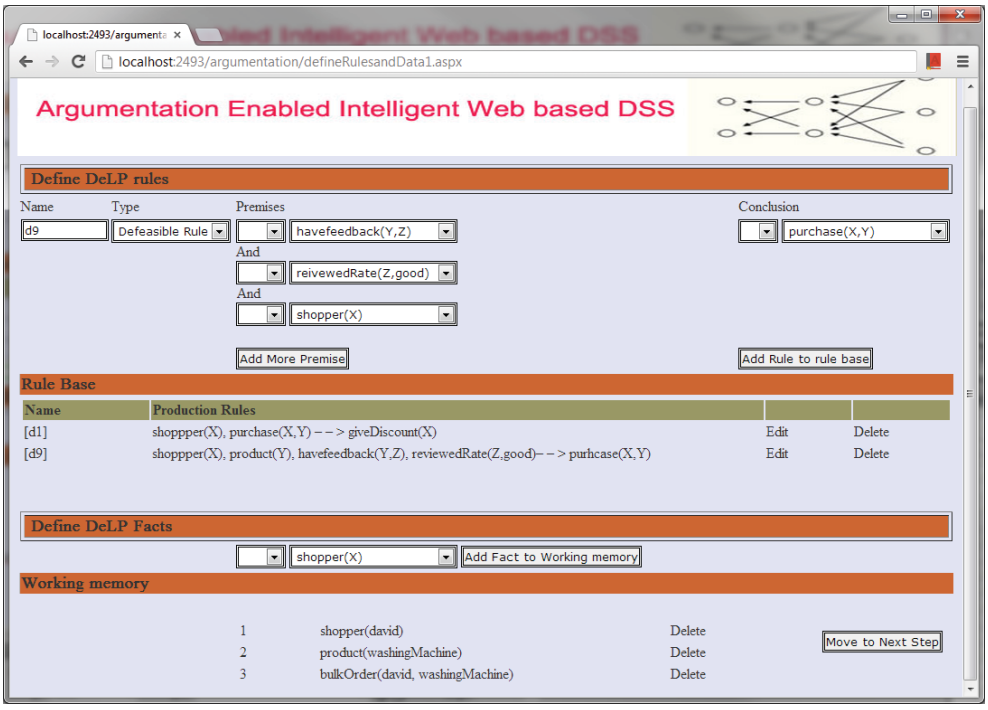


Figure 8.11: Web-based form of Web@IDSS to define production rules and facts

The next step performed by Web@IDSS after importing and defining the production rules and facts is to undertake hybrid reasoning (i.e argumentation-driven reasoning). During hybrid reasoning, Web@IDSS takes into account all the production rules present in the knowledge base and the facts in the working memory to perform hybrid reasoning and displays the reasoning results to the decision maker in the form of a reasoning chain as depicted in Figure 8.12. The oval-shaped nodes in represent the facts and the rectangular nodes represent the claim of the production rules. The dotted lines from the nodes represent the defeasible inference and the solid lines represent the strict inference. The rectangular nodes without a border represent the undefeated arguments in a reasoning chain. If Web@IDSS produces more than one reasoning chain, the decision maker can use a category filter, such as 'Mixed Reasoning Chains', as shown in Figure 8.12 to display only those reasoning chains that belong to that category and Figure 8.13 display a dialectical tree against the query $\sim giveDiscount(david)$.

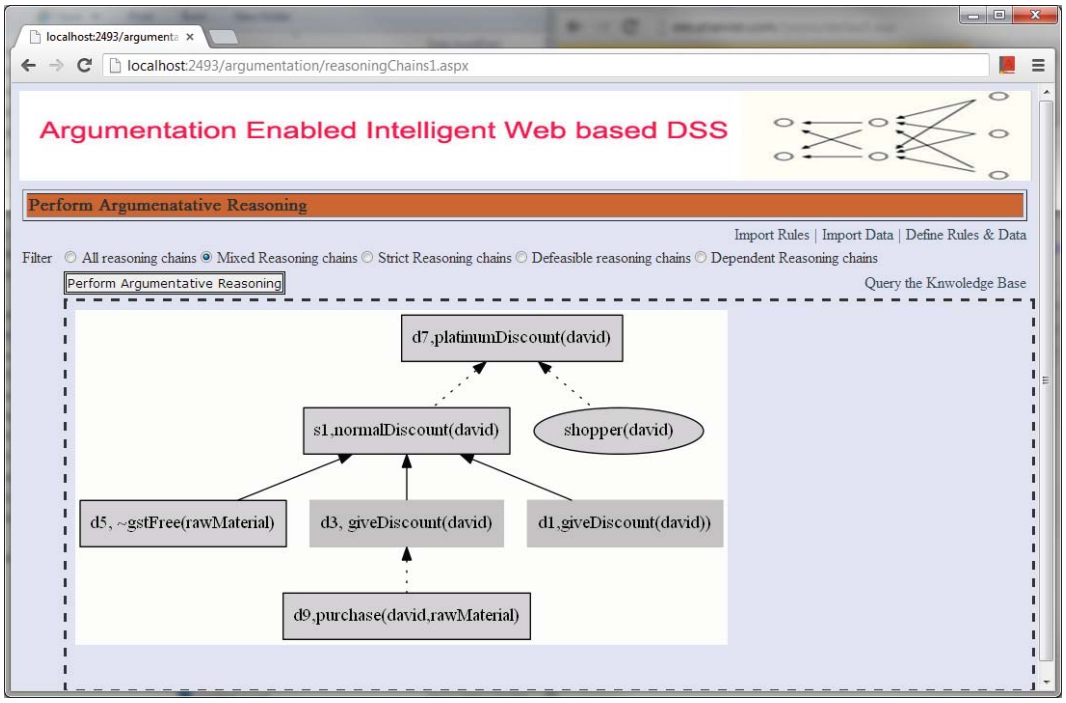


Figure 8.12: Graphical representation of reasoning results with justifications by Web@IDSS

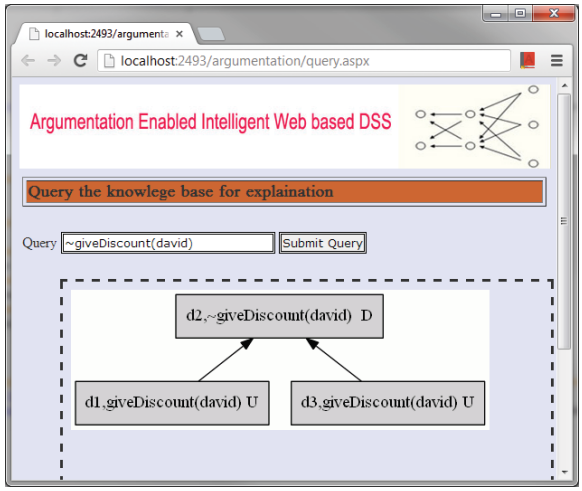


Figure 8.13: Web-based form by Web@IDSS for querying the knowledge base

The graphical representation of the reasoning process takes into account the business policies of BigW and the customer's feedback on its products in the decision-making process. The decision maker can understand the reasoning process by understanding how the arguments support each other as follows:

- arguments d9, d1 and d3 state that if he purchases raw material from BigW, he

is eligible to receive a discount;

- arguments d5 and s1 state that if no GST applies on the raw material, he might receive a normal discount, and
- argument d7 states that if the decision maker receives a normal discount, he is eligible for a platinum discount.

In order to formulate a strategy for product B, Mr David has to perform the same activities with the other suppliers in order to identify the supplier who may offer a maximum discount. As a result, he will obtain 'n' number of reasoning chains, each of which provides a different degree of discount under different conditions. By going through the graphical representations of the reasoning chains, Mr David can easily identify a supplier who may offer him a maximum discount considering his business requirements and conditions with more strict rules. The graphical representation of the reasoning process will also help him to communicate his decision to the enterprise's CEO about why and how he reached the decision to select a particular supplier for raw material for the development of a new product.

8.5.1.3 Achievement of the Aims of Web@IDSS

In this section, the steps performed by Web@IDSS to achieve the aims mentioned in Section 8.5.1.1 are listed.

1. As represented in Figure 8.8 and Figure 8.9, Web@IDSS provides a Web-based form to download the structured public policies of an enterprise (i.e. BigW) in RuleML format. Once the download is complete, the decision maker selects and submits the required information for translation by the Web@IDSS. Once the Web@IDSS completes the translation, it saves the translated information as DeLP rules in the rule base.

2. As represented in Figure 8.10,Web@IDSS provides a Web-based form to download feedback or reviews published on WWW in OWL/RDF format. Once the download is complete, the decision maker selects and submits the required information for translation by the Web@IDSS. Once the Web@IDSS completes the translation, it saves the translated information as DeLP facts in the working memory.

3. As represented in Figure 8.11, Web@IDSS provides a Web-based form to the decision maker to specify his requirements in the form of production rules. Once the decision maker specifies and submits the information, the Web@IDSS saves it

the form of DeLP rules in the rule base. Similarly, the decision maker's specified facts are saved as DeLP facts in the working memory.

4. Figure 8.12 provides the graphical representation of the reasoning results after reasoning over underlying information obtained from the previous steps has been performed. During this process, if conflicts among arguments exist, they are resolved using an automated process, namely the Generalize conflict resolution strategy, and the reasoning process is then displayed to the decision maker in a graphical format. Figure 8.13 represents a Web-based form provided to the decision maker to query the knowledge base and obtain a justification for the reasoning results.

8.5.2 Features evaluation of Web@IDSS

In this section, the features evaluation of Web@IDSS is provided by comparing its functionalities with existing applications in the literature such as Dr Prolog (Antoniou and Bikakis, 2007), Dr-Device (Kontopoulos et al., 2011; Bassiliades et al., 2004) and SweetJess (Grosof et al., 2002), each of which is discussed in Section 2.7.2. Table 8.1 provides the features matrix where a comparative study of Web@IDSS with existing applications is provided and these are discussed in terms of three important aspects as follows:

- Structured, incomplete and/contradictory representation

  The applications presented in Table 8.1 except Web@IDSS, are capable of representing incomplete and/or contradictory structured information only when it comes from a single source by defining priorities between the contradictory rules at compile time before starting the reasoning process. However, Web@IDSS is capable of representing incomplete and/or contradictory structured information which comes from different sources. It does not require definition of priorities between the contradictory rules at compile time before starting the reasoning process.

- Reasoning over such information

  As evident from Table 8.1 , all applications except Web@IDSS provide either data-driven reasoning or goal-driven reasoning. Data-driven reasoning is used to move from current facts to a conclusion, whereas goal-driven reasoning is backward chain reasoning used to move from a conclusion to facts. In the case of Semantic Web applications, both types of reasoning are needed: that is, data-driven reasoning to create a path from initial facts to a conclusion

and goal-driven reasoning to identify reasons and justifications for a particular conclusion which none of the existing applications provide. Another drawback of existing applications is that they define individual preferences at compile time; i.e. the decision maker decides the priorities between the contradictory rules, whereas Web@IDSS does not need any such pre-conditions because it uses an argumentation-driven methodology which is capable of identifying and resolving conflicts in information coming from different sources/users automatically.

- Reasoning chains and graphical representation

  None of these applications, except Web@IDSS, integrate the output of the reasoning process and provide its graphical representation to the decision maker to assist them in the decision-making process. Additionally, Web@IDSS provides a graphical explanation of conflict resolution to produce more easily understandable results in the form of marked dialectical trees.

Table 8.1: Comparison of Web@IDSS with existing applications

	Dr-Prolog	Dr-Device	Situated Courteous logic	Web@IDSS
Language	Prolog	JESS	JESS	C sharp
Logic	Defeasible logic	Defeasible logic	Situated Courteous logic	Defeasible logic programming
Semantic data	RDFS/OWL	RDF	DAML+OIL	RDFS/OWL
Rules representation	RuleML	RuleML	RuleML	RuleML
Conflict representation	Yes	Yes	Yes	Yes
Data-driven reasoning	No	Yes	Yes	Yes
Goal-driven reasoning	Yes	No	No	Yes
Multiple source of info	No	No	No	Yes
Conflict resolution	Individual preferences	Individual preferences	Individual preferences	Argumentation
Explanation	Textual	Textual	Textual	Graphical
Graphical reasoning chais	No	No	No	Yes

8.5.3 Functionality validation of Web@KIDSS

Section 6.3.2 provides the sequence of steps performed by Web@KIDSS for EKI in order to facilitate either the intra-enterprise or inter-enterprise decision-making process when the underlying information is incomplete and/or contradictory. In the next sub-section, the aims to be met during the validation of the functionalities of the Web@KIDSS are listed.

8.5.3.1 Aims for the development of Web@KIDSS

In this section, the aims for the development of Web@KIDSS previously identified in Section 6.2 are listed as follows:

1. a Web-based form to download the recommendations/reasoning chains published by different hybrid reasoning engines over an enterprise's intranet or Internet;

2. a Web-based form for a decision maker to specify his criteria for the valuation of the recommendations/reasoning chains. The system uses the decision maker's criteria and applies it to the downloaded recommendations/reasoning chains, and

3. a reasoning mechanism in the system that can resolve the conflicts present among diverse recommendations and integrate them into an integrated knowledge base, generate its graphical representation to assist decision makers in the intra-enterprise or inter-enterprise decision-making process.

In order to validate the functionality of Web@KIDSS, in the next subsection, the working of Web@KIDSS is demonstrated with an example of a case study discussed in Section 6.2 to achieve the aims discussed above.

8.5.3.2 Working of Web@KIDSS

Figure 8.14 depicts the Web-based form provided by Web@KIDSS for the decision maker to import the recommendations published in the form of reasoning chains in AIF format over the enterprise's intranet. In this way, the decision maker can download all the required recommendations published either on an enterprise's intranet or on the Internet and consider them during the decision making process. The Web-based form also shows the list of downloaded AIF compliant reasoning chain files and the decision maker is able to either view or remove them from the Web@KIDSS.

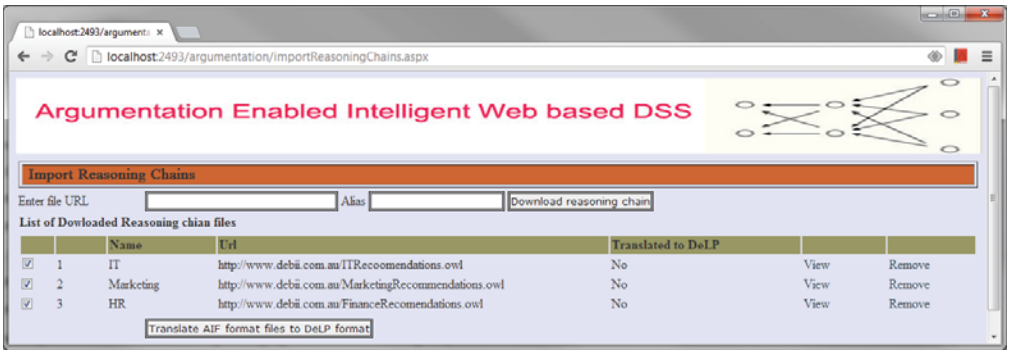


Figure 8.14: Web-based form of Web@KIDSS to import reasoning chains

Once the decision maker has finished downloading the recommendation files, Web@KIDSS needs to translate them to DeLP format in order to reason over them. Figure 8.14 depicts the Web-based form where the decision maker selects the files and Web@IDSS translates them to DeLP format . The translated information is saved in the knowledge base and it is used by the hybrid reasoning engine during the decision-making process.

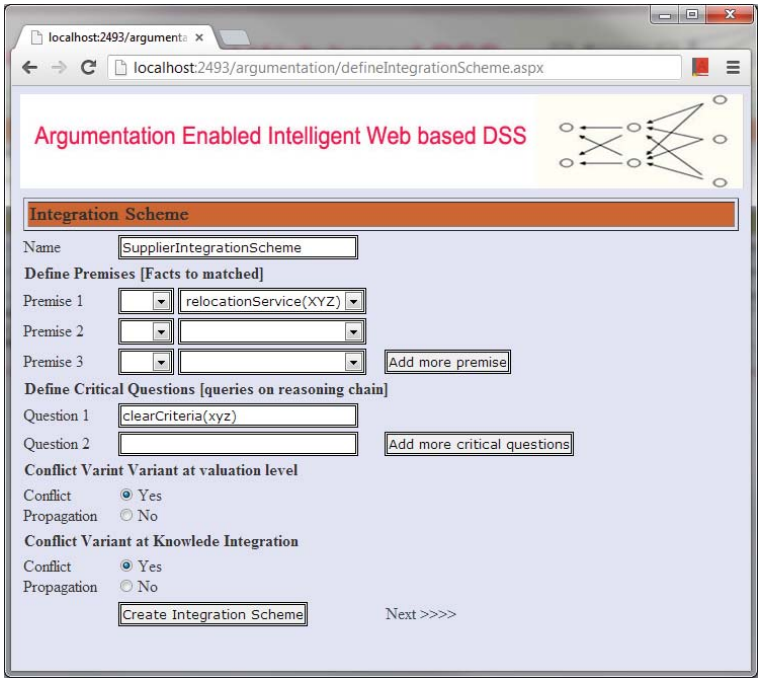


Figure 8.15: Web-based form of Web@KIDSS to define integration scheme

Once the decision maker has finished importing the AIF files, he can define an integration scheme for the valuation of the reasoning chains. Figure 8.15 depicts an interface where a decision maker defines premises that need to be matched, queries to

be executed, and conflict blocking variant at the valuation of a reasoning chain and knowledge integration levels. The decision maker also gives a name to the integration scheme. Once the decision maker has finished the integration scheme, the next step performed by Web@KIDSS is the valuation of the reasoning chains. Figure 8.16 depicts the Web-based form where the decision maker can select the reasoning chains and submit them for the valuation process where Web@KIDSS applies all the premises that need to be matched and queries to be executed on the reasoning chains.

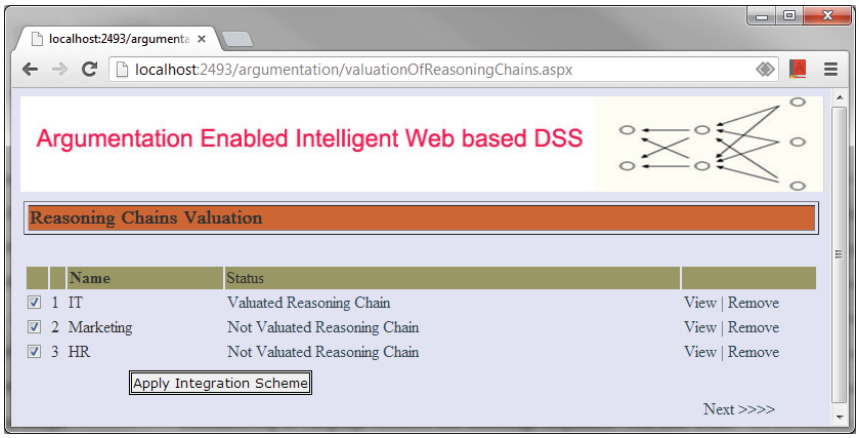


Figure 8.16: Web-based form of Web@KIDSS to select reasoning chains and apply the integration scheme

Figure 8.17 depicts the Web-based form for the decision maker to see the results of the valuation process on a reasoning chain. The text in red shows the conflict between the integration scheme and the contents of a reasoning chain. As a result of the valuation, only those reasoning chains that pass the criteria defined in the integration scheme by the decision maker qualify for further processing.

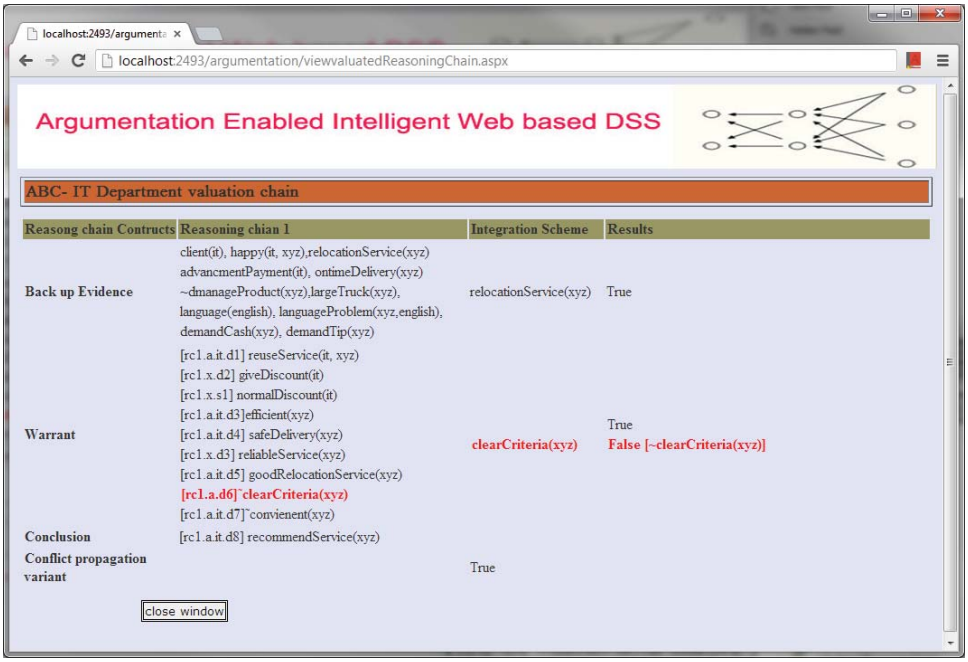


Figure 8.17: Web-based form of Web@KIDSS depicting the results of valuation of a reasoning chain

After the valuation of the reasoning chains, the next step is knowledge integration whereby all the reasoning chains are integrated in the form of a reasoning chain, known as an integrated recommendations space. Figure 8.18 depicts the graphical representation that helps the decision maker of enterprise ABC to understand the whole reasoning process which can result in two recommendations: either recommend XYZ or not. He can identify the reasons for the recommendations as follows:

1. Recommend Service provider XYZ

   The manager of the IT department recommends service provider XYZ for the relocation of enterprise ABC. His recommendation is based on the following information:

   - XYZ considers an enterprise ABC is eligible for a discount. In light of the current available information for decision making, he will offer a normal discount to enterprise ABC.
   - Although XYZ may be inconvenient and not able to capture the enterprise's criteria, the supplier is reliable and will likely provide safe delivery of the enterprise's goods.

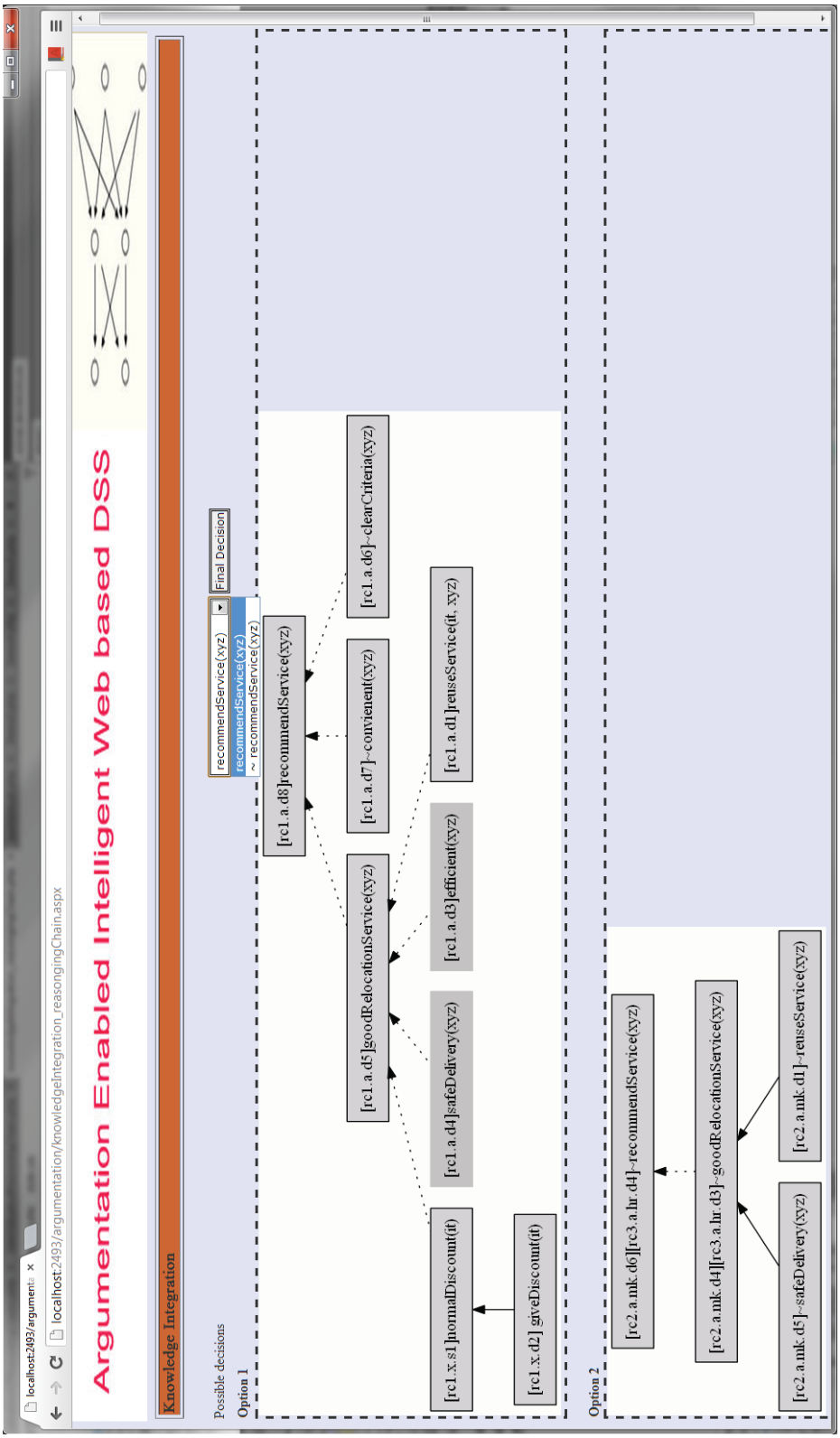


Figure 8.18: Graphical representing of integrated knowledge by Web@KIDSS to facilitate decision making process

- XYZ has been used previously by the IT department and the manager is happy with their service and wants to reuse them for the relocation of the department.

2. Not recommend service provider XYZ

   The managers of the HR and marketing departments do not recommend XYZ for the relocation of the departments of enterprise ABC. Their recommendations are based on the following information:

   - XYZ has been used for relocation services before and the marketing department was not happy with their service.
   - XYZ may not provide safe delivery.
   - Both departments consider XYZ to be a bad relocation service provider.

The final decision needs to be made by the decision maker who selects the result from the drop-down menu and clicks the 'Final Decision' button. This will save the decision maker's preference in the knowledge base.

8.5.3.3 Achievement of the Aims of Web@KIDSS

1. As represented in Figure 8.14, Web@KIDSS provides a Web-based form by which the decision maker can download the recommendations specified by different departments and submit them to the system to translate them into DeLP format and save them in the knowledge base.

2. As represented in Figure 8.15, Web@KIDSS provides a Web-based form to the decision maker to define an integration scheme. The Web@KIDSS takes the decision maker's defined integration scheme for the valuation of the reasoning chains. Figure 8.16 represents the results of the valuation of the reasoning chains selected by the user. Figure 8.17 represents in detail the information of a valued reasoning chain.

3. As represented in Figure 8.18, Web@KIDSS provides the graphical representation of the integration recommendation space to assist the decision maker in the decision-making process for EKI. During this process, Web@IDSS resolves the conflicts among arguments followed by the construction of a new argument and generates an integrated recommendation space.

8.5.4 Features evaluation of Web@KIDSS

In this section, the features evaluation of Web@IDSS is provided by comparing its functionalities with existing applications such as Dr Prolog, Dr-Device, SweetJess and Web@IDSS discussed in Section 8.5.2. Table 8.1 provides the comparative study of Web@KIDSS with existing applications and these are discussed under two important aspects as follows:

- Publication of reasoning chains

  Each of the applications discussed in Table 8.1 except Web@KIDSS does not publish their reasoning results on an enterprise's intranet or on the Internet in a standard format so that the results can be considered by different Semantic Web applications either within the organisation or in other organisaitons.

- Knowledge integration

  As discussed in Section 8.5.2, Web@KIDSS, similar to Web@IDSS, performs hybrid reasoning for the construction of arguments during data-driven reasoning and the resolution of conflicts during goal-driven reasoning. However, it has some additional features which are not present in Web@IDSS and the other existing applications in the literature. These are as follows:

  1. it provides a set of conflict resolution strategies to the decision maker and uses the decision maker's selected strategy during conflict resolution;

  2. it can transform the reasoning chains generated by the hybrid reasoning engine to AIF format and publish them on an enterprise's intranet or on the Internet in OWL/RDF format, and

  3. it provides a solution for knowledge integration. It takes into account the recommendations published over an enterprise's intranet or on the Internet and through the reasoning process, integrates them to generate an integrated recommendation space. It also provides a graphical representation of the integrated recommendation space to the decision maker to assist him in the decision-making process.

Table 8.2: Comparison of defeasible logic based Web IDSS applications

	Dr-Prolog	Dr-Device	Situated Courteous logic	Web@IDSS	Web@KIDSS
Language	Prolog	JESS	JESS	C sharp	C sharp
Logic	Defeasible logic	Defeasible logic	Situated Courteous logic	Defeasible Logic Programming	Defeasible Logic Programming
Semantic data	RDFS/OWL	RDF	DAML+OIL	RDFS/OWL	RDFS/OWL
Rules representation	RuleML	RuleML	RuleML	RuleML	N/A
Incomplete knowledge representation	Yes	Yes	Yes	Yes	Yes
Conflict representation	Yes	Yes	Yes	Yes	Yes
Data-driven reasoning	No	Yes	Yes	Yes	Yes
Goal-driven reasoning	Yes	No	No	Yes	Yes
Multiple source of info	No	No	No	Yes	Yes
Conflict resolution	Individual prefemces	Individual's preferences	Individual's preferences	Single (Generalize) strategy	Multiple argumentation-driven strategies
Explanation	Textual	Textual	Textual	Graphical	Graphical
AIF reification	No	No	No	No	Yes
knowlege Integration	No	No	No	No	Yes

8.5.5 Functionality validation of KR@PMD

Section 7.4.2 provides the sequence of steps performed by KR@PMD to represent and reason over unstructured information, identify and resolve conflicts followed by the integration and graphical representation of the information in such a format that may assist the decision maker in the intra-enterprise or inter-enterprise decision-making process. The next sub-section lists the aims to be met during the validation of the functionalities provided by KR@PMD.

8.5.5.1 Aims for the development of KR@PMD

In this section, the aims for the development of Web@IDSS previously identified in Section 7.3 are listed as follows:

1. a Web-based form for the decision makers to load the unstructured information (e.g. business policy documents), extract the concepts (i.e. elements of the process) from the loaded information and annotate them with the domain ontology;

2. a Web-based form for the decision makers to specify business rules by using the concepts extracted and annotated with the domain ontology in the previous step;

3. a hybrid reasoning engine to perform hybrid reasoning in order to activate/execute the business rules. The engine provides different conflict resolution strategies to the decision maker to identify and resolve the conflicts among the activated business rules, and

4. integrate the output of the hybrid reasoning engine (i.e. business process map) and provide its graphical representation to the decision maker to assist him in the decision-making process.

In order to validate the functionality of KR@PMD, in the next subsection, the working of KR@PMD is described with an example of the case study discussed in Section 7.3 to achieve the aims discussed above.

8.5.5.2 Working of KR@PMD

The KR@PMD provides a Web-based form for the decision maker to extract and annotate the domain model concepts, as depicted in Figure 8.19. Users are provided with two options, i.e. they can either download the business policy document already on the Web by providing the URL, or they can browse the business policy document

on the local system. Following the specification of the business policy document, the Web application loads the business policies and displays them to the decision maker so he can extract the domain model concepts and annotate them by selecting the appropriate data properties and object properties from drop-down menus provided on the Web-based form. As a result, the unstructured information that represents a business policy of an enterprise is transformed to structured information in form of predicates. Once the extraction of domain model concepts and their annotation with the process ontology (as shown in Appendix A.3) then next step is business rules specification.

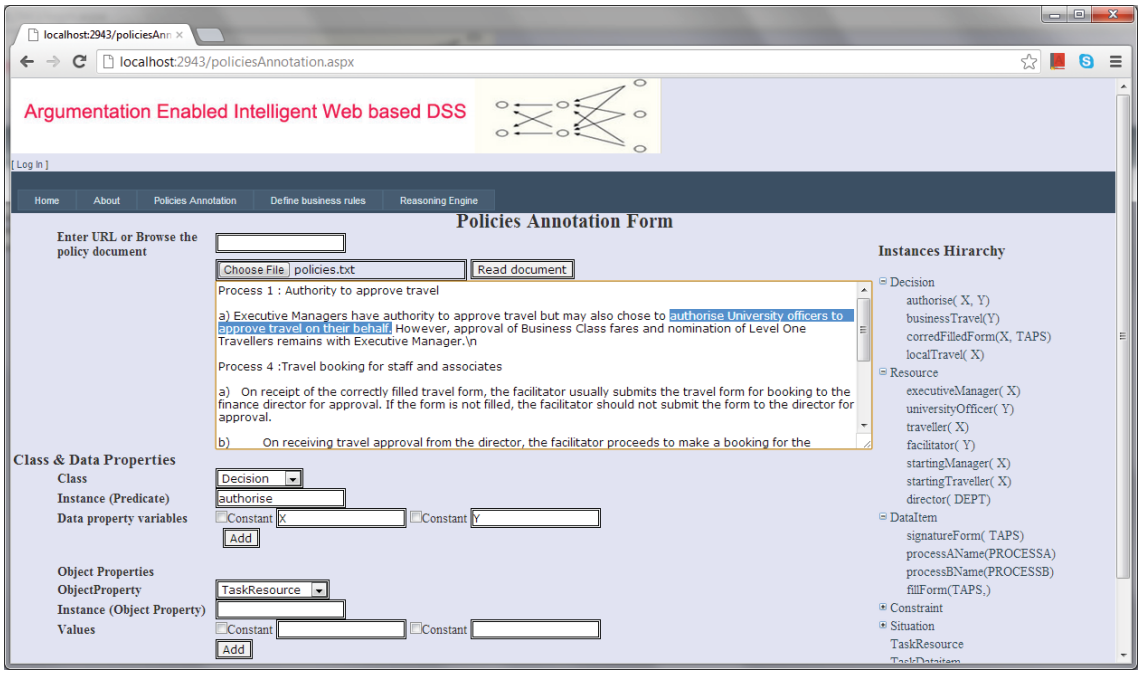


Figure 8.19: Web-based form of KR@PMD for business policies semantic annotation

Figure 8.20 shows the Web-based form for the decision maker to specify the production rules and save them in the rule base. The decision maker creates a production rule by assigning it a name, inference type, set of premises and a conclusion. The decision maker can also view the list of production rules created and either edit or delete them. As a result of such activity, the annotated predicates saved in the previous steps are used to build the process/flow of an activity defined in the unstructured information document. Using the Web-based form depicted in Figure 8.20, the decision maker can also define facts and save them in the working memory. The hybrid reasoning engine uses the production rules and facts saved in the knowledge base during the decision-making process.

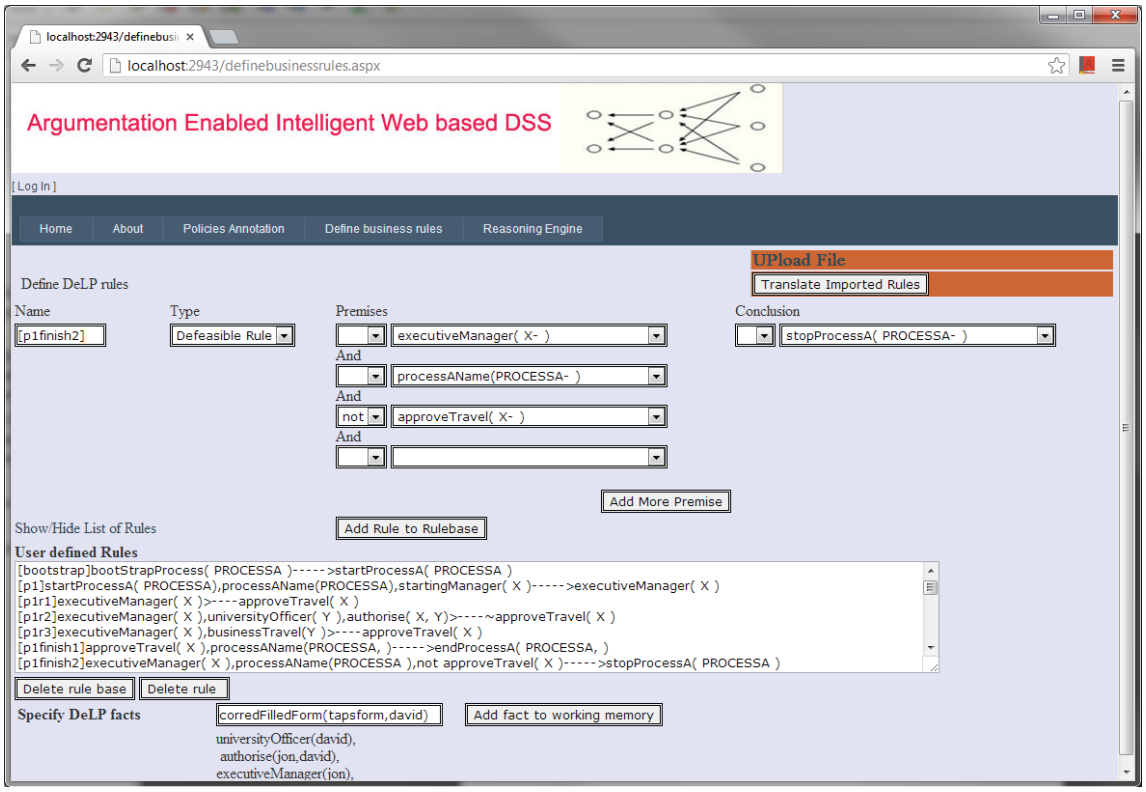


Figure 8.20: Web-based form of KR@PMD for business rules specification

Once the decision maker has specified the production rules, the KR@PMD guides the decision maker through the following steps:

Step 1 argumentative reasoning.

Step 2 conflict resolution algorithms.

Step 3 define fuzzy preferences if fuzzy preference criteria are selected for conflict resolution by the user.

Step 4 define weighted voting by the users and experts if voting-based conflict resolution is selected by the user.

Step 5 produce business process map.

Step 6 query and explain results.

Step 1 is shown in Figure 8.21 where the decision maker starts the argumentative reasoning by clicking the 'Perform argumentative reasoning' button. The Web application takes into account all the business rules present in the knowledge-base

and the facts in the working memory to start performing argumentative reasoning. It displays all the activated business rules, along with the business rules that are in conflict with one another.

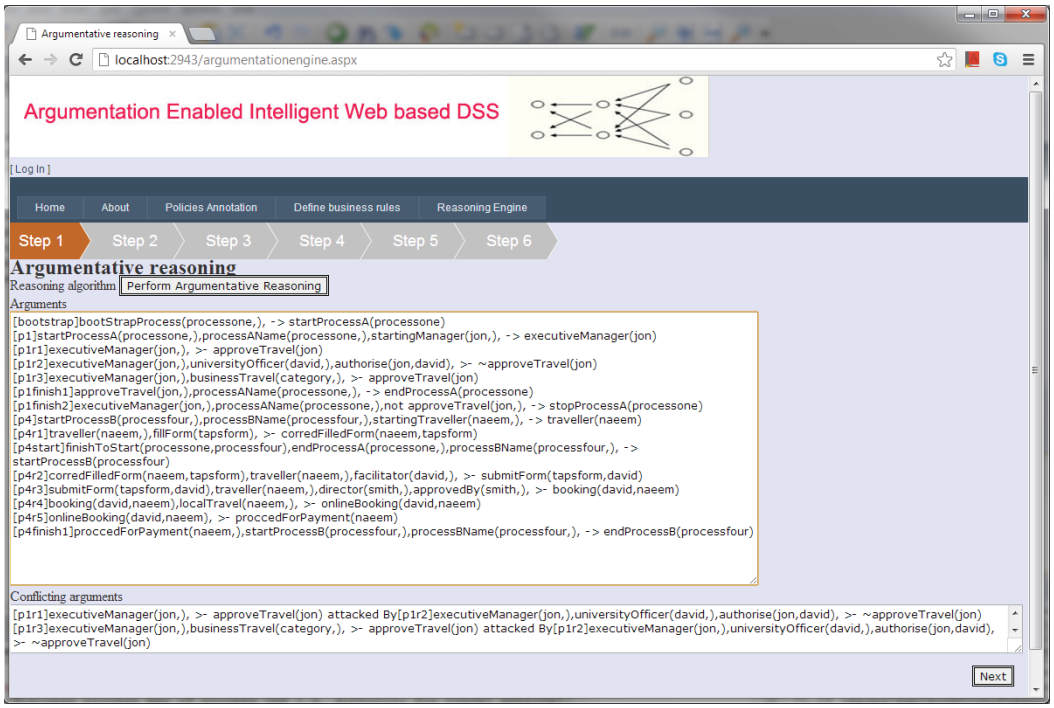


Figure 8.21: Web-based form of KR@PMD shwoing set of arguments and conflict set

Step 2 involves the selection of conflict resolution algorithms as demonstrated in Figure 8.22. The decision maker can select any one of the available algorithms and proceed to Step 3. If the decision maker selects DeLP or Dung style, the KR@PMD executes the conflict resolution algorithms without the decision maker's intervention and takes the decision maker to Step 5.

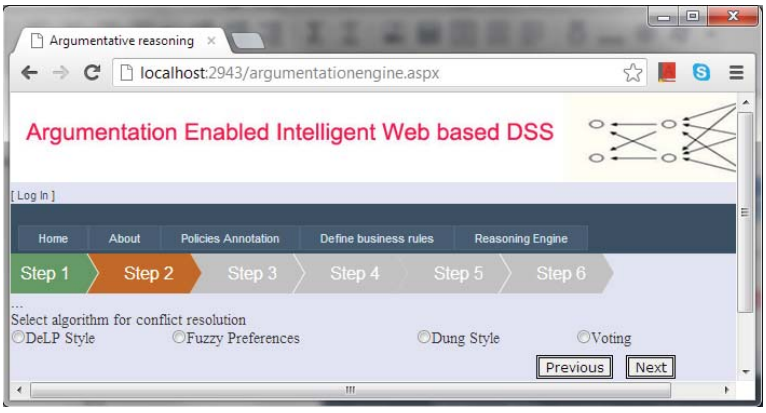


Figure 8.22: Web-based form of KR@PMD representing different algorithms for conflicts resolution

However, if the decision maker selects the fuzzy preferences-based algorithm for conflict resolution, the wizard takes the decision maker to Step 4 where fuzzy preferences can be defined against contradictory business rules, as demonstrated in Figure 8.23. The Web-based form provides a drop-down menu from which the decision maker can select a preference ranging from 0.0 to 1.0. If the decision maker selects the voting-based algorithm for conflict resolution, a Web-based form as depicted in Figure 8.24 is presented to the decision maker to select from three available options: defeated, undecided, survive.

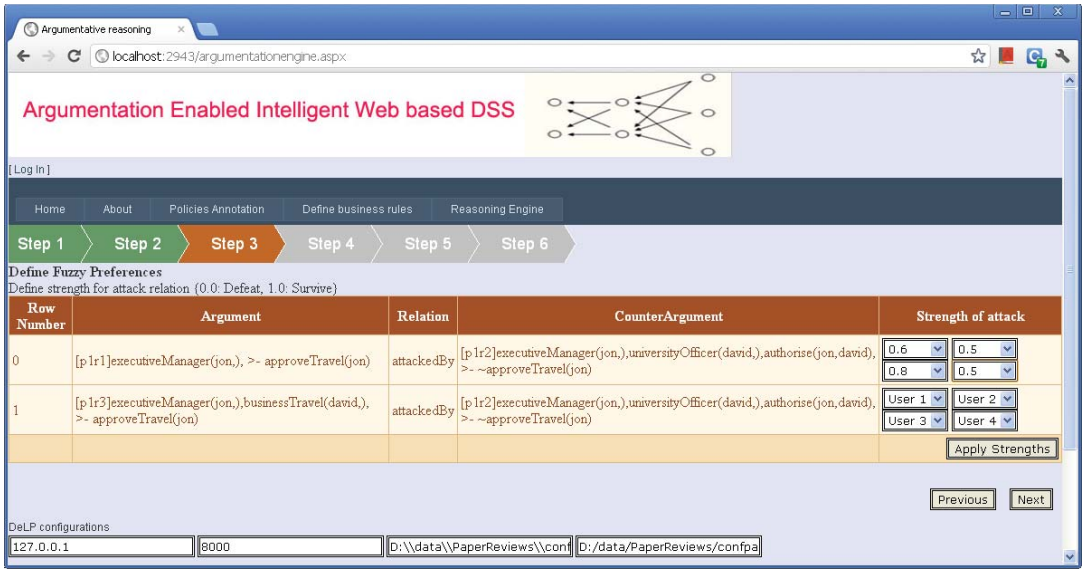


Figure 8.23: Web-based form of KR@PMD for specification of fuzzy preferences

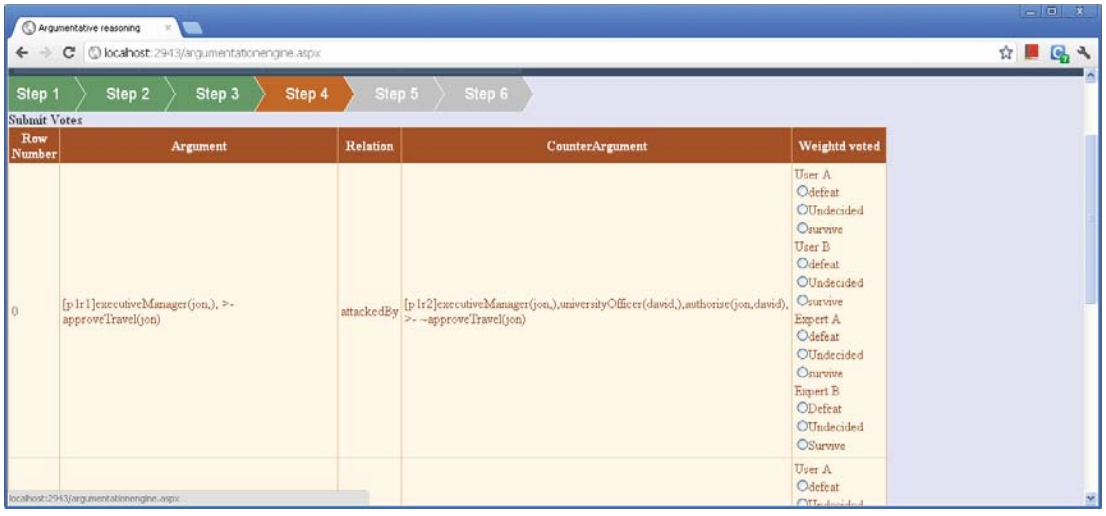


Figure 8.24: Web-based form of KR@PMD for specification of votes

Step 5 generates a graphical flow diagram of business processes called a business process map. The elements of a task specified as premises of a business rule are depicted in the business process map[9] as shown in Figure 8.25 and Figure 8.26.

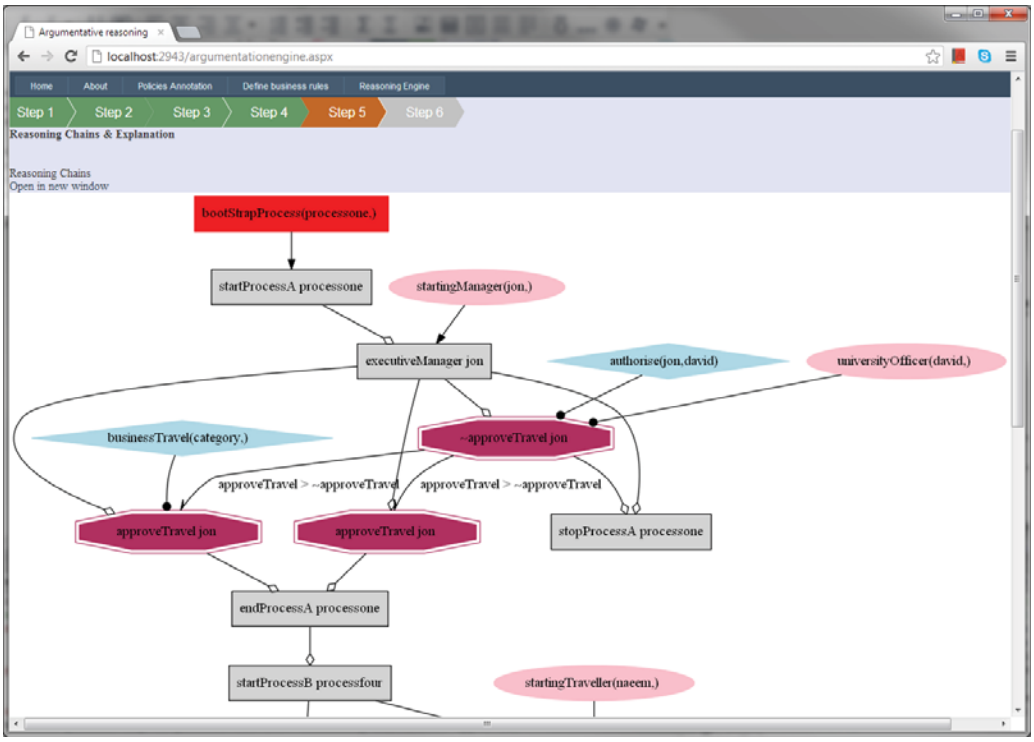


Figure 8.25: Graphical representation of business process map of process 1 by KR@PMD

The graphical representation of the business process map generated from unstructured business policies by KR@PMD enables decision makers in the following ways:

- Identify the important tasks in a business process and how they follow one another as defined by the business policy. They can identify and resolve conflicts that may be present in current business policies so that they can ensure either successful collaboration between departments located within the enterprise and/or in other enterprises or they can ensure they are in accordance with the legal regulations to which the enterprises should comply.

- The business process map can also be used as a validation tool to ensure compliance of operational business processes with business policies. The decision maker can forward a printout of the generated business process map to the departments in order to check the compliance of operational business processes

[9]The claim of a business rule is represented by a grey, rectangular-shaped box

with the enterprise's business policies. The manager of each department can easily identify if a contradictory situation in a business process exists, e.g.,'*Authority to approve travel*' and how it should be resolved. He can further learn that Process 2 can be started only if travel approval is given to 'Jon', otherwise, Process 1 will result in the stopping phase and no further processing will take place.

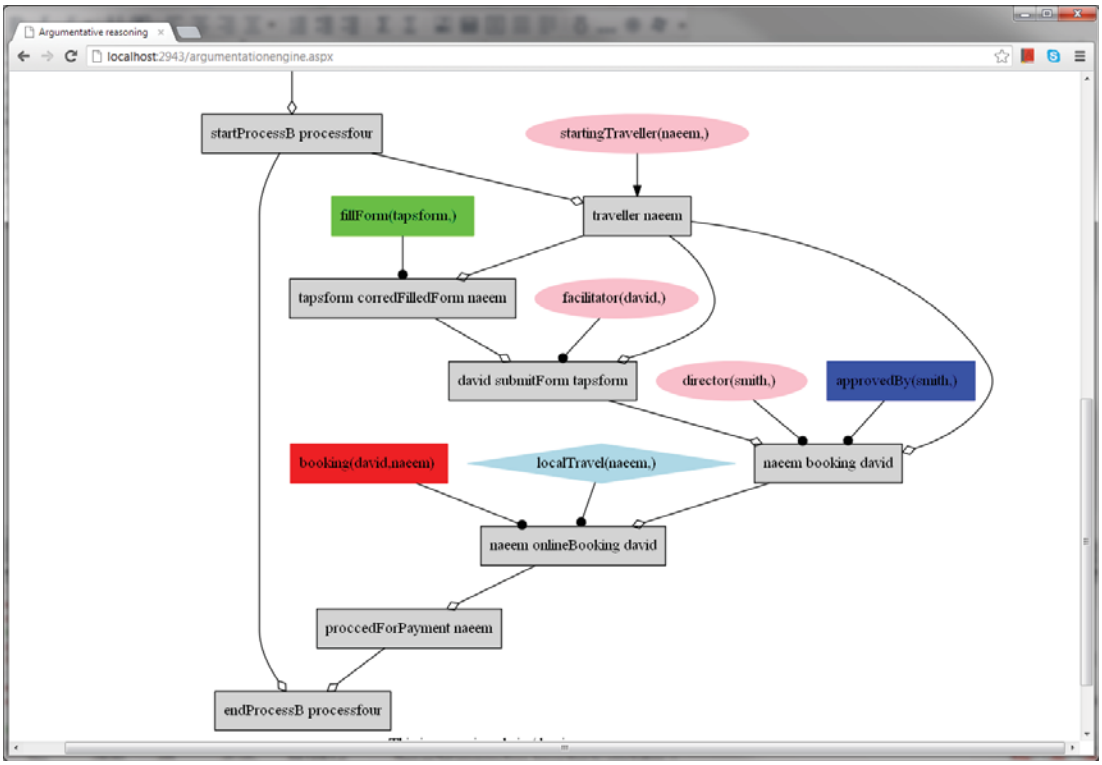


Figure 8.26: Graphical representation of business process map of process 2 by KR@PMD

Step 6 provides an interface for business managers to run queries to obtain an explanation of the decisions made during the process of conflict resolution. Figure 8.27 shows that the Web application provides an interface for query entry (e.g., who is responsible for $approveTravel(X)$ and why?). When the 'Post query' button is pressed, the Web application passes the query to the DeLP server and retrieves the results which are then displayed to the business manager in the form of a dialectical tree.

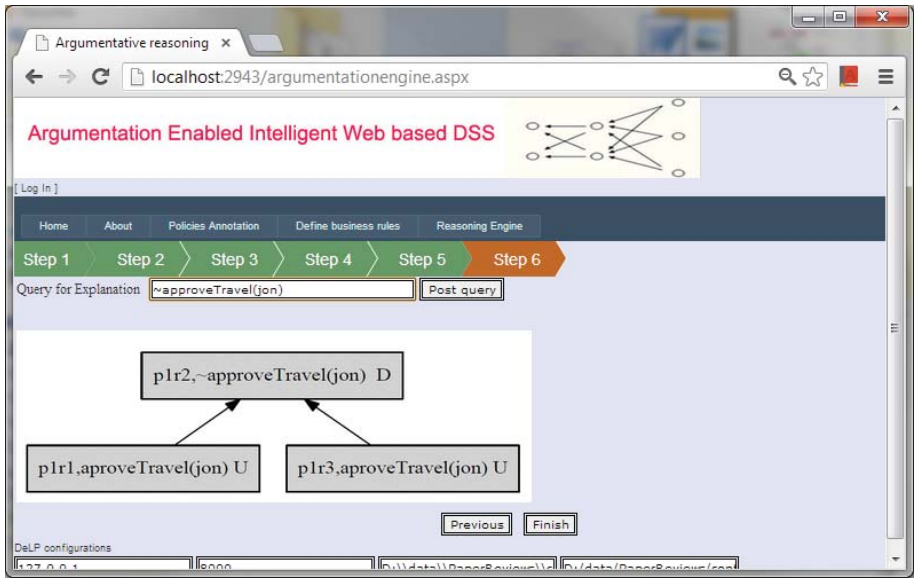


Figure 8.27: Web-based form of KR@PMD for querying the knowledge base and explanation of the results

8.5.5.3 Achievement of the Aims of the KR@PMD

1. As represented in Figure 8.19 , KR@PMD provides a Web-based form to the decision makers to load the business policy document, manually extracts the concepts (i.e. elements of the process) and annotates them with the concepts of the domain ontology.

2. As represented in Figure 8.20, KR@PMD provides a Web-based form to the decision maker to specify the business rules by using the concepts extracted and annotated with the domain ontology.

3. Figure 8.21 provides a set of arguments constructed as result of hybrid reasoning. It also displays the conflict set where arguments and their counter-arguments are displayed. Figure 8.22 provides a Web-based form where a decision maker can select an algorithm to resolve the conflicts among arguments. Figure 8.23 depicts a Web-based form to define the fuzzy preferences over the contradictory arguments. Figure 8.24 depicts a Web-based form to define voting values defined by different users over the arguments in conflict.

4. Figure 8.25 and Figure 8.26 depicts the graphical representation of the business process map and provides its graphical representation to the decision makers so that it can be used as a validation tool to check and ensure the compliance of operational business processes with business policies.

8.5.6 Features evaluation of KR@PMD

The compliance of operational business processes with business policies has increasingly become a subject of interest for enterprises seeking solutions to possible business mergers or to ensure that they are working in accordance with legal regulations or the policies of the government. In order to ensure their compliance, in the literature, different tools and technologies have been proposed to match the enterprise business policies with operational business processes. Table 8.3 provides a comparative study of KR@PDM with the existing most closely related software tools.

- Unstructured information representation proofread

  The KR@PDM is innovative by proposing a 'process ontology' to annotate the business policies and uses a knowledge-based-driven approach to capture the business policies in the form of business rules, whereas the other applications do not take into account the unstructured business policies of an enterprise or collaborating enterprises.

- Argumentation reasoning

  The existing approaches in the literature determine the compliance of business processes with business policies, however, none of them except KR@PMD and Governatori et al. (2006), take into consideration the defeasible nature of a business policy for the generation of a business process map. Governatori et al. (2006) proposed the use of formal models of normative systems to represent the obligations, permissions and prohibitions in a business process. They used the Formal Contract Language (FCL) as a formalism to represent the business policies in the form of rules and through defeasible reasoning, the violations in the business process model are identified and depicted in the business process map. However, they define only individual preferences in the form of priorities among the contradictory rules coming from a single source and do not provides a solution for conflict resolution when information that may be incomplete and/or contradictory information comes from different sources present within an enterprise and/or in other enterprises. KR@PMD addresses the issues faced by Governatori et al. (2006). It allows the representation and reasoning over information coming from different sources and provides different argumentation-driven conflict resolution strategies to resolve the conflicts present in the underlying information.

Table 8.3: Comparison of KR@PMD with existing techniques to check compliance of business policy with business process

	Governatori et al. (2006)	Wang et al. (2009)	Weigand et al. (2011)	Awad et al. (2011)	KR@PMD
Input	Business policies	Business policies	Business policies	Compliance policies	Business policies
Output	Business process map	Business process map	A set of executable rules in RIF format	Visual compliance violations	Business process map
Knowledge representation language	Formal Contract Language (FCL)	N/A	Deontic constraint language (DCL)	Computational Tree logic (CTL)	Defeasible logic programming (DeLP)
Approach for business policies annotation	N/A	Tagging	Semantic annotation via Rule ontology SBVR	Visual language BPMN-Q	Semantic annotation via Process ontology
Information specification form	FCL rules	Tagged information	DCL rules	CTL formula	The business rules are defined in Defeasible Logic Programming (DeLP)
Reasoning model	Goal-driven reasoning	N/A	N/A	Goal-driven reasoning	Hybrid reasoning (data-driven and gaol-driven reasoning)
Handling of defeasible business policies	Yes	N/A	N/A	N/A	Yes
Conflicts resolution	individual preferences to resolve conflicts	N/A	N/A	N/A	Argumentation-driven reasoning to resolve conflicts

- Process map generation

  All the applications except Wang et al. (2009) provide the results as a business process map. However, none of them, except KR@PMD, depict the contradictory information that is present in the business process map and how the conflicts have been resolved during the hybrid reasoning process.

8.6 Conclusion

In this chapter, the developed Semantic Web applications were demonstrated with the support of argumentation to assist the decision maker in the intra-enterprise or inter-enterprise decision making process. Illustrative examples the working of the various phases of the Semantic Web application were given and an explanation was presented as to how they assist the decision maker in either translating or specifying information, reason over it, resolve conflicts and produce a graphical representation of the reasoning results.

Chapter 9 - Recapitulation and Future Work

9.1 Introduction

In the existing literature, the approaches proposed for information representation and reasoning in Semantic Web applications do not provide any solution for Enterprise Information Integration (EII) and Enterprise Knowledge Integration (EKI) when the underlying information (both structured and unstructured) is potentially incomplete and/or contradictory and exists within the enterprise and/or in other enterprises. At the same time, defeasible reasoning-based implementations on the Semantic Web proposed in the literature have the capability to represent and reason over incomplete and/or contradictory information, only if it comes from an individual user/source with the help of predefined priorities between contradictory rules. However, such approaches fail to provide a solution in a group decision-making scenario where information may come from different sources/users and where priorities are not defined between contradictory rules in advance i.e. before reasoning.

In order to overcome this disadvantage and to provide monological argumentation support in Semantic Web applications by enabling them to represent, reason and integrate incomplete and/or contradictory information, five major research objectives have been identified (in Section 3.5) and addressed in this thesis. In Section 9.2, the different research issues that have been identified and addressed in this thesis are recapitulated. In Section 9.3, the contributions made by this thesis to the literature by successfully addressing the research issues are highlighted. In Section 9.4, areas for future work are identified and in Section 9.5 the chapter is concluded.

9.2 Recapitulation

The World Wide Web (WWW) is one of the major sources of information for software agents and Web applications to generate new knowledge and assist in the decision-making process. The extension of WWW i.e. the Semantic Web, provides a language stack (i.e. Semantic Web stack) that enables software agents and Web applications to represent and understand this information and process it autonomously. However, the current languages that lie at the logic layer of the Semantic Web are incapable of representing and reasoning over information that may be incomplete and/or contradictory. Although approaches have been proposed in the literature by different researchers to exploit defeasible reasoning in the area of the Semantic Web, none of them present an approach by which Semantic Web applications can represent, reason and integrate information (i.e. that may be incomplete and/or contradictory) when information comes from heterogenous sources. As a result, Semantic Web applications in an enterprise are not able to consider the information which exists within the enterprise and/or in other enterprises and fail to provide solutions for EII and EKI.

One way by which the above mentioned problem has been addressed in the area of AI is by using 'argumentation'. Argumentation formalisms are considered a pivotal methodology to reach a conclusion in the presence of incomplete and/or contradictory information coming from different sources/users. However, due to a lack of reusable components from AI, current argumentation-driven Semantic Web applications are dialogical in nature and provide no solution for monological argumentation on information which exists within the enterprise and/or in other enterprises to assist the decision maker in the decision-making process. So, in the course of the research documented in this thesis, the broad issue to be addressed i.e. *the design and development of a generic framework for monological argumentation in Semantic Web applications. Such a framework can be exploited for the development of different Semantic Web applications to represent, reason and integrate information exists within an enterprise and/or in other enterprises for enhanced business intelligence,* was identified. Several sub-problems were identified to solve the broad issue as follows:

1. Propose a methodology for incomplete and/or contradictory information representation in Semantic Web applications. Such information may be present within the enterprise and/or in other enterprises and may need to be considered during the intra-enterprise or inter-enterprise decision-making process.

2. Propose a methodology for monological argumentation performed by a hybrid reasoning engine to reason over incomplete and/or contradictory information.

The proposed methodology needs to:

(a) Extend the Rete network in order to compile rules that may represent incomplete and/or contradictory information.

(b) Define syntax and semantics for data-driven reasoning over underlying information for arguments construction.

(c) Define syntax and semantics for goal-driven reasoning to identify and resolve conflicts between arguments.

(d) Propose a methodology for different argumentation-driven conflict resolution strategies to resolve conflicts between arguments and their counter-arguments during goal-driven reasoning.

3. Propose a mechanism to integrate the information being produced by different argumentation-driven hybrid reasoning engines and provide a graphical representation for the decision maker for a better understanding of the reasoning process and its results.

4. Propose a mechanism to export the generated reasoning chains to other Semantic Web applications and vice versa. This will help to bring inter-operability between different information systems and pave the way for knowledge integration.

5. Propose a mechanism to query the knowledge base once the hybrid reasoning is complete in order to obtain an explanation of the reasoning results.

6. Propose a methodology to integrate the reasoning chains produced by different information systems into a coherent reasoning chain. Such knowledge integration will provide a complete picture about information spanning across different information systems.

7. Exploit the proposed GF@SWA in different Semantic Web applications to support intelligent decision making.

8. Validate the functionality and evaluate the features of the proposed GF@SWA.

9.3 Contributions of the thesis

The major contribution of this thesis to the literature is that it proposes a defeasible logic programming-based framework for monological argumentation support

in Semantic Web applications, which enables them to consider incomplete and/or contradictory which exists within the enterprise and/or in other enterprises to obtain better decision-making support in the intra-enterprise or inter-enterprise decision-making process. The contributions of this thesis are as follows:

1. Proposed a methodology for incomplete and/or contradictory information representation by extending Defeasible logic programming (DeLP) in order to represent incomplete and/or contradictory information in Semantic Web applications to assist group decision making. The methodology also proposed a translation mechanism to translate the information present in either RuleML or OWL/RDF format to DeLP format.

2. Proposed a methodology for monological argumentation performed by a hybrid reasoning engine. The hybrid reasoning engine performs data-driven reasoning for argument construction and goal-driven reasoning for conflict identification and resolution.

3. Proposed a methodology for different argumentation-driven conflict resolution strategies to resolve conflicts between arguments and their counter-arguments.

4. Proposed a methodology that can integrate the output of a hybrid reasoning engine in the form of a reasoning chain (called information integration). Such methodology links the facts to a conclusion and represents the reasoning chain in a graphical format.

5. Proposed a methodology for importing/exporting integrated information (i.e. reasoning chain) to different Semantic Web applications.

6. Proposed a methodology that involves the definition and application of an argumentation scheme over the reasoning chains followed by argumentative reasoning to integrate knowledge that comes from different hybrid reasoning engines into a single reasoning chain to facilitate enterprise-wide decision making.

7. Proposed a methodology for the hybrid reasoning engine to have a querying and answering capability backed by an explanation of conflict resolution and/or the conclusions drawn for the decision maker.

8. Demonstrated the application of GF@SWA in different Semantic Web applications to support intelligent decision making over incomplete and/or contradictory information which exists within the enterprise and/or in other enterprises.

In the following, a brief explanation of the contributions which this thesis has made to the existing literature is given.

9.3.1 Contribution 1: Methodology for incomplete and/or contradictory information representation

The first contribution of this thesis to the existing literature is that it proposes a methodology to represent information which exists within the enterprise and/or in other enterprises that may be incomplete and/or contradictory for consideration in the decision-making process. In the proposed methodology, first Defeasible logic programming (DeLP) is selected for information representation and the reasons behind the selection are discussed in Chapter 4. By using DeLP, the information presented in the different Semantic Web applications (each of which are discussed in Chapters 5, 6 and 7) is captured in DeLP format either directly (with the help of Web-based forms) or indirectly (translation/transformation of existing information with the help of a translator). For transformation of existing structured information, two translators were developed i.e. RuleML translator and OWL/RDF translator, which are discussed in Chapter 5. To consider unstructured information in the decision-making process, a semantic annotation mechanism was proposed, as discussed in Chapter 7.

To the best of my knowledge, DeLP has been discussed in AI literature for information representation for software agents, critic and recommender systems etc., but it has been not used for information representation in Semantic Web applications for BI.

9.3.2 Contribution 2: Methodology for monological argumentation performed by a hybrid reasoning engine

The second contribution of this thesis to the existing literature is that it proposes a methodology for monological argumentation performed by hybrid reasoning to reason over information represented using DeLP language. The methodology was discussed in Chapter 5. In the proposed methodology, the Rete network was extended to compile DeLP rules and make them ready for the hybrid reasoning engine. The hybrid reasoning engine performs two types of reasoning: firstly, data-driven reasoning for arguments construction; and secondly goal-driven reasoning for conflicts identification between arguments and their resolution. For knowledge integration, the working of hybrid reasoning was further extended with syntax and semantics, as discussed in Chapter 6.

To the best of my knowledge, there is no methodological approach proposed in literature where monological argumentation is performed by a hybrid reasoning

engine that can reason over underlying information which may be incomplete and/or contradictory and use it in Semantic Web applications for BI.

9.3.3 Contribution 3: Methodology for different argumentation-driven conflict resolution strategies to resolve conflicts between arguments and their counter-arguments

The third contribution of this thesis to the existing literature is that it proposes a methodology for the provision of different argumentation-driven conflict resolution strategies to resolve conflicts between arguments and their counter-arguments. Four different conflict resolution strategies were proposed: the Generalize conflict resolution, Dung's style based conflict resolution, fuzzy preferences and voting-based conflict resolution. The Generalize conflict resolution strategy was discussed in Chapters 5 and 6, whereas the other strategies were discussed in Chapter 7. Each conflict resolution algorithm takes into account different conflict resolution criteria in order to address different contexts.

In the literature, the abovementioned conflict resolution strategies have been used but they have not been exploited in monological argumentation performed by a hybrid reasoning engine to resolve the conflicts among arguments in Semantic Web applications for BI.

9.3.4 Contribution 4: Methodology to integrate the output of a hybrid reasoning engine in the form of a reasoning chain and generate its graphical representation

The fourth contribution of this thesis to the existing literature is that it proposes a methodology to integrate the output of a hybrid reasoning engine in the form of a reasoning chain and provides its graphical representation for decision makers to assist them in the decision-making process. The methodology was discussed in Chapters 5 and 6 with the help of Semantic Web applications, and discussion was provided on how arguments are linked to form a reasoning chain after conflict resolution and how its graphical representation assists the decision maker in different enterprise contexts for decision making. In Chapter 7, the proposed methodology was extended to provide more informative graphical representation of a reasoning chain such as a business process map extracted from the unstructured business policies of an enterprise.

Some argumentation tools have been proposed in the literature to manually

draw and link arguments (i.e. dialogical argumentation) in the format of reasoning chains, however, there is no proposed approach by which the output of monological argumentation performed by a hybrid reasoning engine is integrated in the form a reasoning chain in Semantic Web applications for BI.

9.3.5 Contribution 5: Methodology for importing/exporting integrated information to different Semantic Web applications

The fifth contribution of this thesis to the existing literature is that it proposes a methodology to export the generated reasoning chains in a standard format so that they can be considered by other software systems and vice versa. This methodology was discussed in Chapter 6. By using this methodology, the Semantic Web application working within the enterprise and/or in other enterprises can share/exchange information in AIF format, paving the way for knowledge integration.

In the literature, different approaches have been proposed to import and export information in AIF format, however, none of them provide any mapping of DeLP-based reasoning chains to AIF format and vice versa, therefore they provide no solution for information and knowledge integration using Semantic Web applications for BI.

9.3.6 Contribution 6: Methodology for knowledge integration

The sixth contribution of this thesis to the existing literature is that it proposes a methodology to integrate the reasoning chains produced by different hybrid reasoning engines into a coherent reasoning chain i.e., knowledge integration, in order to provide a complete picture about a subject spanning across different Semantic Web applications. This methodology was discussed in Chapter 6. By using the proposed methodology, a decision maker can define an integration scheme in order to evaluate the reasoning chains followed by argumentative reasoning that results in the construction of an integrated recommendations space.

To the best of my knowledge, no approach or conceptual model has been proposed in the literature which provides a solution for knowledge integration in Semantic Web applications for BI when the underlying information is incomplete and/or contradictory.

9.3.7 Contribution 7: Methodology for the hybrid reasoning engine to have a querying and answering capability

The seventh contribution of this thesis to the existing literature is that it proposes a mechanism to equip the hybrid reasoning engine with a querying and answering capability backed by an explanation of the results achieved through hybrid reasoning. This methodology was presented in Chapter 6. In Chapter 8, different Semantic Web applications were discussed that provide a Web-based interface to query the knowledge base and show the graphical representation of the results which are displayed back to the users.

In the current literature, DeLP query support has been provided to software agents in AI, however, this has not been exploited in Semantic Web applications for BI.

9.3.8 Contribution 8: Application of GF@SWA in different Semantic Web applications to support intelligent decision making

The eighth contribution of this thesis to the existing literature is that it demonstrates how the proposed generic framework i.e. GF@SWA, can be exploited by different Semantic Web applications to represent, reason and integrate information which exists within the enterprise and/or in other enterprises. The Web@IDSS, discussed in Chapter 5, exploits the functionalities of GF@SWA in order to provide a solution for EII when the underlying structured information is incomplete and/or contradictory. In Chapter 7, KR@PMD also exploits the functionalities of GF@SWA to provide a solution for EII when the underlying unstructured information is incomplete and/or contradictory. The Web@KIDSS, discussed in Chapter 6, exploits the functionalities of GF@SWA in order to provide a solution for EKI.

To the best of my knowledge, apart from this thesis, there is no proposed generic framework in the literature on top of which different Semantic Web applications for BI can be built in order to represent, reason and integrate information which exists within the enterprise and/or in other enterprises.

9.4 Future work

In this thesis, a defeasible logic programming-based framework was designed and developed to support monological argumentation in Semantic Web applications. The generic nature of the proposed framework makes it flexible enough to be applied in

different Semantic Web applications, as explained in Chapters 5, 6 and 7, to provide better decision support to decision makers in the intra-enterprise or inter-enterprise decision-making process.

In this section, the future work that will be undertaken in order to strengthen the proposed framework to provide more intuitive results to support the decision-making process is discussed. The possible areas are as follows:

1. Automated production rules extraction from unstructured information.
2. Extension of the proposed framework to work with machine learning algorithms for better classification of information.
3. Extend the proposed framework with an actual/generic argument model for practical reasoning (GAAM).
4. A collaborative framework to reason over qualitative data to assist group decision making.

9.4.1 Automated production rules extraction from unstructured information

As discussed in Chapter 7, the proposed framework provides a semantic annotation methodology to consider unstructured information in Semantic Web applications. In this methodology, a decision maker is provided with a Web-based form on which to load the process ontology and unstructured information e.g. a business policy document, after which the unstructured information is read and the process elements are extracted and annotated with a process ontology, and then the annotated predicates are utilized for the specification of production rules. This methodology may work well for the specification of a small number of rules, however, if the amount of information increases, this methodology does not provide an efficient solution.

This triggers the need for an extensible model for the extraction of production rules as strict and defeasible from unstructured information without human intervention and attaches a strength value to each defeasible production rule for further processing. The strength value may be assigned to a production rule on the basis of certain features, such as the amount of information it carries, how important the information is that it carries etc. The strength of individual production rules can then be used to compute the strength of a reasoning chain.

9.4.2 Extension of the proposed framework to work with machine learning algorithms

The goal of machine learning is to devise learning algorithms that do the learning automatically without human intervention or assistance. A fundamental problem of machine learning is dealing with large spaces of possible hypotheses. In the past decade or so, numerous machine learning methods have been used to automatically learn and recognize complex patterns and make intelligent decisions based on an enterprise data/information. One of the common attributes of these machine learning methods is that their working and functionality is constrained by the amount and nature of the input data. In particular, existing machine learning either doesn't consider domain knowledge during classification, or if it does, then this knowledge holds for the whole domain. Such approaches ignore any specific information or situation that may apply to some small set of chosen learning examples.

In future work, it is intended to enhance the current generation of machine learning techniques with the argumentation formalisms described in this thesis. In such cases, the arguments (undefeated dialectical tree/s) pertinent to specific examples are considered during the mining of an enterprise data[1]. Such work will lay the foundations for performing large-scale analytics on Big data and Cloud computing applications.

9.4.3 Extend the proposed framework as an actual/generic argument model (GAAM) for practical reasoning

Yearwood and Stranieri (2006) proposed an argument structure called the Generic Actual Argument Model (GAAM) for capturing expert reasoning in the form of a certain domain, using a variant of a layout of arguments advanced by Toulmin (2003). In this model, arguments are captured at two levels of abstraction: the generic and actual level. The generic level argument's structure is sufficient to represent claims made by all members of the group and they use this structure to create their actual arguments. The GAAM model represents complex reasoning in such a manner that enables the convenient search and retrieval of relevant information. The argument trees represented using the GAAM framework can be readily converted into a format for rapid deployment and can be made available to other software applications.

In the proposed framework, as discussed in Chapter 6, the reasoning chains have been modelled with respect to the Toulmin model for argument structure i.e. backup evidence, warrant and conclusion. To provide a more practical argument structure, as proposed by GAAM, it is intended to extend the syntax and semantics of the proposed

[1] http://www.ailab.si/martin/abml/

framework and exploit it for modeling the new product development strategy on top of the information present in customer relationship management systems for more practical reasoning.

9.4.4 Collaborative framework for reasoning qualitative models extracted from quantitative data to assist a group decision-making process.

Traditionally, quantitative models over numeric data aim at producing precise numerical results as answers to users' questions about the problem domain. Such precise numerical answers are often overly elaborate, and contain much more information than is actually needed. In everyday life, humans use common sense to reason about problems qualitatively, without numbers.

The logic-based framework discussed in this thesis can be applied by multi-site collaborative teams who exploit information from 'Big data' to generate qualitative models (Suc and Bratko, 2001) that can drive the reasoning process on an underlying rule-base that is specified by different knowledge experts. The outcome of the argumentative reasoning process will assist collaborative teams to discover non-trivial hidden insights in the Big data analytics and Cloud computing applications, explaining how, what and why information about an issue to the user.

9.4.5 Evaluation for correctness of the reasoning chains produced by GF@SWA

Section 8.5 elaborates the functionality validation and features evaluation of GF@SWA and the reasoning results are depicted in form of the reasoning chains. To evaluate the correctness of reasoning chains, in future work, it is intended to compare the reasoning chains depicted in Section 8.5 with the reasoning chains manually generated by a decision maker when he is provided with incomplete and/or contradictory information.

9.5 Conclusion

In this chapter, the work that has been undertaken and documented in this thesis has been recapitulated and the issues addressed in the literature which prompted the work done in this thesis have been highlighted. The different contributions to the literature as the result of outcome of work done in this thesis have also been highlighted. A brief description of the further work that is intended to be undertaken in order to extend

the approaches developed in this thesis were then provided.

The work that was undertaken in this thesis has been published extensively as a part of the proceedings in peer-reviewed international journals and conferences. Selected publications are provided in Appendix B. A complete list of all the publications arising as a result of the work documented in this thesis is given at the beginning of the thesis.

References

Aalst, W. V. D. (2009). Process-Aware Information Systems: Lessons to Be Learned from Process Mining. In Jensen, K. and van der Aalst, W., editors, *Transactions on Petri Nets and Other Models of Concurrency II*, volume 5460 of *Lecture Notes in Computer Science*, pages 1–26. Springer Berlin / Heidelberg.

Alaranta, M. and Henningsson, S. (2008). An approach to analyzing and planning post-merger is integration: Insights from two field studies. *Information Systems Frontiers*, 10(3):307–319.

Aldowaisan, T. A. and Gaafar, L. K. (1999). Business process re-engineering: An approach for Process Mapping. *Omega*, 27(5):515 – 524.

Amgoud, L. and Cayrol, C. (2002). A reasoning model based on the production of acceptable arguments. *Annals of Mathematics and Artificial Intelligence*, 34(1-3):197–215. 10.1023/A:1014490210693.

Amgoud, L., Cayrol, C., Lagasquie-Schiex, M. C., and Livet, P. (2008). On bipolarity in argumentation frameworks. *International Journal of Intelligent Systems*, 23(10):1062–1093.

Anderson, E. E. (1989). A heuristic for software evaluation and selection. *Software: Practice and Experience*, 19(8):707–717.

Andriole, S. (1986). *Software Validation, Verification, Testing and Documentation: A Source Book*. Petrocelli Books, Inc.

Antoniou, G. and Arief, M. (2002). Executable declarative business rules and their use in electronic commerce. In *proceedings of the 2002 ACM Symposium on Applied Computing*, SAC '02, pages 6–10, NY, USA.

Antoniou, G. and Bikakis, A. (2007). Dr-prolog: A system for defeasible reasoning with rules and ontologies on the Semantic Web. *IEEE transactions on knowledge and data engineering*, 19(2):233–245.

Antoniou, G., Damasio, C. V., Grosof, B., Horrocks, I., Kifer, M., Maluszynski, J., and Patel-Schneider, P. F. (2005). Combining Rules and Ontologies: A survey. Technical report, EU FP6 Network of Excellence (NoE), rewerse.net/deliverables/m12/i3-d3.pdf.

Antoniou, G., Skylogiannis, T., Bikakis, A., Doerr, M., and Bassiliades, N. (2007). Dr-brokering: A semantic brokering system. *Knowledge-Based Systems*, 20(1):61 – 72.

Antoniou, G. and Van Harmelen, F. (2004). *A Semantic Web primer*. MIT Press.

Antoniou, G. and Wagner, G. (2003). Rules and defeasible reasoning on the semantic web. In *Second International Workshop Rules and Rule Markup Languages for the Semantic Web*, volume 2876 of *RuleML 2003*, pages 111–120, Sanibel Island, FL, USA. Springer-Verlag Berlin Heidelberg.

Assche, F. V., Layzell, P., Loucopoulos, P., and Speltincx, G. (1988). Information systems development: a rule-based approach. *Knowledge-Based Systems*, 1(4):227 – 234.

Awad, A., Weidlich, M., and Weske, M. (2011). Visually specifying compliance rules and explaining their violations for business processes. *Journal of Visual Languages and Computing*, 22(1):30 – 55.

Baader, F., Horrocks, I., and Sattler, U. (2005). Description logics as ontology languages for the semantic web. In *Mechanizing Mathematical Reasoning, Lecture Notes in Computer Science*, volume 2605, pages 228–248. Springer Berlin / Heidelberg.

Bachler, M., Shum, S. B., Chen-Burger, J., Dalton, J., Roure, D. D., Eisenstadt, M., Komzak, J., Michaelides, D., Page, K., Potter, S., Shadbolt, N., and Tate, A. (2004). Collaboration in the semantic grid: a basis for e-learning. In Mostow, J. and Tedesco, P., editors, *Grid Learning Services workshop (GLS 2004) at the 7th International Conference on Intelligent Tutoring Systems (ITS 2004)*, pages 1–12. Eficiência, Porto Alegre.

Baroni, P., Cerutti, F., Giacomin, M., and Guida, G. (2009). Encompassing attacks to attacks in abstract argumentation frameworks. In Sossai, C. and Chemello, G., editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 5590 of *Lecture Notes in Computer Science*, pages 83–94. Springer Berlin / Heidelberg.

Baroni, P., Fogli, D., and Guida, G. (1998). Modeling argumentation in practical reasoning: a conceptual analysis of argument life cycle. In *7th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 1790–1797. Paris, France.

Bassiliades, N., Antoniou, G., and Vlahavas, I. (2004). Dr-device: A defeasible logic system for the semantic web. In *Principles and Practice of Semantic Web Reasoning, Lecture Notes in Computer Science*, volume 3208, pages 134–148. Springer.

Bench-Capon, T. J. M. (1989). Deep models, normative reasoning and legal expert systems. In *ACM Proceedings of the 2nd international conference on Artificial intelligence and law*, pages 37–45, NY, USA.

Bench-Capon, T. J. M. (2003). Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448.

Berners-Lee, T. (2000). Semantic Web on XML. In *Slides from XML 2000 conference*, Washington DC. W3C.

Berners-lee, T., Connolly, D., Kagal, L., Scharf, Y., and Hendler, J. (2008). N3Logic: A logical framework for the World Wide Web. *Theory and Practice of Logic Programming*, 8(3):249–269.

Besnard, P. and Hunter, A. (2008). *Elements of argumentation*, volume 47. MIT Press Cambridge, MA.

Bex, F., Prakken, H., and Reed, C. (2010). A formal analysis of the AIF in terms of the ASPIC framework. In *Third International Conference on Computational Models of Argument*, COMMA 2010, pages 99–110, Amsterdam, The Netherlands.

Blair, J. A. (1999). D. N. Walton, Argumentation Schemes for Presumptive Reasoning. *Argumentation*, 13(3):338–343.

Bochman, A. (2003). Collective argumentation and disjunctive logic programming. *Journal of Logic Computation*, 13(3):405–428.

Boley, H., Kifer, M., Patranjan, P.-L., and Polleres, A. (2007). Rule interchange on the web. In Antoniou, G., Aßmann, U., Baroglio, C., Decker, S., Henze, N., Patranjan, P.-L., and Tolksdorf, R., editors, *Reasoning Web*, volume 4636 of *Lecture Notes in Computer Science*, pages 269–309. Springer Berlin / Heidelberg.

Bondarenko, A., Toni, F., and Kowalski, R. A. (1993). An assumption-based framework for non-monotonic reasoning. In *Proceedings of the second international workshop on*

Logic programming and non-monotonic reasoning, pages 171–189, Cambridge, MA, USA. MIT Press.

Branting, L. K. (1993). A computational model of ratio decidendi. *Artificial Intelligence and Law*, 2(1):1–31.

Brodie, M. (2008a). Understanding our digital universe: Unleashing natural forces. In *2nd IEEE International Conference on Digital Ecosystems and Technologies*, Phitsanulok, Thailand.

Brodie, M. L. (2008b). The end of the computing era: Hephaestus meets the olympians. In Paige, R. F. and Meyer, B., editors, *Lecture Notes in Business Information Processing*, volume 11, pages 0–1. Springer.

Bryant, D. and Krause, P. (2008). A review of current defeasible reasoning implementations. *The Knowledge Engineering Review*, 23(3):227–260.

Burstein, F. and Gregor, S. (1999). The systems development or engineering approach to research in information systems: An action research perspective. In *Proceedings of the 10th Australasian Conference on Information Systems*, pages 122–134, Wellington, Australia.

Cabrerizo, F., Pérez, I., and Herrera-Viedma, E. (2010). Managing the consensus in group decision making in an unbalanced fuzzy linguistic context with incomplete information. *Knowledge-Based Systems*, 23(2):169 – 181.

Carlsson, C. and Turban, E. (2002). DSS: directions for the next decade. *Decision Support Systems*, 33(2):105 – 110.

Causey, R. L. (1994). Evid: A system for interactive defeasible reasoning. *Decision Support Systems*, 11(2):103 – 131.

Cayrol, C. and Lagasquie-Schiex, M.-C. (2009). Bipolar abstract argumentation systems. In Simari, G. and Rahwan, I., editors, *Argumentation in Artificial Intelligence*, pages 65–84. Springer US.

Cayrol, C. and Lagasquie-Schiex, M.-C. (2010). Coalitions of arguments: A tool for handling bipolar argumentation frameworks. *International Journal of Intelligent Systems*, 25(1):83–109.

Cayrol, C., St-Cyr, F. D. D., and Lagasquie-Schiex, M.-C. (2008). Revision of an argumentation system. In *11th International Conference on Principles of Knowledge Representation and Reasoning*, page 124 134, Sydney, NSW, Australia. AAAI Press.

Ceccaroni, L., Cortés, U., and Sànchez-Marrè, M. (2004). OntoWEDSS: augmenting environmental decision-support systems with ontologies. *Environmental Modelling & Software*, 19(9):785 – 797.

Chesnevar, C., McGinnis, J., Modgil, S., Rahwan, I., Reed, C., Simari, G., South, M., Vreeswijk, G., and Willmott, S. (2006a). Towards an argument interchange format. *The Knowledge Engineering Review*, 21(4):293–316.

Chesnevar, C. I., Maguitman, A. G., and Simari, G. R. (2006b). Argument-based critics and recommenders: a qualitative perspective on user support systems. *Data & Knowledge Engineering*, 59(2):293–319.

Cheung, K. . W. and Cheong, M.-P. (2007). Intelligent On-Line Decision Support Tools For Market Operators. In *International Conference on Intelligent Systems Applications to Power Systems*, pages 1 –6, Taiwan, Japan.

Cirstea, H., Kirchner, C., Moossen, M., and Moreau, P.-E. (2004). Production Systems and Rete Algorithm Formalisation. Contrat A04-R-546 || cirstea04d, http://hal.inria.fr/inria-00099850 [Last accessed, 10/02/2012]. Rapport de contrat.

Clark, P. (1991). *A Model of Argumentation and Its Application in a Cooperative Expert System*. PhD thesis, Turing Institute, Department of Computer Science, University of Strathclyde, Glasgow.

Cohen, R. (1987). Analyzing the structure of argumentative discourse. *Computational linguistics*, 13(1-2):11–24.

Coste-Marquis, S., Devred, C., and Marquis, P. (2005). Prudent semantics for argumentation frameworks. In *17th IEEE International Conference on Tools with Artificial Intelligence*, pages 568–572, Hong Kong, China.

Coste-Marquis, S., Devred, C., and Marquis, P. (2006). Constrained argumentation frameworks. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning*, pages 112–122, Menlo Park, California. The AAAI Press.

Daconta, M., Obrst, L., and Smith, K. (2003). *The Semantic Web: a guide to the future of XML, Web services, and knowledge management*. Wiley.

Dean, M. and Schreiber, G. (2004). OWL Web Ontology Language Reference. W3C recommendation (http://www.w3.org/tr/owl-ref/), W3C.

Dellschaft, K., Engelbrecht, H., Barreto, J., Rutenbeck, S., and Staab, S. (2008). Cicero: Tracking design rationale in collaborative ontology engineering. In Bechhofer, S., Hauswirth, M., Hoffmann, J., and Koubarakis, M., editors, *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, pages 782–786. Springer Berlin / Heidelberg.

Deng, H. and Wibowo, S. (2008). A Rule-Based Decision Support System for Evaluating and Selecting IS Projects. In *Proceedings of the International Multi-Conference of Engineers and Computer Scientists*, pages 1962–1968, Hong Kong, China.

Dix, J., Parsons, S., Prakken, H., and Simari, G. (2009). Research challenges for argumentation. *Computer Science - Research and Development*, 23(1):27–34.

Dong, H., Hussain, F. K., and Chang, E. (2011). A service concept recommendation system for enhancing the dependability of semantic service matchmakers in the service ecosystem environment. *Journal of Network and Computer Applications*, 34(2):619 – 631.

Dong, H., Khadeer Hussain, F., and Chang, E. (2010). A human-centered semantic service platform for the digital ecosystems environment. *World Wide Web*, 13(1-2):75–103.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321 – 357.

Dung, P. M., Kowalski, R. A., and Toni, F. (2009a). Assumption-based argumentation. In Simari, G. and Rahwan, I., editors, *Argumentation in Artificial Intelligence*, pages 199–218. Springer US.

Dung, P. M., Thang, P. M., and Hung, N. D. (2009b). Argument-Based Decision Making and Negotiation in E-Business: Contracting a Land Lease for a Computer Assembly Plant. In *Computational Logic in Multi-Agent Systems, Lecture Notes in Computer Science*, volume 5405, pages 154–172. Springer Berlin / Heidelberg.

Eemeren, F. H. V., Grootendorst, R. F., and Henkemans, F. S. (1996). *Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Applications*. Lawrence Erlbaum Associates, Hillsdale NJ, USA.

Eemeren, F. v. and Grootendorst, R. (2004). A systematic theory of argumentation : the pragma-dialectical approach. *Cambridge University Press*.

Eemeren, F. V., Grootendorst, R., and Henkemans, F. S. (2002). *Argumentation Analysis, Evaluation and Presentation*. Lawrence Erlbaum Associates, Mahwah, NJ, USA.

Fan, X., Toni, F., and Hussain, A. (2010). Two-agent conflict resolution with assumption-based argumentation. In *Proceeding of the 2010 conference on Computational Models of Argument: Proceedings of COMMA 2010*, pages 231–242, Amsterdam, The Netherlands, The Netherlands. IOS Press.

Farnham, S., Chesley, H. R., McGhee, D. E., Kawal, R., and Landau, J. (2000). Structured online interactions: improving the decision-making of small discussion groups. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 299–308, NY, USA. ACM.

Fensel, D. (2003). *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag, 2nd edition.

Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17–37.

Freeley, A. J. and Steinberg, D. L. (2008). *Argumentation and Debate: Critical Thinking for Reasoned Decision Making*, chapter 9, pages 163–187. Wadsworth Publishing Company,, 168 edition.

Freeman, K. (1993). *Toward formalizing dialectical argumentation*. PhD thesis, University of Oregon, Eugene, USA.

Garcia, A. J., Rotstein, N. D., Tucat, M., and Simari, G. R. (2007). An Argumentative Reasoning Service for Deliberative Agents. In Zhang, Z. and Siekmann, J., editors, *Knowledge Science, Engineering and Management, Lecture Notes in Computer Science*, volume 4798, pages 128–139. Springer Berlin Heidelberg.

Garcia, A. J. and Simari, G. R. (2004). Defeasible Logic Programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1-2):95–138.

Garcia-Crespo, A., Ruiz-Mezcua, B., Lopez-Cuadrado, J., and Gonzalez-Carrasco, I. (2011). Semantic model for knowledge representation in e-business. *Knowledge-Based Systems*, 24(2):282 – 296.

Gordon, T. F. and Karacapilidis, N. (1997). The Zeno argumentation framework. In *Proceedings of the 6th international conference on Artificial intelligence and law*, pages 10–18, NY, USA. ACM.

Gordon, T. F., Prakken, H., and Walton, D. (2007). The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10-15):875 – 896.

Gordon, T. F. and Walton, D. (2006). The Carneades Argumentation Framework Using Presumptions and Exceptions to Model Critical Questions. In *Proceeding of the 2006 conference on Computational Models of Argument*, pages 195–207, Amsterdam, The Netherlands,. IOS Press.

Governatori, G., Milosevic, Z., and Sadiq, S. (2006). Compliance checking between business processes and business contracts. In *10th IEEE International Conference on Enterprise Distributed Object Computing*, pages 221 –232, Washington, DC, USA.

Grosof, B., Dean, M., and Kifer, M. (2009). Semantic Rules on the Web. In *International Semantic Web Conference*, Tutorial (http://silk.semwebcentral.org/talk-iswc2009-rules-tutorial.pdf), Washington, DC.

Grosof, B., Gandhe, M., Finin, T., et al. (2002). Sweetjess: Translating damlruleml to Jess. In *Proceedings of the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web at 1st International Semantic Web Conference*, volume 60, Sardinia, Italy.

Grosof, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: combining logic programs with description logic. In *12th international conference on World Wide Web*, pages 48–57, NY, USA. ACM.

Gu, T., Pung, H., and Zhang, D. (2004). Toward an OSGi-based infrastructure for context-aware applications. *Pervasive Computing, IEEE*, 3(4):66 – 74.

Gomez, S. A., Chesnevar, C. I., and Simari, G. R. (2005). Embedding defeasible argumentation in the semantic web: an ontology-based approach. In *7th Workshop de Investigadores en Ciencias de la Computación.*, pages 153–157, Río Cuarto, Argentina.

Haenni, R. (2009). Probabilistic argumentation. *Journal of Applied Logic*, 7(2):155–176.

Hofacker, I. and Vetschera, R. (2001). Algorithmical approaches to business process design. *Computers and Operations Research*, 28(13):1253–1275.

Horrocks, I., Parsia, B., Patel-Schneider, P., and Hendler, J. (2005). Semantic web architecture: Stack or two towers? In Fages, F. and Soliman, S., editors, *Principles*

and Practice of Semantic Web Reasoning, volume 3703 of *Lecture Notes in Computer Science*, pages 37–41. Springer Berlin / Heidelberg.

Iyad Rahwan, C. R. (2009). The Argument Interchange Format. In Rahwan, I. and Simari, G. R., editors, *Argumentation in Artifical Intelligence,*, chapter 19, pages 383–402. Springer.

Johnston, B. and Governatori, G. (2003). Induction of defeasible logic theories in the legal domain. In *Proceedings of the 9th International Conference on Artificial Intelligence and Law*, pages 204–213, NY, USA. ACM.

Kacprzyk, J., Fedrizzi, M., and Nurmi, H. (1992). Group decision making and consensus under fuzzy preferences and fuzzy majority. *Fuzzy Sets and Systems*, 49(1):21 – 31.

Kaplan, B. and Maxwell, J. (2005). Qualitative research methods for evaluating computer information systems. In Anderson, J. and Aydin, C., editors, *Evaluating the Organizational Impact of Healthcare Information Systems*, Health Informatics, chapter 2, pages 30–55. Springer New York, New York.

Kartha, N. and Novstrup, A. (2009). Ontology and rule based knowledge representation for situation management and decision support. In Mott, S., Buford, J. F., Jakobson, G., and J.Mendenhall, M., editors, *Intelligent Sensing, Situation Management, Impact Assessment, and Cyber-Sensing*, pages 288–314. The international society for optics and photonics (SPIE).

Katie Atkinson, T. B.-C. (2008). Abstract argumentation scheme frameworks. In Dochev, D., Pistore, M., and Traverso, P., editors, *Artificial Intelligence: Methodology, Systems, and Applications, Lecture Notes in Computer Science*, volume 5253 of *Artificial Intelligence: Methodology, Systems, and Applications*, pages 220–234. Springer Berlin / Heidelberg.

Klein, M. and Dellarocas, C. (2000). A Knowledge-based Approach to Handling Exceptions inWorkflow Systems. *Computer Supported Cooperative Work*, 9(3-4):399–412.

Kontopoulos, E., Bassiliades, N., and Antoniou, G. (2008). Deploying defeasible logic rule bases for the Semantic Web. *Data & Knowledge Engineering*, 66(1):116–146.

Kontopoulos, E., Bassiliades, N., and Antoniou, G. (2011). Visualizing semantic web proofs of defeasible logic in the dr-device system. *Knowledge-Based Systems*, 24(3):406 – 419.

Kotis, K. (2010). On Supporting HCOME-3O Ontology Argumentation Using Semantic Wiki Technology. In Meersman, R., Tari, Z., and Herrero, P., editors, *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, volume 5333 of *Lecture Notes in Computer Science*, pages 193–199. Springer Berlin / Heidelberg.

Lee, K. C. and Chung, N. (2005). A Web DSS approach to building an intelligent internet shopping mall by integrating virtual reality and avatar. *Expert Systems with Applications*, 28(2):333 – 346.

Lee, T., Hendler, J., Lassila, O., et al. (2001). The Semantic Web. *Scientific American*, 284(5):34–43.

Lee, T. B. (2003). The Semantic Web and Challenges http://www.w3.org/2003/talks/01-sweb-tbl/. W3C.

Lee, T. B. (2005). WWW 2005 Keynote, http://www.w3.org/2005/talks/0511-keynote-tbl/. W3C.

Lee, T. B. (2006). Artificial Intelligence and the Semantic Web: AAAI 2006 Keynote, http://www.w3.org/2006/talks/0718-aaai-tbl/overview.html. W3C.

Letia, I. and Groza, A. (2008). A Planning-Based Approach for Enacting World Wide Argument Web. In Badica, C., Mangioni, G., Carchiolo, V., and Burdescu, D., editors, *Intelligent Distributed Computing, Systems and Applications*, volume 162 of *Studies in Computational Intelligence*, pages 137–146. Springer Berlin / Heidelberg.

Li, T., Feng, S., and Li, L. X. (2001). Information visualization for intelligent decision support systems. *Knowledge-Based Systems*, 14(5-6):259 – 262.

Liu, K. and Ong, T. (1999). A modelling approach for handling business rules and exceptions. *The Computer Journal*, 42(3):221–231.

Lohmann, N., Verbeek, E., and Dijkman, R. (2009). Petri Net transformations for business processes - a survey. In Jensen, K. and Aalst, W. M., editors, *Transactions on Petri Nets and Other Models of Concurrency II*, pages 46–63. Springer-Verlag, Berlin, Heidelberg.

Loui, R. P. (1998). Process and policy: Resource-bounded non-demonstrative reasoning. *Computational intelligence*, 14(1):1–38.

Ma, J., Lu, J., and Zhang, G. (2010). Decider: A fuzzy multi-criteria group decision support system. *Knowledge-Based Systems*, 23(1):23 – 31. Special Issue on Intelligent Decision Support and Warning Systems.

Madison, D. (2005). *Process mapping, process improvement, and process management*. Paton Press.

Markovic, I., Jain, S., El-Gayyar, M., Cremers, A. B., and Stojanovic, N. (2009). Modeling and enforcement of business policies on process models with maestro. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009 Heraklion, pages 873–877, Berlin, Heidelberg. Springer-Verlag.

Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., McGuinness, D. L., Sirin, E., and Srinivasan, N. (2007). Bringing semantics to web services with owl-s. *Journal of World Wide Web*, 10(3):243–277.

Martinez, D. C., Garcia, A. J., and Simari, G. R. (2006). On acceptability in abstract argumentation frameworks with an extended defeat relation. In *Computational models of argument: proceedings of COMMA 2006*. IOS Press, Netherland.

Martinez, D. C., Garcia, A. J., and Simari, G. R. (2008). An abstract argumentation framework with varied-strength attacks. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning*, pages 135–144, Sydney, NSW, Australia.

McTavish, D. G. and Loether, H. J. (1999). *Social Research*. Allyn & Bacon.

Meditskos, G. and Bassiliades, N. (2009). Rule-based owl reasoning systems: Implementations, strengths and weaknesses. In *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, chapter VI, pages 124–148. Hershey, PA.

Meyer, N., Feiner, T., Radmayr, M., Blei, D., and Fleischmann, A. (2011). Dynamic catenation and execution of cross organisational business processes - the jCPEX! approach. In Fleischmann, A., Schmidt, W., Singer, R., and Seese, D., editors, *Subject-Oriented Business Process Management*, volume 138 of *Communications in Computer and Information Science*, pages 84–105. Springer Berlin Heidelberg.

Modgil, S. (2009). Reasoning about preferences in argumentation frameworks. *Artificial intelligence*, 173(9-10):901–934.

Munoz, A. and Botia, J. (2008). ASBO: Argumentation system based on ontologies. In Klusch, M., Pechoucek, M., and Polleres, A., editors, *Cooperative Information Agents XII, Lecture Notes in Computer Science*, volume 5180, pages 191–205. Springer.

Negash, S. and Gray, P. (2003). Business intelligence. In *American Conference on Information Systems*, pages 3190–3199, FL, USA.

Newman, S. E. and Marshall, C. C. (1992). Pushing Toulmin Too Far: Learning From an Argument Representation Scheme. Technical Report SSL-92-45, Xerox PARC, Palo Alto, CA, USA,.

Nicolicin-Georgescu, V., Benatier, V., Lehn, R., and HenriBriand (2010). Ontology-Based Autonomic Computing for Decision Support Systems Management: Shared Resources Allocation between Groups of Data Warehouses. In *Third International Conference on Communication Theory, Reliability, and Quality of Service (CTRQ)*, pages 233 –236, Athens/Glyfada, Greece.

Nielsen, S. H. and Parsons, S. (2007). A generalization of Dungs abstract framework for argumentation: Arguing with sets of attacking arguments. In Maudet, N., Parsons, S., and Rahwan, I., editors, *Argumentation in Multi-Agent Systems, Lecture Notes in Computer Science*, volume 4766 of *Lecture Notes in Computer Science*, pages 54–73. Springer Berlin / Heidelberg.

Noor-E-Alam, M., Lipi, T. F., Hasin, M. A. A., and Ullah, A. (2010). Algorithms for fuzzy multi-expert multi-criteria decision making (ME-MCDM). *Knowledge-Based Systems*, 24(3):367–377.

Norta, A. and Eshuis, R. (2010). Specification and verification of harmonized business-process collaborations. *Information Systems Frontiers*, 12(4):457–479. 10.1007/s10796-009-9164-1.

Nunamaker, Jr., J. F., Chen, M., and Purdin, T. D. M. (1990). Systems development in information systems research. *Journal of Management Information Systems*, 7(3):89–106.

Nute, D. (1988). Defeasible reasoning and decision support systems. *Decision Support Systems*, 4(1):97 – 110.

Nute, D. (1994). Defeasible Logic. In *Web Knowledge Management and Decision Support*, volume 2543 of *Lecture Notes in Computer Science*, pages 151–169. Springer Berlin / Heidelberg.

Obeid, N. (1992). Nonmonotonic Reasoning: Logical Foundation of Commonsense. *The Computer Journal*, 35(2):147–147.

Oguz, G., Proctor, M., and Kuncak, V. (2008). Decision tree learning for drools. Master's thesis, Infoscience | Ecole Polytechnique Federale de Lausanne [http://infoscience.epfl.ch/oai2d.py], http://www.scientificcommons.org/36887627.

Oren, N., Norman, T. J., and Preece, A. (2007). Evidential reasoning in bipolar argumentation frameworks. In *4th International Workshop on Argumentation in Multi-Agent Systems*, Hawaii, USA,.

Palau, R. M. and Moens, M.-F. (2009). Argumentation mining: the detection, classification and structure of arguments in text. In *ICAIL '09: Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 98–107, New York, NY, USA. ACM.

Parsia, B. and Sirin, E. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53.

Parsons, T. (1996). What is Argument? *Journal of Philosophy*, 93(4):164–185.

Patel-Schneider, P. F. and Horrocks, I. (2007). A comparison of two modelling paradigms in the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(4):240–250.

Perelman, C. (1969). *The New Rhetoric: A Treatise on Argumentation*. University of Notre Dame Press.

Pham, D., Governatori, G., Raboczi, S., Newman, A., and Thakur, S. (2008). On Extending RuleML for Modal Defeasible Logic. In Bassiliades, N., Governatori, G., and Paschke, A., editors, *Rule Representation, Interchange and Reasoning on the Web, Lecture Notes in Computer Science*, volume 5321, pages 89–103. Springer Berlin Heidelberg.

Pollock, J. L. (2000). Rational Cognition in OSCAR. In *6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL)*, volume 1757/2000 of *Lecture Notes in Computer Science*, pages 71–90. Springer Berlin / Heidelberg.

Power, D. J. (2002). *Decision support systems: concepts and resources for managers*. Greenwood Publishing Group.

Power, D. J. and Sharda, R. (2009). Decision Support Systems. In Nof, S. Y., editor, *Springer Handbook of Automation*, pages 1539–1548. Springer Berlin Heidelberg.

Prakken, H. and Vreeswijk, G. (2002). Logics for defeasible argumentation. In Gabbay, D. M. and Guenthner, F., editors, *Handbook of Philosophical Logic*, volume 4, pages 219–318. Springer.

Rahwan, I. (2005). Guest editorial: Argumentation in multi-agent systems. *Autonomous agents and multi-agent systems*, 11(2):115–125.

Rahwan, I. and Larson, K. (2008). Mechanism design for abstract argumentation. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1031–1038, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Rahwan, I., Ramchurn, S. D., Jennings, N. R., McBurney, P., Parsons, S., and , L. S. (2004). Argumentation-based negotiation. *The Knowledge Engineering Review*, 18(4):343–374.

Rahwan, I., Zablith, F., and Reed, C. (2007a). Towards large scale argumentation support on the semantic web. In *AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence*, pages 1446–1451. AAAI Press.

Rahwan, I., Zablitha, F., and Reed, C. (2007b). Laying the foundations for a World Wide Argument Web. *Artificial Intelligence*, 171(10-15):897–921.

Rajsiri, V., Lorra, J.-P., Banaben, F., and Pingaud, H. (2010). Knowledge-based system for collaborative process specification. *Computers in Industry*, 61(2):161 – 175.

Reed, C. and Rowe, G. (2004). Araucaria: Software for argument analysis, diagramming and representation. *International Journal of Artifical intelligence Tools*, 13(4):961–980.

Reed, C. and Rowe, G. (2007). A pluralist approach to argument diagramming. *Law, Probability and Risk*, 6(1-4):59–85.

Reed, C. and Walton, D. (2003). Abstract Argumentation schemes in argument-as-process and argument-as-product. In *Proceedings of the conference celebrating informal Logic*. CD-Rom. Ontario Society for the Study of Argumentation, Windsor.

Reed, C., Walton, D., and Macagna, F. (2007). Argument diagramming in logic, law and artificial intelligence. *The Knowledge Engineering Review*, 22(1):87–109.

Reijers, H. A., Limam, S., and Van Der Aalst, W. M. P. (2003). Product-Based Workflow Design. *Journal of Management Information Systems*, 20(1):229–262.

Rotstein, N. D., Moguillansky, M. O., Garcia, A. J., and Simari, G. R. (2010). A dynamic argumentation framework. In Baroni, P., Cerutti, F., Giacomin, M., and Simari, G. R., editors, *Frontiers in Artificial Intelligence and Applications*, volume 216 of *Computational Models of Argument - Proceedings of COMMA 2010*. IOS Press.

Sadiq, S. and Governatori, G. (2009). A methodological framework for aligning business processes and regulatory compliance. In Brocke, J. and Rosemann, M., editors, *Handbook of business process management: 2. Strategic alignment, governance, people and culture*, pages 159–176. Springer-Verlag Berlin Heidelberg.

Saggion, H., Funk, A., Maynard, D., and Bontcheva, K. (2007). Ontology-based information extraction for business intelligence. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 843–856, Berlin, Heidelberg. Springer-Verlag.

Salam, A. (2007). Design and implementation of semantic decision support system for supplier performance contract monitoring and execution: Integrating description logics, semantic web rules and service-oriented computing in the context of the extended enterprise. In *Americas Conference on Information Systems*, paper number 293 [http://aisel.aisnet.org/amcis2007/293].

Shim, J. P., Warkentin, M., Courtney, J. F., Power, D. J., Sharda, R., and Carlsson, C. (2002). Past, present, and future of decision support technology. *Decision Support Systems*, 33(2):111 – 126.

Shum, S. B. (2008). Cohere: Towards Web 2.0 Argumentation. In *Proceeding of the conference on Computational Models of Argument*, pages 97–108, Amsterdam. IOS Press.

Silverman, B. G., Bachann, M., and Al-Akharas, K. (2001). Implications of buyer decision theory for design of e-commerce websites. *International Journal of Human-Computer Studies*, 55(5):815 – 844.

Sprado, J. and Gottfried, B. (2009). Semantic argumentation in dynamic environments. In *International Conference Enterprise Information Systems*, pages 236–241, Milan, Italy. Springer, Heidelberg.

Subsorn, P., Xiao, J., and Singh, K. (2008). A web-based application of group decision making in a fuzzy environment. In *5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, volume 1, pages 17 –20, Krabi, Thailand.

Suc, D. and Bratko, I. (2001). Induction of qualitative trees. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL '01, pages 442–453, London, UK, UK. Springer-Verlag.

Suguri, H., Ahmad, H. F., Pasha, M., and Khalid, N. (2008). *Grid Computing Research Progress Chapters*, chapter Foundation for Autonomous Semantic Grid, pages 151–191. Nova Science Publications, USA.

Thomsen, E. (2003). Biś promised land. *Intelligent Enterprise*, 6(5):20–25.

Toni, F. (2007). E-business in argugrid. In Veit, D. and Altmann, J., editors, *Grid Economics and Business Models*, volume 4685 of *Lecture Notes in Computer Science*, pages 164–169. Springer Berlin / Heidelberg.

Torroni, P., Gavanelli, M., and Chesani, F. (2009). Arguing on the Semantic Grid. In Rahwan, I. and Simari, G. R., editors, *Argumentation in Artificial Intelligence*, pages 423–441. Springer US.

Toulmin, S. E. (2003). *The Uses of argument*. Cambridge University Press.

Tsarkov, D. and Horrocks, I. (2006). FaCT++ description logic reasoner: System description. In *Proceedings of the Third international joint conference on Automated Reasoning*, pages 292–297. Springer-Verlag Berlin.

Turetken, O., Elgammal, A., van den Heuvel, Willem-Jan, and Papazoglou, M. (2011). Enforcing compliance on business processes through the use of patterns. In *European Conference on Information Systems*, paper 5, Finland.

Tzagarakis, M., Karousos, N., Karacapilidis, N., and Nousia, D. (2009). Unleashing Argumentation Support Systems on the Web: The case of CoPe-it. In *Proceedings of the Web Science: Society On-Line*, Athens, Greece.

Uldis Bojars, Tudor Groza, J. G. B., Handschuh, S., and Lange, C. (2008). Expressing argumentative discussions in social media sites. In *First Workshop on Social Data on the Web at the 7th International Semantic Web Conference*, Karlsruhe, Germany. http://sdow.semanticweb.org/2008/.

Vahidov, R. and Kersten, G. E. (2004). Decision station: situating decision support systems. *Decision Support Systems*, 38(2):283 – 303.

Van Emden, M. and Kowalski, R. (1976). The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742.

Vreeswijk, G. (1995). IACAS: an implementation of Chisholm's principles of knowledge. In *The Proceedings of the 2nd Dutch/German Workshop on Nonmonotonic Reasoning*, pages 225–234., Utrecht, Netherland.

Vreeswijk, G. A. W. (1997). Abstract argumentation systems. *Artificial Intelligence*, 90(1-2):225 – 279.

Walton, D. (1989). *Informal Logic: A Handbook for Critical Argument*. Cambridge University Press.

Walton, D. (1997). *Appeal to expert opinion: Arguments from authority*. Penn State University Press.

Walton, D. (2005). Justification of Argumentation Schemes. *The Australasian Journal of Logic*, 3:1–13. http://www.philosophy.unimelb.edu.au/ajl/2005/.

Walton, D. (2009). Argumentation theory: A very short introduction. In Rahwan, I. and Simari, G. R., editors, *Argumentation in Artifical Intelligence*, pages 1–22. Springer.

Walton, D. N. (2006). *Fundamentals of Critical Argumentation*. Cambridge University Press, New York, USA.

Wang, H. J., Zhao, J. L., and Zhang, L.-J. (2009). Policy-driven process mapping (PDPM): Discovering process models from business policies. *Decision Support Systems*, 48(1):267 – 281.

Wang, X., Zhang, D., Gu, T., and Pung, H. (2004). Ontology based context modeling and reasoning using OWL. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pages 18–22, Florida USA.

Weigand, H., van den Heuvel, W.-J., and Hiel, M. (2011). Business policy compliance in service-oriented systems. *Information Systems*, 36(4):791 – 807.

Wells, S., Gourlay, C., and Reed, C. (2009). Argument Blogging. In *9th International Workshop on Computational Models of Natural Argument (CMNA 9)*, California, U.S.

Wen, W., Chen, Y., and Chen, I. (2008). A knowledge-based decision support system for measuring enterprise performance. *Knowledge-Based Systems*, 21(2):148 – 163.

Wielemaker, J. (2011). SWI-Prolog RDF parser , http://www.swi-prolog.org/pldoc/package/rdf2pl.html.

Xue, Y., Ghenniwa, H. H., and Shen, W. (2012). Frame-based ontological view for semantic integration. *Journal of Network and Computer Applications*, 35(1):121 – 131.

Yang, X., Bo, Z., and Bei, Z. (2009). Research on Semantic Decision Support System. In *World Congress on Computer Science and Information Engineering*, volume 5, pages 687 –691, Los Angeles, USA.

Yao, Y., Zhong, N., Liu, J., and Ohsuga, S. (2001). Web Intelligence (WI) Research Challenges and Trends in the New Information Age. In *Web Intelligence: Research and Development*, volume 2198 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin / Heidelberg.

Yearwood, J. L. and Stranieri, A. (2006). The generic/actual argument model of practical reasoning. *Decision Support Systems*, 41(2):358 – 379.

Yue, Z. (2011). An extended TOPSIS for determining weights of decision makers with interval numbers. *Knowledge-Based Systems*, 24(1):146 – 153.

Zadeh, L. (1983). A computational approach to fuzzy quantifiers in natural languages. *Computers & Mathematics with Applications*, 9(1):149–184.

Zarefsky, D. (2009). Argumentation: The study of effective reasoning, 2nd edition. volume 2009. Northwestern University.

Zeleznikow, J. and Stranieri, A. (1995). The split-up system: integrating neural networks and rule-based reasoning in the legal domain. In *Proceedings of the 5th international conference on Artificial intelligence and law*, pages 185–194, NY, USA. ACM.

Appendix A - Information captured by Semantic Web applications

A.1 Production rules of a supplier in RuleML format

```
<?xml version="1.0" encoding="UTF-8"?>
<RuleML xmlns="http://www.ruleml.org/0.91/xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.ruleml.org/0.91/xsd

http://debii.curtin.edu.au/~naeem/delp-valid-xsd.xsd">
    <Assert>
        <Implies ruletype="defeasiblerule">
            <oid><Ind>d9</Ind></oid>
            <head>
                <Atom>
                     <Rel>purchase</Rel>
                     <Var>X</Var>
                     <Var>Y</Var>
                 </Atom>
            </head>
            <body>
                         <Atom>
                          <Rel>shopper</Rel><Var>X</Var>
                         </Atom>
                                <Atom>
                    <Rel>product</Rel>
                                          <Var>Y</Var>
                </Atom>
                <Atom>
                    <Rel>havefeedback</Rel>
                                          <Var>F</Var>
                                          <Var>Y</Var>
                </Atom>
                <Atom>
                    <Rel>reviewRate</Rel>
                                          <Var>G</Var>
                                          <Var>F</Var>
                </Atom>
            </body>
        </Implies>
```

```
<Implies ruletype="defeasiblerule">
    <oid><Ind>d1</Ind></oid>
    <head>
        <Atom>
        <Rel>giveDiscount</Rel>
        <Var>X</Var>
        </Atom>
    </head>
    <body>
        <Atom>
            <Rel>shopper</Rel>
                <Var>X</Var>
        </Atom>

        <Atom>
            <Rel>purchase</Rel>
              <Var>X</Var>
              <Var>Y</Var>
        </Atom>
    </body>
</Implies>
<Implies ruletype="defeasiblerule">
    <oid><Ind>d2</Ind></oid>
    <head>
        <Neg>
            <Atom><Rel>giveDiscount</Rel>
                <Var>X</Var>
                </Atom>
        </Neg>
    </head>
    <body>
           <Atom>
            <Rel>shopper</Rel><Var>X</Var>
        </Atom>
        <Atom>
            <Rel>product</Rel>
                <Var>Y</Var>
        </Atom>
        <Atom>
        <Neg>
            <Rel>advancePayment</Rel><Var>X</Var><Var>Y</Var>
        </Neg>
            </Atom>

    </body>
</Implies>
<Implies ruletype="defeasiblerule">
    <oid><Ind>d3</Ind></oid>
    <head>
        <Atom><Rel>giveDiscount</Rel>
                <Var>X</Var>
                </Atom>
    </head>
    <body>
            <Atom>
            <Rel>purchase</Rel>
                <Var>X</Var>
                <Var>Y</Var>
```

```
        </Atom>
        <Atom>
            <Rel>shopper</Rel><Var>X</Var>
        </Atom>
        <Atom>
            <Rel>product</Rel><Var>Y</Var>
        </Atom>
        <Atom>
            <Rel>bulkOrder</Rel><Var>X</Var><Var>Y</Var>
        </Atom>
    </body>
</Implies>
<Implies ruletype="defeasiblerule">
    <oid><Ind>d5</Ind></oid>
   <head>
        <Neg>
            <Atom><Rel>gstFree</Rel>
                <Var>Y</Var>
                </Atom>
        </Neg>
    </head>
    <body>
        <Atom>
            <Rel>product</Rel><Var>Y</Var>
        </Atom>
                        <Atom>
            <Rel>eShop</Rel><Var>Z</Var>
        </Atom>
            <Atom>
          <Neg>
            <Rel>packaging</Rel>
                <Var>Y</Var>
                <Var>Z</Var>
          </Neg>
        </Atom>
    </body>
</Implies>
<Implies ruletype="strictrule">
    <oid><Ind>s1</Ind></oid>
    <head>
                <Atom><Rel>normalDiscount</Rel>
                <Var>X</Var>
                </Atom>
        </head>
    <body>
        <Atom>
            <Rel>product</Rel>
                <Var>Y</Var>
        </Atom>
            <Atom>
                <Neg>
            <Rel>gstFree</Rel><Var>Y</Var>
                </Neg>
        </Atom>
        <Atom>
                    <Rel>giveDiscount</Rel><Var>X</Var>
       </Atom>
    </body>
```

```
        </Implies>
    </Assert>
   </RuleML>
```

A.2 Feedback information in OWL/RDF format

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY ecommerceOntology "http://www.debii.com/~naeem/ecommerceOntology#" >
]>
<rdf:RDF xmlns="http://www.debii.com/~naeem/ecommerceOntology#"
     xml:base="http://www.debii.com/~naeem/ecommerceOntology"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:ecommerceOntology="http://www.debii.com/~naeem/ecommerceOntology#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about=""/>

    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->
    <!-- http://www.debii.com/~naeem/ecommerceOntology#haveFeedback -->
   <owl:ObjectProperty rdf:about="#haveFeedback">
        <rdfs:domain rdf:resource="#Products"/>
        <rdfs:range rdf:resource="#Reviews"/>
    </owl:ObjectProperty>
    <!-- http://www.debii.com/~naeem/ecommerceOntology#purchaseProducts -->
   <owl:ObjectProperty rdf:about="#purchaseProducts">
        <rdfs:domain rdf:resource="#Customer"/>
        <rdfs:range rdf:resource="#Products"/>
    </owl:ObjectProperty>
    <!-- http://www.debii.com/~naeem/ecommerceOntology#sell -->
    <owl:ObjectProperty rdf:about="#sell">
        <rdfs:range rdf:resource="#Products"/>
        <rdfs:domain rdf:resource="#Store"/>
    </owl:ObjectProperty>

   <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Data properties
    //
```

```
   ////////////////////////////////////////////////////////////////////////////////
  -->
   <!-- http://www.debii.com/~naeem/ecommerceOntology#cashPayment -->
  <owl:DatatypeProperty rdf:about="#cashPayment">
       <rdfs:domain rdf:resource="#Customer"/>
       <rdfs:subPropertyOf rdf:resource="#makePayment"/>
       <rdfs:range rdf:resource="&xsd;boolean"/>
   </owl:DatatypeProperty>
   <!-- http://www.debii.com/~naeem/ecommerceOntology#clubMember -->
   <owl:DatatypeProperty rdf:about="#clubMember">
       <rdfs:domain rdf:resource="#Customer"/>
       <rdfs:range rdf:resource="&xsd;boolean"/>
   </owl:DatatypeProperty>
 <!-- http://www.debii.com/~naeem/ecommerceOntology#creditCardPayment -->
  <owl:DatatypeProperty rdf:about="#creditCardPayment">
       <rdfs:domain rdf:resource="#Customer"/>
       <rdfs:subPropertyOf rdf:resource="#makePayment"/>
       <rdfs:range rdf:resource="&xsd;boolean"/>
   </owl:DatatypeProperty>
   <!-- http://www.debii.com/~naeem/ecommerceOntology#customerName -->
   <owl:DatatypeProperty rdf:about="#customerName">
       <rdfs:domain rdf:resource="#Customer"/>
       <rdfs:range rdf:resource="&xsd;string"/>
   </owl:DatatypeProperty>
   <!-- http://www.debii.com/~naeem/ecommerceOntology#hasCustomerPolicy -->
  <owl:DatatypeProperty rdf:about="#hasCustomerPolicy">
       <rdfs:domain rdf:resource="#eShop"/>
       <rdfs:subPropertyOf rdf:resource="#hasPolicies"/>
       <rdfs:range rdf:resource="&xsd;anyURI"/>
   </owl:DatatypeProperty>
 <!-- http://www.debii.com/~naeem/ecommerceOntology#hasDiscountPolicy -->
   <owl:DatatypeProperty rdf:about="#hasDiscountPolicy">
       <rdfs:domain rdf:resource="#eShop"/>
       <rdfs:subPropertyOf rdf:resource="#hasPolicies"/>
       <rdfs:range rdf:resource="&xsd;anyURI"/>
   </owl:DatatypeProperty>
   <!-- http://www.debii.com/~naeem/ecommerceOntology#hasPolicies -->
   <owl:DatatypeProperty rdf:about="#hasPolicies"/>
  <!-- http://www.debii.com/~naeem/ecommerceOntology#hasProductCatalog -->
   <owl:DatatypeProperty rdf:about="#hasProductCatalog">
       <rdfs:domain rdf:resource="#eShop"/>
       <rdfs:subPropertyOf rdf:resource="#hasProducts"/>
       <rdfs:range rdf:resource="&xsd;anyURI"/>
   </owl:DatatypeProperty>
   <!-- http://www.debii.com/~naeem/ecommerceOntology#hasProductReviews -->
   <owl:DatatypeProperty rdf:about="#hasProductReviews">
       <rdfs:domain rdf:resource="#eShop"/>
       <rdfs:subPropertyOf rdf:resource="#hasProducts"/>
       <rdfs:range rdf:resource="&xsd;anyURI"/>
   </owl:DatatypeProperty>
   <!-- http://www.debii.com/~naeem/ecommerceOntology#hasProducts -->
   <owl:DatatypeProperty rdf:about="#hasProducts"/>
   <!-- http://www.debii.com/~naeem/ecommerceOntology#hasRefundPolicy -->
   <owl:DatatypeProperty rdf:about="#hasRefundPolicy">
       <rdfs:domain rdf:resource="#eShop"/>
       <rdfs:subPropertyOf rdf:resource="#hasPolicies"/>
       <rdfs:range rdf:resource="&xsd;anyURI"/>
   </owl:DatatypeProperty>
```

```
 <!-- http://www.debii.com/~naeem/ecommerceOntology#hasValue -->
 <owl:DatatypeProperty rdf:about="#hasValue">
     <rdfs:range rdf:resource="&xsd;boolean"/>
 </owl:DatatypeProperty>
<!-- http://www.debii.com/~naeem/ecommerceOntology#id -->
 <owl:DatatypeProperty rdf:about="#id">
     <rdfs:domain rdf:resource="#eShop"/>
     <rdfs:range rdf:resource="&xsd;integer"/>
 </owl:DatatypeProperty>
<!-- http://www.debii.com/~naeem/ecommerceOntology#makePayment -->
 <owl:DatatypeProperty rdf:about="#makePayment"/>
 <!-- http://www.debii.com/~naeem/ecommerceOntology#name -->
 <owl:DatatypeProperty rdf:about="#name">
     <rdfs:domain rdf:resource="#eShop"/>
 </owl:DatatypeProperty>
 <!-- http://www.debii.com/~naeem/ecommerceOntology#productID -->
 <owl:DatatypeProperty rdf:about="#productID">
     <rdfs:domain rdf:resource="#Products"/>
     <rdfs:range rdf:resource="&xsd;integer"/>
 </owl:DatatypeProperty>
 <!-- http://www.debii.com/~naeem/ecommerceOntology#productName -->
 <owl:DatatypeProperty rdf:about="#productName">
     <rdfs:domain rdf:resource="#Products"/>
     <rdfs:range rdf:resource="&xsd;string"/>
 </owl:DatatypeProperty>
 <!-- http://www.debii.com/~naeem/ecommerceOntology#productPrice -->
 <owl:DatatypeProperty rdf:about="#productPrice">
     <rdfs:domain rdf:resource="#Products"/>
     <rdfs:range rdf:resource="&xsd;float"/>
 </owl:DatatypeProperty>
 <!-- http://www.debii.com/~naeem/ecommerceOntology#regularCustomer -->
 <owl:DatatypeProperty rdf:about="#regularCustomer">
     <rdfs:domain rdf:resource="#Customer"/>
     <rdfs:range rdf:resource="&xsd;boolean"/>
 </owl:DatatypeProperty>
 <!-- http://www.debii.com/~naeem/ecommerceOntology#reviewedComments -->
 <owl:DatatypeProperty rdf:about="#reviewedComments">
     <rdfs:domain rdf:resource="#Reviews"/>
     <rdfs:range rdf:resource="&xsd;string"/>
 </owl:DatatypeProperty>
<!-- http://www.debii.com/~naeem/ecommerceOntology#reviewedRate -->
 <owl:DatatypeProperty rdf:about="#reviewedRate">
     <rdfs:domain rdf:resource="#Reviews"/>
     <rdfs:range rdf:resource="&xsd;string"/>
 </owl:DatatypeProperty>
 <!-- http://www.debii.com/~naeem/ecommerceOntology#slowToPay -->
 <owl:DatatypeProperty rdf:about="#slowToPay">
     <rdfs:domain rdf:resource="#Customer"/>
     <rdfs:range rdf:resource="&xsd;boolean"/>
 </owl:DatatypeProperty>
 <!--
 ///////////////////////////////////////////////////////////////////////////////////
 //
 // Classes
 //
 ///////////////////////////////////////////////////////////////////////////////////
  -->
 <!-- http://www.debii.com/~naeem/ecommerceOntology#Client -->
```

```
    <owl:Class rdf:about="#Client">
        <owl:equivalentClass rdf:resource="#Customer"/>
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.debii.com/~naeem/ecommerceOntology#Customer -->
    <owl:Class rdf:about="#Customer">
        <owl:equivalentClass rdf:resource="#Shopper"/>
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.debii.com/~naeem/ecommerceOntology#Items -->
    <owl:Class rdf:about="#Items">
        <owl:equivalentClass rdf:resource="#Products"/>
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.debii.com/~naeem/ecommerceOntology#Products -->
    <owl:Class rdf:about="#Products">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.debii.com/~naeem/ecommerceOntology#Reviews -->
    <owl:Class rdf:about="#Reviews">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.debii.com/~naeem/ecommerceOntology#Shopper -->
    <owl:Class rdf:about="#Shopper">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.debii.com/~naeem/ecommerceOntology#Store -->
    <owl:Class rdf:about="#Store">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.debii.com/~naeem/ecommerceOntology#eShop -->
    <owl:Class rdf:about="#eShop">
        <rdfs:subClassOf rdf:resource="#Store"/>
    </owl:Class>
    <!-- http://www.debii.com/~naeem/ecommerceOntology#physical -->
    <owl:Class rdf:about="#physical">
        <rdfs:subClassOf rdf:resource="#Store"/>
    </owl:Class>
    <!-- http://www.w3.org/2002/07/owl#Thing -->
    <owl:Class rdf:about="&owl;Thing"/>
</rdf:RDF>



<!-- Generated by the OWL API (version 2.2.1.974) http://owlapi.sourceforge.net -->

<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY products "http://www.BigW.com/products.owl#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY ecommerceOntology "http://www.debii.com/~naeem/ecommerceOntology#" >
]>
```

```
<rdf:RDF xmlns="http://www.BigW.com/products.owl#"
     xml:base="http://www.BigW.com/products.owl"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:products="http://www.BigW.com/products.owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:ecommerceOntology="http://www.debii.com/~naeem/ecommerceOntology#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">
    <owl:Ontology rdf:about="">
        <owl:imports rdf:resource="http://www.debii.com/~naeem/BigW"/>
    </owl:Ontology>
  <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Data properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->
   <!-- http://www.debii.com/~naeem/ecommerceOntology#productID -->
    <owl:DatatypeProperty rdf:about="&ecommerceOntology;productID">
        <rdfs:range rdf:resource="&xsd;integer"/>
    </owl:DatatypeProperty>
    <!-- http://www.debii.com/~naeem/ecommerceOntology#productName -->
    <owl:DatatypeProperty rdf:about="&ecommerceOntology;productName"/>
   <!-- http://www.debii.com/~naeem/ecommerceOntology#productPrice -->
    <owl:DatatypeProperty rdf:about="&ecommerceOntology;productPrice"/>
     <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->
    <!-- http://www.debii.com/~naeem/ecommerceOntology#Products -->
    <owl:Class rdf:about="&ecommerceOntology;Products"/>
    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Individuals
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->
    <!-- http://www.BigW.com/products.owl#rawMaterial -->
    <ecommerceOntology:Products rdf:about="#rawMaterial">
        <ecommerceOntology:productID rdf:datatype="&xsd;integer">1265</ecommerceOntology:productID>
        <ecommerceOntology:productPrice rdf:datatype="&xsd;float">236.3</ecommerceOntology:productPrice>
        <ecommerceOntology:productName rdf:datatype="&xsd;string"
            >Philips Top loader</ecommerceOntology:productName>
    </ecommerceOntology:Products>
</rdf:RDF>
<!-- Generated by the OWL API (version 2.2.1.974) http://owlapi.sourceforge.net -->
```

A.3 Process ontology in OWL/RDF format

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY travelOntology "http://www.semanticweb.org/ontologies/travelOntology.owl#" >
]>


<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/travelOntology.owl#"
     xml:base="http://www.semanticweb.org/ontologies/travelOntology.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:travelOntology="http://www.semanticweb.org/ontologies/travelOntology.owl#">
    <owl:Ontology rdf:about="http://www.semanticweb.org/ontologies/travelOntology.owl"/>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Datatypes
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.semanticweb.org/ontologies/travelOntology.owl#ProcessToProcess -->

    <owl:ObjectProperty rdf:about="&travelOntology;ProcessToProcess">
        <rdfs:domain rdf:resource="&travelOntology;Process"/>
        <rdfs:range rdf:resource="&travelOntology;Process"/>
    </owl:ObjectProperty>
```

```
<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Task-Constraint-Relationship -->

<owl:ObjectProperty rdf:about="&travelOntology;Task-Constraint-Relationship">
    <rdfs:range rdf:resource="&travelOntology;Constraints"/>
    <rdfs:domain rdf:resource="&travelOntology;Tasks"/>
</owl:ObjectProperty>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Task-DataItem-Relationship -->

<owl:ObjectProperty rdf:about="&travelOntology;Task-DataItem-Relationship">
    <rdfs:range rdf:resource="&travelOntology;DataItem"/>
    <rdfs:domain rdf:resource="&travelOntology;Tasks"/>
</owl:ObjectProperty>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Task-Resource-Relationship -->

<owl:ObjectProperty rdf:about="&travelOntology;Task-Resource-Relationship">
    <rdfs:range rdf:resource="&travelOntology;Resource"/>
    <rdfs:domain rdf:resource="&travelOntology;Tasks"/>
</owl:ObjectProperty>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Task-Task-Relationship -->

<owl:ObjectProperty rdf:about="&travelOntology;Task-Task-Relationship">
    <rdfs:range rdf:resource="&travelOntology;Tasks"/>
    <rdfs:domain rdf:resource="&travelOntology;Tasks"/>
</owl:ObjectProperty>



<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Data properties
//
///////////////////////////////////////////////////////////////////////////////////
 -->



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#METHOD -->

<owl:DatatypeProperty rdf:about="&travelOntology;METHOD"/>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#TYPE -->

<owl:DatatypeProperty rdf:about="&travelOntology;TYPE"/>
```

```
<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#X -->

<owl:DatatypeProperty rdf:about="&travelOntology;X"/>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Y -->

<owl:DatatypeProperty rdf:about="&travelOntology;Y"/>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Z -->

<owl:DatatypeProperty rdf:about="&travelOntology;Z"/>




<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////
 -->



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Constraints -->

<owl:Class rdf:about="&travelOntology;Constraints">
    <rdfs:subClassOf rdf:resource="&travelOntology;Tasks"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#DataItem -->

<owl:Class rdf:about="&travelOntology;DataItem">
    <rdfs:subClassOf rdf:resource="&travelOntology;Tasks"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Process -->

<owl:Class rdf:about="&travelOntology;Process"/>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Resource -->

<owl:Class rdf:about="&travelOntology;Resource">
    <rdfs:subClassOf rdf:resource="&travelOntology;Tasks"/>
```

```
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Situation -->

<owl:Class rdf:about="&travelOntology;Situation">
    <rdfs:subClassOf rdf:resource="&travelOntology;Tasks"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Tasks -->

<owl:Class rdf:about="&travelOntology;Tasks">
    <rdfs:subClassOf rdf:resource="&travelOntology;Process"/>
</owl:Class>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#action -->

<owl:Class rdf:about="&travelOntology;action">
    <rdfs:subClassOf rdf:resource="&travelOntology;Tasks"/>
</owl:Class>



<!-- http://www.w3.org/2002/07/owl#Thing -->

<owl:Class rdf:about="&owl;Thing"/>



<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Individuals
//
///////////////////////////////////////////////////////////////////////////////////
 -->



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Authority_to_approve_travel_ -->

<owl:NamedIndividual rdf:about="&travelOntology;Authority_to_approve_travel_">
    <rdf:type rdf:resource="&travelOntology;Process"/>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#Travel_booking_for_staff_and_associates -->

<owl:NamedIndividual rdf:about="&travelOntology;Travel_booking_for_staff_and_associates">
    <rdf:type rdf:resource="&travelOntology;Process"/>
</owl:NamedIndividual>
```

```
<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#airFairPayment -->

<owl:NamedIndividual rdf:about="&travelOntology;airFairPayment">
    <rdf:type rdf:resource="&travelOntology;DataItem"/>
    <X></X>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#approve -->

<owl:NamedIndividual rdf:about="&travelOntology;approve">
    <rdf:type rdf:resource="&travelOntology;Tasks"/>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#booking -->

<owl:NamedIndividual rdf:about="&travelOntology;booking">
    <rdf:type rdf:resource="&travelOntology;Tasks"/>
    <X></X>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#contactTravelConsultant -->

<owl:NamedIndividual rdf:about="&travelOntology;contactTravelConsultant">
    <rdf:type>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&travelOntology;Task-Resource-Relationship"/>
            <owl:someValuesFrom rdf:resource="&travelOntology;Tasks"/>
        </owl:Restriction>
    </rdf:type>
    <METHOD></METHOD>
    <Z></Z>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#creditCard -->

<owl:NamedIndividual rdf:about="&travelOntology;creditCard">
    <rdf:type rdf:resource="&travelOntology;Resource"/>
    <TYPE></TYPE>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#director -->

<owl:NamedIndividual rdf:about="&travelOntology;director">
    <rdf:type rdf:resource="&travelOntology;Resource"/>
</owl:NamedIndividual>
```

```
<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#enterRecord -->

<owl:NamedIndividual rdf:about="&travelOntology;enterRecord">
    <rdf:type rdf:resource="&travelOntology;Tasks"/>
    <X></X>
    <TYPE></TYPE>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#examineProfile -->

<owl:NamedIndividual rdf:about="&travelOntology;examineProfile">
    <rdf:type rdf:resource="&travelOntology;Tasks"/>
    <Y></Y>
    <X></X>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#facilitator -->

<owl:NamedIndividual rdf:about="&travelOntology;facilitator">
    <rdf:type rdf:resource="&travelOntology;Resource"/>
    <Y></Y>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#fax -->

<owl:NamedIndividual rdf:about="&travelOntology;fax">
    <rdf:type rdf:resource="&travelOntology;Tasks"/>
    <Y></Y>
    <METHOD></METHOD>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#filled -->

<owl:NamedIndividual rdf:about="&travelOntology;filled">
    <rdf:type rdf:resource="&travelOntology;DataItem"/>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#finishToFinish -->

<owl:NamedIndividual rdf:about="&travelOntology;finishToFinish">
    <rdf:type>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&travelOntology;Task-Task-Relationship"/>
            <owl:someValuesFrom rdf:resource="&travelOntology;Tasks"/>
        </owl:Restriction>
```

```
    </rdf:type>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#finishToStart -->

<owl:NamedIndividual rdf:about="&travelOntology;finishToStart">
    <rdf:type>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&travelOntology;Task-Task-Relationship"/>
            <owl:someValuesFrom rdf:resource="&travelOntology;Tasks"/>
        </owl:Restriction>
    </rdf:type>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#hotelPayment -->

<owl:NamedIndividual rdf:about="&travelOntology;hotelPayment">
    <rdf:type rdf:resource="&travelOntology;DataItem"/>
    <X></X>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#lateRecordEntry -->

<owl:NamedIndividual rdf:about="&travelOntology;lateRecordEntry">
    <rdf:type rdf:resource="&travelOntology;DataItem"/>
    <X></X>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#onlineBooking -->

<owl:NamedIndividual rdf:about="&travelOntology;onlineBooking">
    <rdf:type rdf:resource="&travelOntology;Tasks"/>
    <X></X>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#proceedForPayment -->

<owl:NamedIndividual rdf:about="&travelOntology;proceedForPayment">
    <rdf:type rdf:resource="&travelOntology;Tasks"/>
    <X></X>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#startToFinish -->

<owl:NamedIndividual rdf:about="&travelOntology;startToFinish">
```

```
    <rdf:type>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&travelOntology;Task-Task-Relationship"/>
            <owl:someValuesFrom rdf:resource="&travelOntology;Tasks"/>
        </owl:Restriction>
    </rdf:type>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#startToStart -->

<owl:NamedIndividual rdf:about="&travelOntology;startToStart">
    <rdf:type>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&travelOntology;Task-Task-Relationship"/>
            <owl:someValuesFrom rdf:resource="&travelOntology;Tasks"/>
        </owl:Restriction>
    </rdf:type>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#submit -->

<owl:NamedIndividual rdf:about="&travelOntology;submit">
    <rdf:type rdf:resource="&travelOntology;Tasks"/>
    <X></X>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#travelConsultant -->

<owl:NamedIndividual rdf:about="&travelOntology;travelConsultant">
    <rdf:type rdf:resource="&travelOntology;Resource"/>
    <Z></Z>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#travelForm -->

<owl:NamedIndividual rdf:about="&travelOntology;travelForm">
    <rdf:type rdf:resource="&travelOntology;DataItem"/>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/ontologies/travelOntology.owl#traveller -->

<owl:NamedIndividual rdf:about="&travelOntology;traveller">
    <rdf:type rdf:resource="&travelOntology;Resource"/>
    <X></X>
</owl:NamedIndividual>
```

```
</rdf:RDF>


<!-- Generated by the OWL API (version 3.2.3.1824) http://owlapi.sourceforge.net -->
```

Appendix B - Selected Publications arising from this thesis

Knowledge-Based Systems 32 (2012) 9–27

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Web@IDSS – Argumentation-enabled Web-based IDSS for reasoning over incomplete and conflicting information

Naeem Khalid Janjua, Farookh Khadeer Hussain *

Digital Ecosystem and Business Intelligence Institute, Curtin University, Perth, WA, Australia

ARTICLE INFO

Article history:
Available online 24 September 2011

Keywords:
IDSS
Argumentation
DSS
Reasoning
Defeasible reasoning
Semantic web

ABSTRACT

Over the past few decades, there has been a resurgence of interest in using high-level software intelligence for business intelligence (BI). The objective is to produce actionable information that is delivered at the right time, easily comprehendible and exportable to other software to assist business decision-making processes. Although the design and development of decision support systems (DSS) has been carried out for over 40 years, DSS still suffer from many limitations such as poor maintainability, poor flexibility and less reusability. The development of the Internet and WWW has helped information systems to overcome those limitations and Web DSS is now an active area of research in business intelligence, impacting significantly on the way information is exchanged and businesses are conducted. However, to remain competitive, companies rely on business intelligence (BI) to continually monitor and analyze the operating environment (both internal and external), to identify potential risks, and to devise competitive business strategies. However, the current Web DSS applications are not able to reason over information present across organizational boundaries which could be incomplete and conflicting. The use of an argumentation-based mechanism has not been explored to address such shortcomings in Web DSS. Argumentation is a kind of commonsense reasoning used by human beings to reach a justifiable conclusion when available information is incomplete and/or inconsistent among participants. In this paper, we propose and elaborate in detail a conceptual framework and formal argumentation-based semantics for Web enabled Intelligent DSS (Web@IDSS). We evaluate the use of argumentative reasoning in Web DSS with the help of a case study, prototype development and future directions. Applications built according to the proposed framework will provide more practical, understandable results to decision makers.

1. Introduction

Business intelligence (BI) is the use of high-level software intelligence to produce actionable information delivered at the right time, and that is immediately accessible, easily comprehendible and exportable to other software to assist the business decision-making process [1]. Over the past few decades, advancements in Internet, World Wide Web (WWW) and Artificial Intelligence (AI) technologies have engendered a resurgence of interest in the use of software intelligence for business applications. While the term BI is relatively new, computer-based business intelligence systems go back, in one form or another, for close to 40 years [2]. BI as a term has been used interchangeably with decision support systems (DSS), executive information systems, and management information systems [3]. DSS are the core of BI and are defined as a broad category of interactive computer-based information systems for informing and supporting decision makers and are intended to improve and expedite processes by which people make and communicate decisions [2]. The advancements in Internet technologies have enabled the development of new DSS architectures, from stand-alone DSS to Group Decision Support Systems (GDSS), OLAP and data warehouse technology in DSS [4].

Although DSS design and development has spanned over 40 years, DSS still suffer from many limitations. Firstly, they have problems of poor maintainability, poor flexibility and less reusability [5]. Secondly, most DSS applications [6] are not equipped with inference capabilities; hence, much of the implicit information remains undiscovered thereby resulting in sub-optimal business decisions and business strategies. Thirdly, they are more focused on data within an organization for extracting business intelligence and ignore the fact that the underlying information which could be used to derive business intelligence for formulating business strategies could be present outside the organization's boundaries, e.g. suppliers' business policies or information present in third party portals such as Amazon.com [7].

* Corresponding author.
E-mail addresses: naeem.janjua@gmail.com (N.K. Janjua), farookh.hussain@cbs.curtin.edu.au (F.K. Hussain).

0950-7051/$ - see front matter Crown Copyright © 2011 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2011.09.009

The advent and widespread adoption of the World Wide Web has led to the development of Intelligent Web-DSS [8]. The WWW technologies have the potential to reduce some of the limitations of the DSS, making them flexible, maintainable and reusable [4]. As a result, the Web-based Decision Support Systems (Web DSS) have become a new trend in DSS research, and organizations are focusing on applying Web technologies to improve the functionality of DSS applications [9]. One of the immediate benefits of Web-DSS is the ability to publish and share the useful information in a wide range of enterprise decision-making processes.

However, to remain competitive, companies rely on business intelligence (BI) for the ongoing monitoring and analysis of the operating environment in order to identify potential risks, and to devise competitive business strategies [10]. Therefore, it has become pivotal for organizations to make use of Intelligent DSS (IDSS) for getting business insights. A critical step in this process is modelling and representation of the business rules of organizations reflecting changes in business policies, procedures or other constraints over a period of time to run their business activities. Much research is being carried out on the design and development of business rules-based Intelligent systems such as Web-DSS for product recommendation, auctions, identification of requirements, vendor selection, negotiation, agent communication and information integration leading to great impact of IDSS on Web e-markets [11–15]. However, like traditional DSS, these systems perform reasoning under certain assumptions such as:

1. The given problem can be fully addressed with available information (solution to the problem lies within the available information). In order to elucidate it, let us consider an example. An organization wants to improve its product and they believe that all the information it holds internally is sufficient to identify the issues and improve the product's quality. The organization ignores any information present outside of its own boundaries.
2. The information or specification for business rules for decision making is consistent. In other words, they assume that there will be no conflicting rules during the decision-making process.
3. New information will be consistent with the already available information or specifications.
4. New information does not lead to retraction of previous conclusions.

Because of the limitations mentioned above, the existing IDSS systems rely on analyzing a company's internal data and do not take into account a great deal of useful information present on the Web outside of the organization's boundaries for decision making purposes. Especially with the emergence of Web 2.0, the large number of customers generating information such as product reviews can often provide useful information which will guide a company to improve its products; thus, customer opinion becomes a pivotal source of information for effective decision making processes. Additionally, during business decision making, it would be helpful to take into account the external operating business environments (such as publically available policies of the prospective interacting party, etc.). Usually, the incorporation of polices during the inference mechanism process might lead to conflicts among the rules driving the inference mechanism.

The current generation of Web decision support systems is not able to represent and reason over incomplete and inconsistent information, irrespective of whether this information emerges from within the organization or outside it [16]. It is important to note here that when we refer to 'information' we mean the following:

- the business polices or rules governing the inference mechanism, and
- the data over which the inference mechanism is being applied.

There has been much discussion in the literature on the development of Web-DSS systems that can handle conflicts among rules by defining priorities at compile time [17–19]. However, such Web-DSS systems suffer from two main limitations:

- They provide formalism to represent and handle only individual preferences in the form of priorities among the conflicting rules. However, DSS systems are subject to inconsistencies deriving from multiple sources and multiple users; therefore, it is not possible to define priorities in advance in order to resolve conflict among rules derived from multiple sources/users.
- The use of these priorities is usually embedded in the derivation mechanism and competing rules are compared individually during the derivation process. Therefore, the derivation notion is bound to one single comparison criterion. In such scenario, the explanation of the results is based on single criteria only and fails to take into account the multiple factors important for decision making.

In contrast, if we look at Artificial Intelligence research, the challenge of incomplete and conflicting knowledge representation and reasoning over it in software agents has been addressed using logic-based argumentation formalisms, i.e. defeasible logic programming (DeLP) [20]. Argumentation formalisms are defeasible reasoning systems which work by considering the reasons that lead to a given conclusion (claim) through a piece of reasoning (the supporting arguments) and potential challenges (counter arguments) for accepting the conclusion [21]. Argumentation plays a pivotal role in identifying and organizing what can be justifiably concluded, and presenting it systematically to human users or merging it with the justified conclusions of other machines in the absence of complete or accurate information.

As suggested by Carlsson and Turban [16] and Shim et al. [4], there is the need to design and develop Intelligent Web-DSS in order to transform incomplete information into useful knowledge alongside qualitative insights. Therefore, the limitations of current Web DSS, i.e. the capability to handle incomplete and conflicting information, can be addressed by applying logic-based argumentation formalisms.

In order to address this critical shortcoming of the existing Web-DSS systems, we propose Argumentation-enabled Web IDSS (Web@IDSS) that exploits the power of logic-based argumentation formalism for reasoning and representation of incomplete and conflicting knowledge. The system is based on a hybrid reasoning approach: forward chaining (data-driven) for the construction of arguments, and backward chaining (goal-driven) for evaluation of conclusions. For the forward chaining process, the Rete algorithm is used to generate all possible arguments. Conflicts between arguments are detected and resolved during the backward chaining process.

From the BI perspective, such powerful Web-DSS systems would be able to carry out reasoning on data across organizational boundaries. This is a potential area of growth and research in Intelligent Web-DSS as depicted in Fig. 1. This will enable organizations to analyze the information present across the boundaries of the organization for decision support, eventually adding to BI.

The rest of this paper is structured as follows: Section 2 presents a review of literature, elaborating and focusing on logic based implementations of IDSS with their limitations. Section 3 describes a case study related to reasoning over incomplete and inconsistent information spanning across organizational boundaries. Section 4 discusses the argumentation and its current application in the area of DSS. It also discusses the defeasible logic programming (DeLP) and its limitations in the context of using it in enterprise for BI. Section

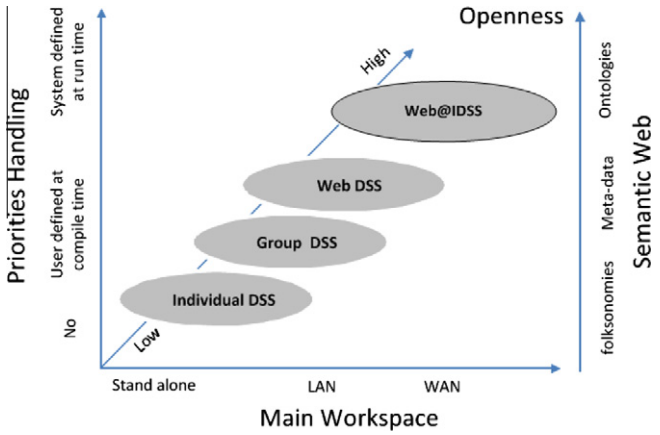


Fig. 1. Evolution towards Argumentation-enabled Web IDSS [extended from Lee and Chung [22]].

5 describes the proposed conceptual framework. This is followed by Section 6 which elaborates upon the argumentative engine in detail. In this section we formally define the functionality of the argumentative engine. Section 7 describes the translation of RuleML policies and RDF/XML data to DeLP rules and facts respectively, followed by Implementation and Prototype development in Section 8. Section 9 presents the conclusion and proposed future work.

2. Review of literature

Information is essential for any decision making process and processing it manually is becoming increasingly difficult because of the following trends [23]:

- Information technology and sophisticated analysis provided by intelligent information analysis, promote accurate and reliable decision making.
- There has been an explosion of data in recent years also termed a tsunami of data which may traverse organizational boundaries and decisions need to be based on this huge corpus of data [24,25].
- Rapid access to remote information such as information about partners, experts, etc. is now possible.

The traditional DSS are inflexible in responding to dynamic situations and lack the ability to make judgments, unlike humans who may be able to make decisions even in situations where information may be deficient. Hence, we now have the special category of systems known as decision support systems (DSS) which are auxiliary systems in that they have not been developed to replace the skilled decision makers, but rather, they assist in a wide range of enterprise decision-making processes. DSS provide a framework that allows decision makers to view alternatives and make good decisions in a timely manner [26,27,2].

The importance of Web-based DSS in business applications has been identified by a number of researchers over a period of time [28–30]. According to Bhargava et al. [31], there are two ways in which the DSS community adopted the internet and World Wide Web: firstly as Web-based Decision support where the entire DSS application is implemented using Web technologies; secondly, as Web-enabled Decision support where the key parts of DSS like a database remain on a legacy system, but the application can be accessed using a Web technology component and displayed in a browser.

There is a plethora of research on DSS and IDSS. It is not within the scope of this paper to review and analyze all the existing DSS and IDSS. Hence, we restrict ourselves to logic-based IDSS. We classify logic-based IDSS into two broad categories, namely, 'fuzzy logic based IDSS' and 'Defeasible logic-based IDSS'.

2.1. Fuzzy logic based IDSS

A number of researchers use fuzzy logic-based quantitative approaches to group decision-making processes. Subsorn et al. [32] proposed a Web-based group decision support system framework to deal with imprecise decision-making problems. The framework is based on a fuzzy analytic hierarchy process for group decision-making. The framework enables group members to develop satisfactory group solutions and allows group leaders to form the final/acceptable, satisfactory group solutions. Ma et al. [33] proposed 'Decider', a fuzzy multi-criteria group decision-making (MCGDM) process model that aims to support preference-based decisions over the available alternatives that are characterized by multiple criteria in a group. The model can handle information expressed in linguistic terms, Boolean values, as well as numeric values to assess and rank a set of alternatives within a group of decision makers. Noor-E-Alam et al. [34] also addressed the issue of multi-criteria decision-making (MCDM) involving multiple experts and pointed out that the participation of many experts makes the conflict aggregation process difficult. They developed a decision support system (DSS) based on types of fuzzy based conflict aggregation algorithms, namely, a possibility measure and averaging conflict aggregation. Yue [35] addressed the issue of multiple attribute decision-making (MADM) and developed an algorithm for determining weights of decision makers within a group decision environment, in which the information regarding each individual decision is expressed by a matrix in interval numbers. He also defined positive ideal and negative ideal solutions of group opinion, the separation measures and the relative closeness from the positive ideal solution. Cabrerizo et al. [36] used fuzzy logic to address the issue of consensus building among experts when information is incomplete. They developed a consensus model to address the group decision-making problems with incomplete unbalanced fuzzy linguistic information. The working of the model is supported by consistency and consensus measures, and with the help of a feedback mechanism, personalized advice is provided to the experts for modifications to their unbalanced fuzzy linguistic preference relations. Similarly, efforts are being made to represent the results of the decision-making process to the end user in an easily comprehendible form such as that of Li et al. [37] who proposed a visualized information retrieval engine based on fuzzy control.

The aforementioned fuzzy logic-based IDSS, being quantitative, have been criticized for their inability to generate easy-to-understand and logically clear results for justification purposes. These approaches follow monotonic behavior whereby once a conclusion has been drawn, it cannot be retracted. Additionally, they lack inference reasoning capability over conflicting information; for BI, we need such inference mechanisms.

2.2. Defeasible logic based Web IDSS

Nute [38] highlighted the importance of defeasible reasoning in decision support systems and developed logic for defeasible reasoning by extending Prolog. The new logic is comprised of facts and presumption, absolute rules and defeasible rules, and introduced another kind of weak rule known as a 'defeater'. Causey [39] developed 'EVID'; system for interactive defeasible reasoning and Johnston and Governatori [40] developed an algorithm that integrates defeasible logic into a decision support system by automatically deriving its knowledge from databases of precedents.

Dr Prolog [17] is a prolog-based implementation for carrying out defeasible reasoning on the Web. It provides declarative system support rules, facts, ontologies, RuleML, and both monotonic and non-monotonic rules. It takes into consideration both open world and closed world assumptions and provides features for reasoning with inconsistencies. The system provides a number of variants such as ambiguity blocking, ambiguity propagation and conflicting literals. Defeasible theories are imported in defeasible logic or RuleML syntax and translated into logic programs with the help of a logic translator. The reasoning engine compiles the logic programs and the meta-program which corresponds to the DL version that the user selects (ambiguity blocking/ propagating), and evaluates the answers to the user's queries. They extended RuleML DTDs to represent defeasible theories in XML format. Dr Brokering [41] is a Dr-Prolog-based software agent implementation to address the problem of brokering and matchmaking; i.e. how a requester's requirements and preferences can be matched against a set of offerings collected by a broker.

Dr-Device [42,18] is a CLISP-based defeasible reasoning implementation provided with a VDR-Device reasoning system, RDF loader/translator and rule loader/translator component. The VDR-Device is an integrated development environment equipped with a graphical front end used for deploying defeasible rules on top of RDF schema ontologies. The rule base is initially submitted to the rule loader which transforms the rules into CLISP-like syntax through an XSLT stylesheet. The resulting program is forwarded to the rule translator where defeasible logic rules are compiled into a set of CLISP production rules. In parallel, the RDF downloader downloads the RDF documents and translates them into CLISP objects according to the RDF-to-Object scheme. The reasoning system performs inference on transited RDF metadata using defeasible rules and generates the objects that constitute the result of the initial rule program. The RDF extractor exports the resulted objects in the form of RDF/XML to the user. Dr-Device is implemented in Jess and integrates well with RuleML and RDF. Compared to prolog, Dr-Device supports only one variant: ambiguity blocking. At present, it does not support OWL ontologies. In addition, Dr-Prolog uses logic programs with well-founded semantics, which is formally equivalent to the formal model. In contrast, Dr-Device uses the logic meta-program as a guiding principle, but there is no formal proof of the correctness of the implementation. On the other hand, Dr-Device has the relative advantage of easier integration with mainstream software technologies.

Sweetjess [19] is another defeasible reasoning system based on Jess and closely resembles courteous logic programs. It integrates well with RuleML but it can perform reasoning only in DML + OIL ontologies and not on RDF data as does Dr-Device and Dr-Prolog. However, it allows for procedural attachment and it implements only one reasoning variant. Moreover, it imposes a number of restrictions on the programs so that it can map on Jess.

Table 1 compares different defeasible logic-based implementations. It is evident that they provide either data-driven reasoning or goal-driven reasoning. Data-driven is used to move from current facts to conclusion, whereas goal-driven reasoning is backward chain reasoning used to move from conclusion to the facts. In the case of semantic Web-DSS, both types of reasoning are needed: that is, data-driven reasoning to create a path from initial facts to conclusion and goal-driven reasoning to identify reasons and justifications for a particular conclusion. Another drawback is that they define individual preference at compile time; i.e. the user decides the priorities between the conflicting rules. Additionally, all of them provide only textual explanation to the end user and the output is not exportable in AIF format.

3. Case study

As depicted in Fig. 2, Mr. David is the marketing manager of Organization A. He is responsible for formulating and suggesting business strategies to increase the sale of the company's products and revenue (of the existing and new products) to the Chief Executive Officer of Organization A. Organization A intends to manufacture a new product (say Product B). One of the important aspects that Mr. David identifies is "the greater the discount that A receives from the supplier, the cheaper would be the new product", and negotiation plays an important part in securing the maximum discount. Mr. David identifies that the materials for manufacturing the product will be sourced from 'N' different suppliers, each of whom offers varying discount levels. Mr. David would like to formulate a strategy for manufacturing Product B by individually analyzing the business policies of each supplier against his company requirements along with feedback about the raw materials supplied by different companies. Finally, he would like to be able to justify the formulated strategy to a higher authority such as the CEO of Organization A.

As we can see from the above case study, an IDSS to accomplish Mr. David's business requirements would require an interface to define his requirements in the form of business rules such as 'Purchase product from supplier only if product feedback is good' and certain facts or information to realize those rules. An example of fact or information could be a type of feedback (washingMachine, good) which means that the feedback for washing machine is good. The IDSS should also provide an interface to download the suppliers' products information and suppliers' public policies against the possible discount on products and their shipment. Additionally, IDSS should be able to download feedback or reviews against the suppliers' products from a third party forum like Amazon and take that into consideration in the decision-making process. In situations where business policies may be incomplete or negotiation with the providers is needed to resolve some conflicting interests with the supplier, a system should be able to cater for these and provide a means of resolving these conflicts, with justified

Table 1
Comparison of defeasible logic based Web IDSS applications.

	Dr-Prolog [17]	Dr-Device [42,18]	Situated Courteous logic [19]
Language	Prolog	JESS	JESS
Logic	Defeasible logic	Defeasible logic	Situated Courteous logic
Semantic data	RDFS/OWL	RDF	DAML + OIL
Rules representation	RuleML	RuleML	RuleML
Incomplete knowledge representation	Yes	Yes	Yes
Conflict representation	Yes	Yes	Yes
Data-driven reasoning	No	Yes	Yes
Goal-driven reasoning	Yes	No	No
Conflict resolution	User defined priorities at compile time	User defined priorities at compile time	User defined priorities at compile time
Explanation	Textual	Textual	Textual
AIF reification	No	No	No

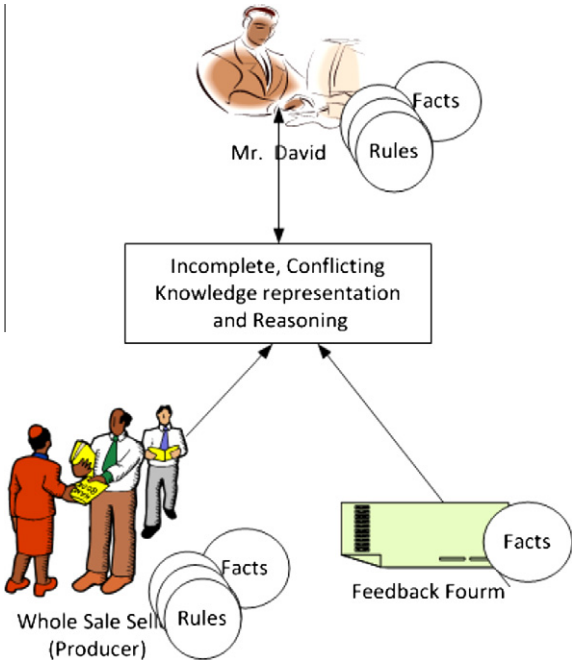


Fig. 2. Web@IDSS reasoning over data across organization boundaries.

explanation, during the reasoning process. Finally, IDSS should be able to export the result of the reasoning process to other software for any further processing so that decisions can be communicated to higher authorities effectively.

Here, we formalize David's requirements as follows:

- A declarative language for specifications of business requirements of an organization.
- Language having the capability of representing incomplete and conflicting information (i.e. business rules and data).
- A mechanism for sharing and reusing business rules over the Web.
- An inference mechanism that can perform reasoning pertaining to incomplete and conflicting information in the knowledge base.
- Justifiable explanation of the reasoning results.
- Ability to export results to other software systems.

Assumption.

- Organization A, Supplier and Feedback forum share a common vocabulary defined in RDF/XML format and the predicates defined in the vocabulary are used for specification of business rules and policies.

4. Argumentation

Argumentation is the study of effective reasoning in the way that humans deal with conflicting information by taking into account arguments and counter-arguments relevant to a certain issue. Argumentation is inherently a process rather than an instant picture and the building blocks of argumentation are arguments and the relationship between those arguments [43]. According to Walton [44], an argument is a set of statements (propositions) made up of three parts: a conclusion, a set of premises, and inference from premises to conclusion. Dung [45] proposed a very influential semantic foundation for an argumentative framework based on the notion of acceptability of the arguments. He defined an argumentation framework that emerged from logic programming, as a pair AF = ⟨AR,attacks⟩ where AR is set of the arguments and attack is the binary relation on AR i.e. attack $\subseteq$ AR *X* AR, where *X* represented attack relationship between arguments. Different attempts have been made to extend Dung's framework with the notion of joint attacks [46], integration of meta-level argumentation about preferences between arguments to add more semantics to attack the relationship between arguments [47] and value-based argumentation framework in order to quantify the strength of arguments and discuss the possibility of persuasion in the face of uncertainty and disagreement [48]. It also plays an important role in agent-based applications [49], agents' negotiations [50], belief revision [51] and learning in virtual communities [52]. Similarly, argumentation has been exploited in the development of different systems such as OSCAR [53], IACAS [54] and critics and recommender systems [55].

4.1. Argumentation based IDSS

Argumentation formalism has been used in the past by different researchers in DSS for practical reasoning purposes. Morge [56] proposed a DSS based on abductive reasoning which helps the decision maker to select a business location after evaluating different alternatives actions; it suggests some solutions and provides an interactive and intelligible explanation of those choices. To further develop DSS, Morge [56] used a logic language to represent knowledge, goals and actions with quantitative priorities attached to them to represent likelihood of knowledge, preferences between goals and expected utilities of actions respectively. Similarly, Chesnevar et al. [55] identified that the current critic and recommender systems are incapable of dealing with the defeasible nature of information. They presented a novel approach to integrating DSS, using critics and recommender systems with a defeasible argumentation framework to enhance the practical reasoning capabilities of such systems. Vasine [57] presented a confidence system for solving real-world problems with argumentation semantics. He designed and developed a system using a two-dimensional confidence vector using argumentation system i.e. Calvin. Additionally, attempts have been made to introduce ontologies in argumentative decision support systems [58,59].

Although argumentation formalism has been developed to bring enhanced intelligence to DSS, it needs to be implemented in e-business for BI. The applications discussed above exploit the power of argumentation but they cannot adequately provide decision support for BI for the following reasons:

- No semantics are available for data-driven reasoning over incomplete and inconsistent information.
- No tools are available that can provide logic-based applications interoperability with Web technologies standards like RuleML and RDF/XML, and generate graphical results in the form of reasoning chains which would give a more intuitive explanation of results to the end user.

4.2. Defeasible logic programming (DeLP) and its limitations

DeLP uses the argumentation formalism for reasoning over contradictory information by identifying conflicting information in the knowledge base and applying the dialectical process to decide which information prevails during the argumentative reasoning process. Some of the existing formalisms on reasoning (using conflicting rules) define explicit priorities among rules and use these priorities to decide between competing conclusions. The use of these priorities is usually embedded in the derivation mechanism and competing rules are compared individually during the derivation process. In such formalisms, the derivation notion, i.e. rules execution pattern, is bound to one single comparison criterion. With DeLP, in order to decide between competing conclusions,

the arguments that support the conclusions are compared. Thus, the comparison criterion is independent of the derivation process, and could be replaced in a modular way.

An interpreter of DeLP has been implemented in Prolog, and can be used through the Web (see http://cs.uns.edu.ar/~ajg/DeLP.html). Also, an abstract machine called JAM (Justification Abstract Machine) has been designed for the implementation of DeLP, as an extension of the Warren's Abstract Machine (wam). For more details about DeLP, please refer to Garcia and Simari [20].

Although DeLP addresses the challenge of incomplete and conflicting knowledge representation and reasoning, it cannot be used as such for the development of Web IDSS because of following limitations [60]:

1. DeLP uses backward chaining or goal-driven reasoning only, whereas most of the reasoning in DSS is primarily data-driven or forward chain reasoning.
2. There is no tool available for translating RuleML format into DeLP format.
3. There is no tool available that allows the business user to define rules and perform argumentative reasoning over incomplete or inconsistent information in the business domain.
4. There is no tool that provides proof explanation via graphical representation of reasoning chains generated by an argumentative reasoner for traversal by end users.

5. Proposed conceptual framework

In this section, we elaborate on the proposed conceptual framework for Web@IDSS for carrying out argumentative reasoning about incomplete and inconsistent information spanning organizational boundaries. As mentioned previously, we use the term 'information' to refer to the:

- business polices or rules governing the inference mechanism; and
- data to which the inference mechanism is being applied.

The proposed framework uses DeLP as knowledge representation and reasoning language with certain extensions to overcome the limitations of DeLP defined in Section 4.2. Fig. 3 depicts the proposed conceptual framework. The key components of our proposed framework are as follows:

1. **Data pre-processing and business rules translation**
   Business rules can be used to express computational or business logic, policies, etc. in Web applications. Organizations are increasingly publishing their public business polices in the form of RuleML via the Web. This module provides the functionality of parsing and translating business rules defined in RuleML notation to DeLP rules and saves them in the knowledge base. Similarly, the data in RDF/XML format is processed and transformed into DeLP facts.
2. **Working memory and knowledge base**
   A collection of facts is called a 'working memory' and a collection of rules is called a 'knowledge base'. In the proposed framework, the knowledge base comprises strict rules and defeasible rules. The system provides an interface whereby the end user can define and update DeLP facts and DeLP rules in the working memory and knowledge base respectively. The end user is given an interface with which to query the knowledge base.
3. **Forward chain reasoning for arguments construction**
   In the proposed framework, we are extending the DeLP rule engine with data-driven rules. We are using the Rete [61] based algorithm without any conflict resolution strategy. The Rete algorithm involves two steps. The first is the compilation of rules in the form of a network called a Rete network. The second step is the matching phase, in which the rule engine matches the conditions of the rules in the knowledge base against the DeLP facts in the working memory. For each match, a rule instance is created and put into the Active rule set. Once the matching phase is completed, instances of all the rules in Active rule should be fired. Firing the rule instance will:
   - add a new fact to the working memory, and
   - add instance of rule to the active argument set.

   The matching phase starts again and only the new inferred facts filter through the compiled rules network and result in the construction of an active rule set and the process continues. The process will stop when no more rules match the new inferred facts. This whole phase is known as 'arguments construction'. A key issue to be noted here is that such new inferred facts may conflict with the existing knowledge base. The purpose is to retain conflicting information instead of eliminating it, in order to obtain a better insight when deciding on business strategies.
4. **Conflict detection and resolution**
   Once the argument construction process is complete, the conflict detection, resolution and justified explanation phase is initiated. The process of argumentation starts when an argument may be defeated by other arguments, since counter arguments are also arguments which in turn may be defeated, and process results in the construction of dialectical trees. This is an interesting property of the argumentation approach which involves dialectical proof procedures that are quite close to the process used by humans to discuss an issue. This similarity with human-style discussions gives argumentation an advantage that can be useful in many contexts. As a result, the process of dialectical analysis will lead to the establishment of dynamic priorities between conflicting arguments. The last step of an argumentation process is the construction of reasoning chains.
5. **Export reasoning results**
   In Web@IDSS, we make use of Argument Interchange Format (AIF) to export the output of the argumentation process (i.e. reasoning chains), so that it can be shared with other argumentative systems and even published over the Web. The Argument Interchange Format (AIF) is the result of an international effort to develop a representational mechanism for exchanging argument resources between research groups, tools, and domains using a semantically rich language [62]. This enables the output of Web@IDSS to be merged with the justified conclusions of other machines. This makes possible the sharing of results with other users or systems.
6. **User interface**
   User interface is the graphical representation of reasoning engine output for the end user. The user interface component is responsible for representing the argumentation process and justified conclusion to the user in the form of an inverted tree-like structure, and the user is able to interact with and query the results.

6. Argumentation engine

A rule-based Web DSS is an application of rule-based technologies on the Web that provides some form of artificial intelligence in Web applications. It consists primarily of a set of inferential rules about behavior, a collection of facts known as working memory and a rule interpreter. The rules, also termed 'productions', are a basic representation found to be useful in automated planning, expert systems and action selection. A production system provides the mechanism necessary to execute productions in order to achieve some goal for the system. Productions consist of two parts: a rule body and a head. If a production's rule body matches the current state of the world, then the production is said to be fired and new inferred facts are added to the working memory.

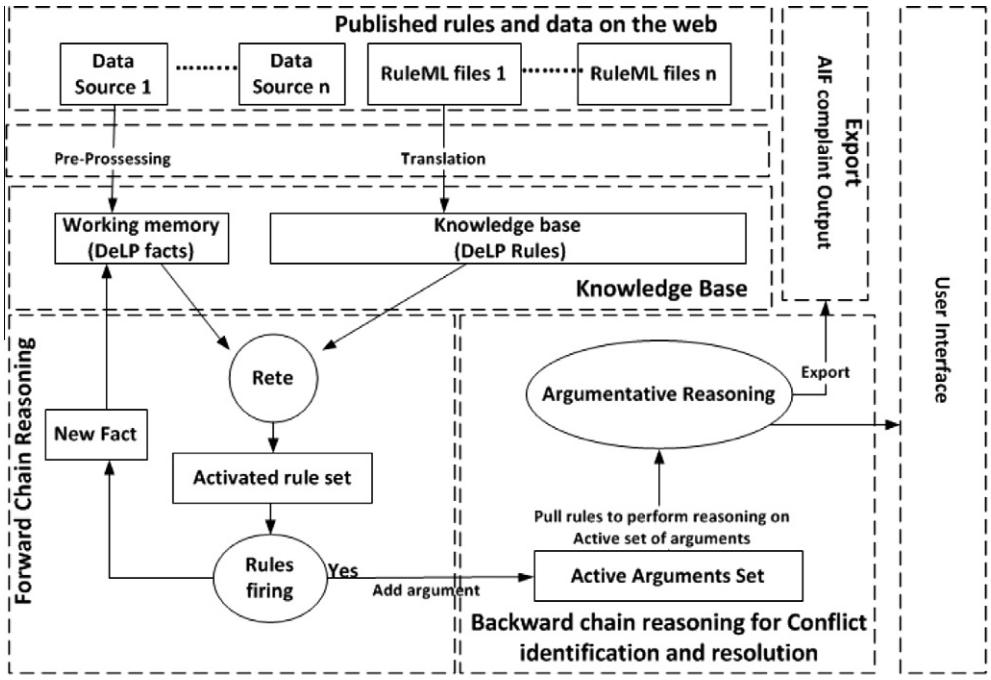


Fig. 3. Conceptual framework of Web@IDSS.

After a thorough review and analysis of the existing literature [63,20,64,65], in the following section, we provide the formal argumentative semantics for rule-based Web@IDSS.

6.1. Formal syntax and semantics

(**Language**). A language is a set Φ containing a set of predicates $\mathcal{P}$, a set of functions $\mathcal{F}$, an infinite set of variables $\mathcal{X}$, a finite set of symbols $\mathcal{S}$, and a set of labels $\mathcal{L}$. Mathematically, we define language as follows:

$$\Phi = \{\mathcal{P}, \mathcal{F}, \mathcal{X}, \mathcal{L}, \mathcal{S}\}. \quad (1)$$

The language supports two types of negation: *strong negation*, represented by the symbol $\sim \in \mathcal{S}$ to represent contradictory knowledge, and *weak negation* which represents negation as failure represented by the symbol $not \in \mathcal{S}$ which is used to represent incomplete information.

(**Working memory**). Considering a set $\mathcal{P}$ of predicates and an infinite set of variable $\mathcal{X}$, a fact is ground predicate $f \in \mathcal{P}$, or negated ground predicate $\sim f \in \mathcal{P}$. A set of facts represented by $\mathcal{WM}$ is called a 'working memory'. Mathematically, we define working memory as follows:

$$\mathcal{WM} = \{f \cup \sim f | f, \sim f \text{ are ground predicates}\}, \quad (2)$$

where ground predicate is a predicate whose input arguments are constant. The predicate $p(a,b)$ and $not\ p(a,b)$ are ground predicates. Facts represent the current state of the world and these provide some sort of evidence as a basis for activating rules of inference to infer new facts. If there are no facts in the system, then no inference rules will be activated.

(**Production rule**). A production rule $\mathcal{A}$ is of the form: [rule identifier] [rule body] [type of inference rule] [conclusion]. Mathematically, we define a production rule as follows:

$$[\mathcal{A}] \nabla \vdash \alpha, \quad (3)$$

where

- [rule identifier]: $\mathcal{A} \in \mathcal{L}$ is used as the identifier or name of the production rule;
- [rule body]: ∇ is a pattern in the body of a production rule $\mathcal{A}$. A pattern is a tuple of predicates i.e. $\nabla \subseteq \mathcal{P}$, and defined as $\nabla = (C_i, \ldots, C_j)$ where $0 < i < j$, C_i is a predicate in pattern;
- [conclusion]: α is a predicate whose instances could be intuitively considered to be added to the working memory when the rule is fired during argument construction defined later on; and
- [type of inference]: $\vdash$ indicates the inference that associates the rule body with the conclusion.

The production rule represents a reasoning step for α from tuple of predicates $\{C_1, \ldots, C_n\}$. The language supports two types of inferences in production rules. One is strict inference represented by the symbol $\rightarrow \in \mathcal{S}$ and the second is defeasible inference represented by the symbol $\dashrightarrow \in \mathcal{S}$. Strict inference is used to represent information about which there is no ambiguity, whereas defeasible inference is used to represent ambiguous or tentative information. The strong negation is allowed at the conclusion of the rule, whereas weak negation is allowed only in the body of the rule.

(**Knowledge base**). The set of production rules is known as the knowledge base denoted by $\mathcal{R}$. Mathematically, we define the knowledge base as follows:

$$\mathcal{R} = \{production\ rule\} \quad (4)$$

(**Strict production rule**). A production rule $\mathcal{S} \in \mathcal{R}$ is a strict production rule of the following form if the rule is based on strict inference

$$[\mathcal{S}] \nabla \rightarrow \alpha. \quad (5)$$

The strict production rule $\mathcal{S} \in \boldsymbol{\mathcal{R}}$ is used to represent truthful information which contains no ambiguity. Let us consider a rule r1 which states that "if a person is innocent and has no crime history then he is not guilty" and rule r2 which states that "if someone is not guilty, then he is free". These rules can be represented as strict production rules thus:

[r1] innocent (X), hasCrimeHistory (X, no) $\rightarrow \sim$ guilty (X) Illustration (1)

[r2] *not* guilty (X) $\rightarrow$ free (X) Illustration (2)

(**Defeasible production rule**). A production rule $\mathcal{D} \in \mathcal{R}$ is a defeasible production rule of the form if the rule is based on defeasible inference.

$$[\mathcal{D}]\nabla \dashrightarrow \alpha \tag{6}$$

The defeasible production rule $\mathcal{D} \in \mathcal{R}$ is used to represent tentative information which may change in due course. Let us consider a rule r3 that states: "assume that someone is innocent whenever it has not been proven that he is guilty" and rule r4 that expresses: "generally, do not cross the railway tracks if it cannot be proven that no train is coming". These rules can be represented as defeasible production rules as follows:

[r3] *not* guilty $(X) \dashrightarrow innocent(X)$ Illustration (3)
[r4] $not \sim$ train_is_coming $\dashrightarrow \sim$ cross_railway_tracks (X) Illustration (4)

(**Argumentative production system**). Formally, we define an argumentative production system as follows:

$$(\mathbb{P} = (\mathcal{WM}, \mathcal{R}, Args). \tag{7}$$

- Where $\mathbb{P} \in \mathcal{L}$ is a label to identify the argumentative production system.
- $\mathcal{WM}$ represents the initial collection of facts in the argumentative production system.
- $\mathcal{R}$ is the set of rules comprised of both strict and defeasible production rules in the argumentative production system.
- *Args* is an active argument set which contains arguments generated during the argument construction phase (defined later on). Prior to the argument construction phase, the *Args* is an empty set.

Consider the following argumentative production system of the case study discussed in Section 3 by representing the formalism in Illustrations 5, 6 and 7.

$$\mathbb{ESHOP} = (\mathcal{WM}, \mathcal{R}, Args) \tag{8}$$

where

$$\mathcal{WM} = \left\{ \begin{array}{l} shopper(david), eShop(BigW), product(washingMachine) \\ havefeedback(washingMachine, feedback), \\ reviwedRate(feedback, good), bulkOrder(david, washingMachine) \end{array} \right\} Illustration(5)$$

$$\mathcal{R} = \left\{ \begin{array}{l} [s2] gstFree(Y), giveDiscount(X) \\ \quad \rightarrow ordinaryDiscount(X) \\ [s1] not\ gstFree(Y), giveDiscount(X) \\ \quad \rightarrow normalDiscount(X) \\ [d1] shopper(X), purchase(X, Y) \\ \quad \dashrightarrow giveDiscount(X) \\ [d2] shopper(X), not\ advancePayment(X, Y) \\ \quad \dashrightarrow \sim giveDiscount(X) \\ [d3] shopper(X), purchase(X, Y), bulkOrder(X, Y) \\ \quad \dashrightarrow giveDiscount(X). \\ [d4] eShop(Z), packaging(Y, Z) \\ \quad \dashrightarrow gstFree(Y). \\ [d5] eShop(Z), not\ packaging(Y, Z) \\ \quad \dashrightarrow \sim gstFree(Y) \\ [d7] shopper(X), normalDiscount(X) \\ \quad \dashrightarrow platinumDiscount(X) \\ [d8] shopperX), normalDiscount(X), slowToPay(X) \\ \quad \dashrightarrow \sim platinumDiscount(X) \\ [d9] shopper(X), product(Y), havefeedback(Y, Z), \\ \quad reviwedRate(Z, good) \dashrightarrow purchase(X, Y) \end{array} \right\} Illustration(6)$$

$Args = \{\}$ *Illustration*(7)

The defeasible production rule d9 is defined by Mr. David, whereas the remaining rules are defined in supplier business polices. However, the data or fact e.g. 'bulkOrder' is defined by Mr. David, whereas the 'havefeedback', 'reviewedRate' are from a third party reviewing site.

(**Consistency**). A set of rules is consistent if and only if, there are no two rules with mutually contradictory predicates as their conclusion. Mathematically, we represent this as follows:

$$\mathcal{R}_{consis} = \{\forall r, s \in \mathcal{R} | if\ r \vdash \alpha\ then\ s \not\vdash \sim \alpha\}. \tag{9}$$

Given a program $\mathbb{P}$ in Eq. (7), the set of strict rules must be consistent, whereas the set defeasible rules *and* $\mathbb{P}$ itself can be inconsistent. In the $\mathbb{ESHOP}$ argumentative production system, the set {s1,s2} is consistent, whereas set {d1,d2} is inconsistent.

(**Argument construction**). We define argument construction as a recursive process which involves interpretation of production rules with function match $(\mathcal{WM}, \mathcal{R}) \in \mathcal{F}$ which looks for rules from a knowledge base whose pattern matches the facts present in $\mathcal{WM}$ and, on a successful match, executes the production rule which then adds the rules conclusion, i.e. ground predicate, to the working memory. The argument construction process continues until all the matched rules in the knowledge base have been processed. This interpretation of a production rule is also known as 'firing of a rule'.

$$\begin{aligned} &\forall r \in \mathcal{R}\{\nabla \in r, \alpha \in r, r \notin Args | ifmatch(\nabla, \mathcal{WM})\ then\ \mathcal{WM}' \\ &= \mathcal{WM} \cup \alpha'\ and\ Args = Args \cup r'\}, \end{aligned} \tag{10}$$

where α' is the ground predicate and r' is interpreted rule by function $match(\mathcal{WM}, \mathcal{R}) \in \mathcal{F}$. The *Args* contains interpreted rules or fired rules known as arguments defined later in the section. After the argument construction process, we arrive at the following argumentative production system:

$$\mathbb{ESHOP} = (\mathcal{WM}', \mathcal{R}, Args), \tag{11}$$

where $\mathcal{WM}'$ represents the new state of the working memory after the addition of new inferred facts.

Let us consider the case study discussed in Section 3. After the construction of arguments, the production system with updated working memory and populated with the argumentation would be as follows:

$$\mathcal{WM}' = \left\{ \begin{array}{l} shopper(david), eShop(BigW), product(washingMachine) \\ havefeedback(washingMachine, feedback), \\ reviwedRate(feedback, good), bulkOrder(david, washingMachine) \\ purchase(david, washingMachine), \sim gstFree(washingMachine), \\ giveDiscount(david), \sim giveDiscount(david), \\ normalDiscount(david), platinumDiscount(david), \end{array} \right\} \quad Illustration(8)$$

$$\mathcal{R} = \{same - as - above - in - ilustration - 6\} \quad Illustration(9)$$

$$Args = \left\{ \begin{array}{l} [d1]shopper(david), purchase(david, washingMachine) \\ \quad \dashrightarrow giveDiscount(david) \\ [d2]shopper(david), not\ advancePayment(david, washingMachine) \\ \quad \dashrightarrow \sim giveDiscount(david) \\ [d3]shopper(david), purchase(david, washingmachine), \\ bulkOrder(david, washingMachine) \\ \quad \dashrightarrow giveDiscount(david). \\ [d5]eShop(BigW), not\ packaging(BigW, washingMachine) \\ \dashrightarrow \sim gstFree(washingMachine) \\ [s1]not\ gstFree(washingMachine), giveDiscount(david) \\ \quad \rightarrow normalDiscount(david) \\ [d7]shopper(david), normalDiscount(david) \\ \quad \dashrightarrow platinumDiscount(david) \\ [d9]shopper(david), product(washingMachine), \\ havefeedback(washingMachine, feedback), \\ reviewRate(feedback, good) \dashrightarrow purchase(david, washingMachine) \end{array} \right\} \quad Illustration(10)$$

The argument construction process may produce new facts in the working memory which may contradict the already existing facts in the memory. The objective is to retain the contradictory information and use it in the reasoning process rather than eliminate it.

(**Strict argument**). A fired production rule in an active argument set with strict inference is called a 'strict argument'. Mathematically, we represent this as follows:

$$[\mathcal{S}]\beta_1, \ldots, \beta_n \rightarrow \alpha, \quad (12)$$

where

1. $\mathcal{S} \in \mathcal{L}$ is the label of the argument.
2. α is a ground predicate known as 'claim of an argument'. Function $claim(\mathcal{S}) \in \mathcal{F}$ returns claim of a given argument $\mathcal{S}$.
3. β_i is a ground predicate known as the premise of an argument, supporting the claim of an argument. Function $premises(\mathcal{S})$ returns set of premise of argument $\mathcal{S}$.
4. $\rightarrow$ represents strict inference from the set of premises to the claim.

In illustration 10, argument 's1' is strict argument.

(**Defeasible argument**). A fired production rule in an active argument set with defeasible inference is called a 'defeasible argument'. Mathematically, we represent this as follows:

$$[\mathcal{D}]\beta_1, \ldots, \beta_n \dashrightarrow \alpha, \quad (13)$$

where

1. $\mathcal{D} \in \mathcal{L}$ is the label of an argument.
2. α is a ground predicate known as 'claim of an argument'. Function $claim(\mathcal{D}) \in \mathcal{F}$ returns claim of a given argument $\mathcal{D}$.
3. β_i is a ground predicate known as the premise of an argument, supporting the claim of an argument. Function $premises(\mathcal{D})$ returns set of premise of argument $\mathcal{D}$.
4. $\dashrightarrow$ represents defeasible inference from the set of premises to the claim.

In illustration 10, arguments d1, d2, d3, d5, d7 and d9 are defeasible arguments.

To avoid any fallacies in the argumentation process, we consider the following restrictions on strict and defeasible argument structure:

1. A premise in an argument cannot simultaneously be a conclusion i.e. $\beta_i \notin \alpha$.
2. A negation of a claim cannot become the premise of a claim i.e. $\beta_i \neq \sim\alpha$.
3. There is no redundancy of premise in a pattern. $\beta_i \neq \beta_j$ where $1 < i, j < n$.

(**Counter-argument**). An argument r counter-argues argument s if and only if $claim(r)$ is inconsistent with $claim(s)$ or $claim(r)$ is inconsistent with the $premises(s)$. Mathematically, we define counter-argument as:

$$(\forall r, s\{\text{if}(!\text{Consistent}(\text{claim}(s), \text{claim}(r))\text{then } r \diamondsuit s\}, \quad (14)$$

where $\diamondsuit$ is used to represent the counter-argument relationship between two arguments.

If argument r counter-argues argument s such that $claim(r)$ is inconsistent with $claim(s)$, it is called a 'direct counter-argument', and if argument r counter-argues s such that $claim(r)$ is inconsistent with $premises(s)$, then it is called an 'indirect

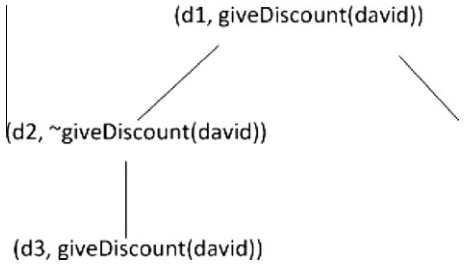


Fig. 4. Dialectical tree for counterarguments.

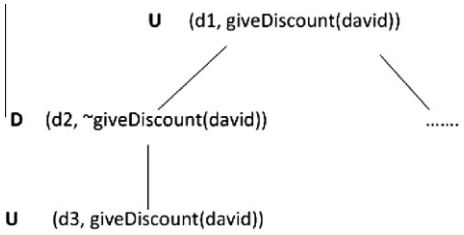


Fig. 5. Marked dialectical tree.

counter-argument'. Mathematically, direct and indirect counter-arguments are represented as follows:

$$\forall s, r\{\text{if!Consistent}(\text{claim}(s), \text{claim}(r))\text{then } s\diamondsuit_{direct} r\}, \quad (15)$$

$$\forall s, r\{\text{if!Consistent}(\text{claim}(s), \text{premises}(r))\text{then } s\diamondsuit_{indirect} r\}. \quad (16)$$

A strict rule cannot counter-argue another strict rule because of the definition of consistency.

(**Static defeat**). Under certain conditions, an argument r defeats its counter-argument s by establishing its preference over its counter-argument. Such defeat is known as a 'static defeat'.

The conditions for static defeat are as follows:

- If a strict argument counter-argues a defeasible argument, strict argument always defeats a defeasible argument. In other words, strict argument has higher priority than defeasible argument. Mathematically, we represent this as follows:

$$\forall d, s \in Args\{\text{if } s, d \text{ are strict and defeasible argument respectively } |s\diamondsuit_{direct} d \text{ then } s > d\}. \quad (17)$$

- If a defeasible argument directly counter-argues a strict argument, then the strict argument defeats the defeasible argument. Mathematically, we represent this as follows:

$$\forall s, d \in Args\{\text{if } s, d\text{are strict and defeasible argument respectively } |d\diamondsuit_{direct} s \text{ then } s > d\} \quad (18)$$

(**Dialectical tree**. [20]) If an argument $\mathcal{A}$ counter-argues argument $\mathcal{B}$, and no static defeat exists, then we construct a dialectical tree for argument $\mathcal{A}$ to determine whether argument $\mathcal{A}$ defeats argument $\mathcal{B}$ or vice versa.

Let $\mathcal{A}$ be an argument. A dialectical tree for argument $\mathcal{A}$, is $\Sigma(\mathcal{A}, \text{h})$ where h is claim $(\mathcal{A})$, is recursively defined as follows:

(1) A single node labeled with an argument $(\mathcal{A}, \text{h})$ with no counter-argument is by itself a dialectical tree for $(\mathcal{A}, \text{h})$. This node is also the root of the tree.

(2) Suppose that $\Sigma(\mathcal{A}, \text{h})$ is an argument with counter-arguments $(\mathcal{A}_1, \text{h}_1), (\mathcal{A}_2, \text{h}_2), \ldots, (\mathcal{A}_n, \text{h}_n)$, we construct the dialectical tree for $(\mathcal{A}, \text{h})$, $\Sigma(\mathcal{A}, \text{h})$ by labeling the root note with $(\mathcal{A}, \text{h})$ and by making this node the parent of the root of dialectical trees for $(\mathcal{A}_1, \text{h}_1), (\mathcal{A}_2, \text{h}_2), \ldots, (\mathcal{A}_n, \text{h}_n)$ i.e. $\Sigma(\mathcal{A}_1, \text{h}_1), \Sigma(\mathcal{A}_2, \text{h}_2), \ldots, \Sigma(\mathcal{A}_n, \text{h}_n)$.

Taking into consideration illustration 10, where argument d1 counter-argues argument d2, in order to identify the result of the counter-argument, we construct a dialectical tree for argument d1 as shown in Fig. 4.

(**Marking of dialectical tree** [20]).

(1) Leaves of $\Sigma(\mathcal{A}, \text{h})$ are U-nodes.
(2) Let $(\mathcal{B}, \text{q})$ be an inner node of $\Sigma(\mathcal{A}, \text{h})$. Then (B, q) will be a U-node iff every child of $(\mathcal{B}, \text{q})$ is a D-node. The node $(\mathcal{B}, \text{q})$ will be a D-node if it has at least one U-node as a child.

Considering the example from Section 3, the marked dialectical tree is shown in Fig. 5 below. The status is represented as $\Sigma_U(d1,\text{giveDiscount (david)})$.

(**Dynamic defeat**). Let $\Sigma_U(\mathcal{A}, \text{h})$ be marked dialectical tree for argument $\mathcal{A}$ and $\Sigma_D(\mathcal{B}, \tilde{}\text{h})$ is marked dialectical tree for its counter-argument $\mathcal{B}$, then argument $\mathcal{A}$ establishes its preference over its counter-argument $\mathcal{B}$ known as dynamic defeat. The dynamic defeat results in establishment of preference of an argument over its counter-argument and such preference is known as dynamic a priority. Mathematically, we define dynamic priority as follows:

$$\forall r, s \in Args\{\text{if } r \diamondsuit s, \Sigma_U(r, h), \Sigma_D(s, \sim h) \text{ then } r > s. \quad (19)$$

If an argument $\mathcal{A}$ has undefeated dialectical tree, i.e. $\Sigma_U(\mathcal{A}, \text{h})$ and it counter-argue an argument $\mathcal{B}$ which also have undefeated dialectical tree, i.e. $\Sigma_U\mathcal{B}, \sim \text{h})$, then neither argument A nor B can establish its preference over the other, resulting in a blocking situation. Such arguments are considered as blocking arguments.

(**Sub-argument**). Given an active argument set *Args*, an argument s is a sub-argument of r if and only if claim $(s) \subseteq premise(r)$ and, if there exists say counter-argument g, then marked dialectical tree of an argument s is undefeated and marked dialectical tree of is of argument g is defeated. Mathematically, the condition for a sub-argument can be represented as follows:

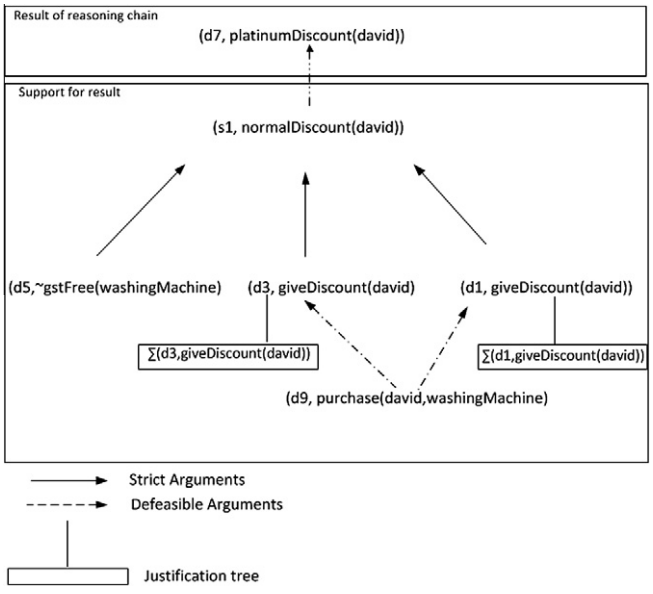


Fig. 6. Mixed reasoning chain.

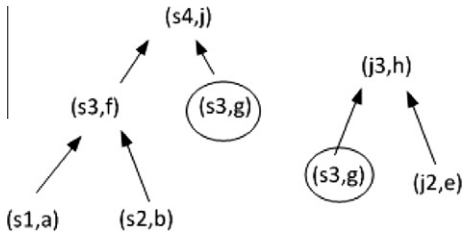


Fig. 7. Dependent reasoning chains $\lambda_{(J,j)}$ and $\lambda_{(H,h)}$.

$$\forall r,s,g\{\text{if}(\text{claim}(s)\subseteq premise(r)\ and\ if(s\diamondsuit\ g)\ then\ s>g)\text{then}\ s\xi r\}, \tag{20}$$

where ξ used to represent the sub-argument relationship between two arguments.

The sub-argument is a supporting argument and it must have the following characteristics:

1. Argument s is consistent w.r.t argument r.
2. There is no $premise(s)$ such that $premise(s)\subseteq claim(r)$.

A sub-argument that provides support to another argument results in a chaining of arguments.

(**Reasoning chain**). An argument $\mathcal{A}$ supported by a chain of sub-arguments produces a reasoning chain $\lambda_{\mathcal{A}}=\mathcal{A}_1,\ldots\ldots,\mathcal{A}_n)$ for an argument $\mathcal{A}$. The claim of supported argument $\mathcal{A}$, is called a 'result' of the reasoning chain and the chain of sub-arguments is called a 'support' for the result of the reasoning chain. Mathematically, we define a reasoning chain as follows:

$$\forall r,s\in Args\{if(s\xi r)\ then\ \lambda_{(r,j)}=\lambda_{(r,j)}\cup s, \tag{21}$$

where ξ is used to represent sub-argument relationship and $\lambda_{(r,j)}$ is used to represent a reasoning chain with result j.

The reasoning chain should have the following characteristics:

1. The reasoning chain is consistent (i.e., there is no contradiction in result and support for the result). Therefore, for example, $giveDiscount(david)$ and $\sim giveDiscount(david)$ will not belong to one reasoning chain, but each one of them can belong to different reasoning chains and those reasoning chains represent alternative paths or choices.
2. There is no defeated argument in a reasoning chain.
3. Two blocking arguments cannot be in the same reasoning chain.

An argument can belong to two different reasoning chains. An argumentation framework may be comprised of a number of reasoning chains.

(**Strict reasoning chain**). A reasoning chain is considered to be strict if all the arguments in the reasoning chain are strict arguments. Mathematically, a strict reasoning chain can be represented as follows:

$$\forall r,\ s\in\lambda_{(r,j)}\{r,s\ \text{are strict arguments}\} \tag{22}$$

This reasoning chain cannot be directly counter-argued by other reasoning chains. However, this reasoning chain can counter-argue and defeat the rest of the reasoning chains in an argumentative production system.

(**Defeasible reasoning chain**). A reasoning chain is known as a defeasible reasoning chain if all arguments in the reasoning chain are defeasible arguments. Mathematically, defeasible reasoning chains can be represented as follows:

$$\forall d,f\in\lambda_{(r,j)}\{d,f\ \text{are defeasible arguments}\}. \tag{23}$$

This reasoning chain can counter-argue or can be counter-argued by other reasoning chains in an argumentative production system. The defeasible arguments must be undefeated and consistent within the defeasible reasoning chain.

(**Mixed reasoning chain**). A reasoning chain is called a mixed reasoning chain if it has a least one defeasible and one strict argument. Mathematically, mixed reasoning chains can be represented as follows:

$$\forall r,s\in\lambda_{(r,j)}\{\exists r\ \text{that is a defeasible argument},\ \exists s\ \text{that is a strict argument}\}. \tag{24}$$

Fig. 6 depicts a mixed reasoning chain representation of illustration 10. The mixed reasoning chain also represents the negotiation process within the argumentation framework.

(**Dependent reasoning chains**). a reasoning chain is dependent upon other reasoning chains if there is at least one common sub-argument. If the common argument is a strict argument, then a reasoning chain is known as a strictly dependent reasoning chain; if a weak argument, then weak dependent, and medium dependent if common arguments are both strict and defeasible. Mathematically, this is represented as follows:

$$\text{if}(\lambda_{(J,j)}\cap\lambda_{(H,h)})\neq\varnothing\ \text{then}\ \lambda_{(J,j)}\ \text{and}\ \lambda_{(H,h)}\ \text{are dependent reasoning chains}. \tag{25}$$

Fig. 7 depicts a dependent reasoning chain with one common argument, i.e. (s3,g). Similarly, all those chains excluding dependent chains are known as independent reasoning chains.

(**Query**). A query 'q', consisting of a predicate, can be executed on the argument set $Args$ with help of the function $executeQuery(q,Args)\in\mathcal{F}$ to check the support for the predicate in the argument set. There are four possible answers to a query, as follows:

- If the answer is 'yes', then the result will be an undefeated dialectical tree. Mathematically, it is presented as follows:

$$\Sigma_U(\mathcal{A},h)=executeQuery(q,Args). \tag{26}$$

- If the answer is 'no', then the result will be a defeated dialectical tree. Mathematically, it is presented as follows:

$$\Sigma_D(\mathcal{A},h)=executeQuery(q,Args). \tag{27}$$

- If the answer is 'undecided', then the result will be a blocked dialectical tree. Mathematically, it is presented as follows:

$$\Sigma_B(\mathcal{A},h)=executeQuery(q\ Args). \tag{28}$$

- Unknown, if the predicate in the query is not in the language of the program. Mathematically, it is presented as follows:

$$\text{unknown}=executeQuery(q\ Args). \tag{29}$$

Algorithm 1: Argumentative Reasoning

```
Data: DeLP rules and DelP facts representing policies and evidences
      respectively.
Result: Category of discount applicable
initialization;
Construct Rete network Alpha and beta nodes etc ;
initialize Rete network;
bool constructArgument ←true;
ActiveRules ActiveRuleSet;
ActiveArgumentSet ArgsSet;
repeat
    foreach rule re ∈ KnowledgeBase do
        if match(re, WM)=true then
            ActiveRuleSet ←ActiveRuleSet ∪ re;
        else
            constructArgument ←false;
        end
    end
    foreach re ∈ ActiveRuleSet do
        WM ← WM ∪ claim(r);
        ArgSet ← ArgSet+interpretationOf(re); end
    until constructArgument ←true ;
    foreach arg_i in ArgSet do
        if arg_i ◊ arg_{i+1} then
            Σ_status(arg_i,h_i)← BuildDialecticalTree(arg_i,h_i);
            Σ_status(arg_{i+1},h_{i+1})← BuildDialecticalTree(arg_{i+1},h_{i+1});
            if Σ_D(arg_i,h_i) and Σ_U(arg_{i+1},h_{i+1}) then
                arg_{i+1} > arg_i;
            end
            if Σ_U(arg_i,h_i) and Σ_D(arg_{i+1},h_{i+1}) then
                arg_i > arg_{i+1};
            end
            if Σ_B(arg_i,h_i) and Σ_B(arg_{i+1},h_{i+1}) then
                arg_i <> arg_{i+1};
            end
        end
    end
    foreach arg_i in ArgSet do
        if arg_i not sub-argument of X where X is argument in ArgSet
        then
            BuildReasoningChain(arg_i);
        end
    end
```

6.2. Argumentative reasoning algorithm

In this section, we describe the working of our argumentative reasoning algorithm. Algorithm 1 invokes Algorithms 2 and 3. Algorithm 1 takes in DeLP rules and facts present in the knowledge base. After initialization of Algorithm 1, a Rete network is built up and facts are sieved through the Rete network which will result in a set of rules matched with the working memory. On activation of rules, new facts are added to the working memory. This process continues until no more rules can be activated. Then the dialectical trees are constructed for all arguments that have a counter-argument using Algorithm 2. The outcome of Algorithm 2 will be a dialectical tree with a status of defeated, undefeated or blocked. The last step is the build-up of reasoning chains which is carried out by Algorithm 3. During this process, all the sub-arguments of an argument with undefeated dialectical trees are linked together as a reasoning chain. This process will continue until all the arguments are linked up into reasoning chains. The top argument i.e. conclusion, of the reasoning chain is called as a result of the reasoning chain, and the chain of sub-arguments supporting the top argument are called to support the conclusion.

7. Extensions to defeasible logic programming (DeLP) for Web@IDSS

Defeasible logic programming (DeLP) is a declarative language based on non-monotonic logic and has been used in software agents for providing goal-driven reasoning. It uses argumentation semantics to resolve conflicts among rules. We identified DeLP limitations (Section 4.2) in the context of Web-based IDSS. In this research, we extend DeLP to make it interoperable or able to be used for argumentative reasoning in Web@IDSS.

Algorithm 2: BuildDialecticalTree[20]

```
Data: (A, h)
Result: Σ_status(A, h)
Let C ← get all counter-arguments of (A, h);
if C ≠ ∅ then
    while there is no Σ_U(A_i, h_i) ∈ C do
        for every argument in C do
            Let (A_i, h_i)← minimal non-labelled element
            BuildDialecticalTree((A_i, h_i)) getting result as Σ(A_i, h_i);
            Put Σ(A_i, h_i) ξ (A, h)
        end
        if there exist some Σ_U(A_i, h_i) then
            Set Σ_D(A, h);
        else
            Set Σ_U(A, h);
        end
    end
else
    Σ(A, h)= (A, h);
    Set Σ(A, h) ← defeated;
end
```

Algorithm 3: BuildReasoningChain

```
Data: (A, h)
Result: λ_(A,h)
Let S ← get all sub-arguments of (A, h);
if S ≠ ∅ then
    foreach (A_i, h_i) ∈ S do
        if noCounterArgument(A_i, h_i) or Σ_U(A_i, h_i) then
            BuildReasoningChain((A_i, h_i)) ;
            Put λ(A_i, h_i) ξ (A, h);
        end
    end
else
    λ_(A,h) = (A, h);
end
```

We made the following extensions to current DeLP:

1. We define semantics for data driven reasoning for DeLP described in Section 6.
2. We extend goal driven reasoning semantics of DeLP described in Section 6.
3. We make DeLP compatible with RuleML by providing translation of RuleML to DeLP rules.
4. We make DeLP compatible with RDF/XML by providing translation of RDF/XML to DeLP facts.
5. We also provide representation of reasoning output in Argument Interchange Format (AIF) to make it shareable among argumentation-based tools.

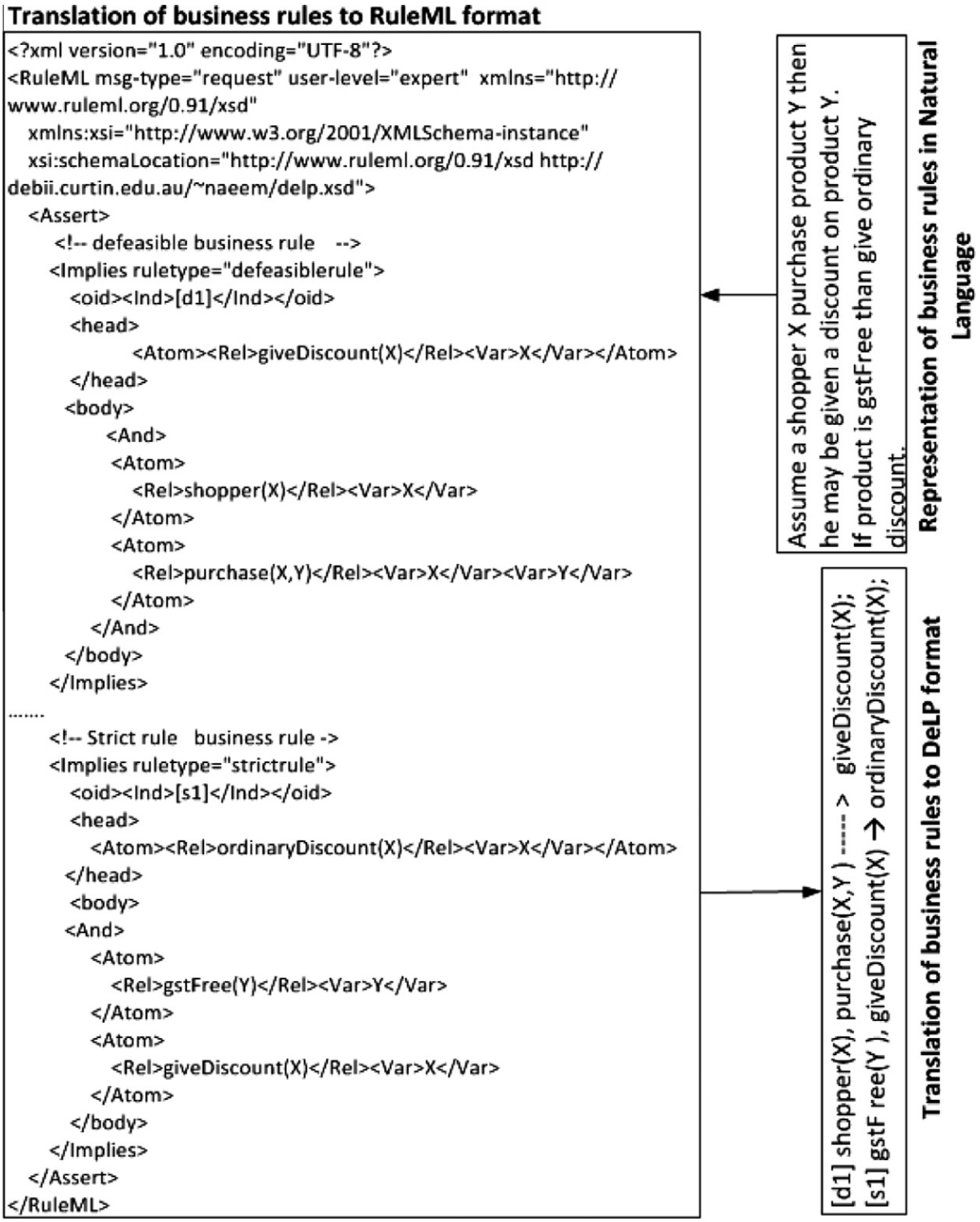


Fig. 8. Transformation of business rules into different formats.

In following sections, we discuss extensions 2, 3 and 4 to DeLP in detail.

7.1. Translation of business rules defined in RuleML to DeLP rules

RuleML already supports different rule types via the *Implies* element and allows them to be named using the oid element. RuleML syntax has been extended to express defeasible rules, defeaters, and superiority relations [18,66]. A *@ruletype* attribute has been added to the Implies element, allowing it to take one of three values: strict rule, defeasible rule or defeater. Because strict rule is implied when *@ruletype* is absent, when non-defeasible RuleML rule sets are imported, the rules are correctly considered as strict. Bassiliades et al. [18] used *@superior* attribute on the superior rule as a link to the *@ruleID* label of the inferior rule. However, Pham et al. [66] found this approach unsuitable and used a predicate,

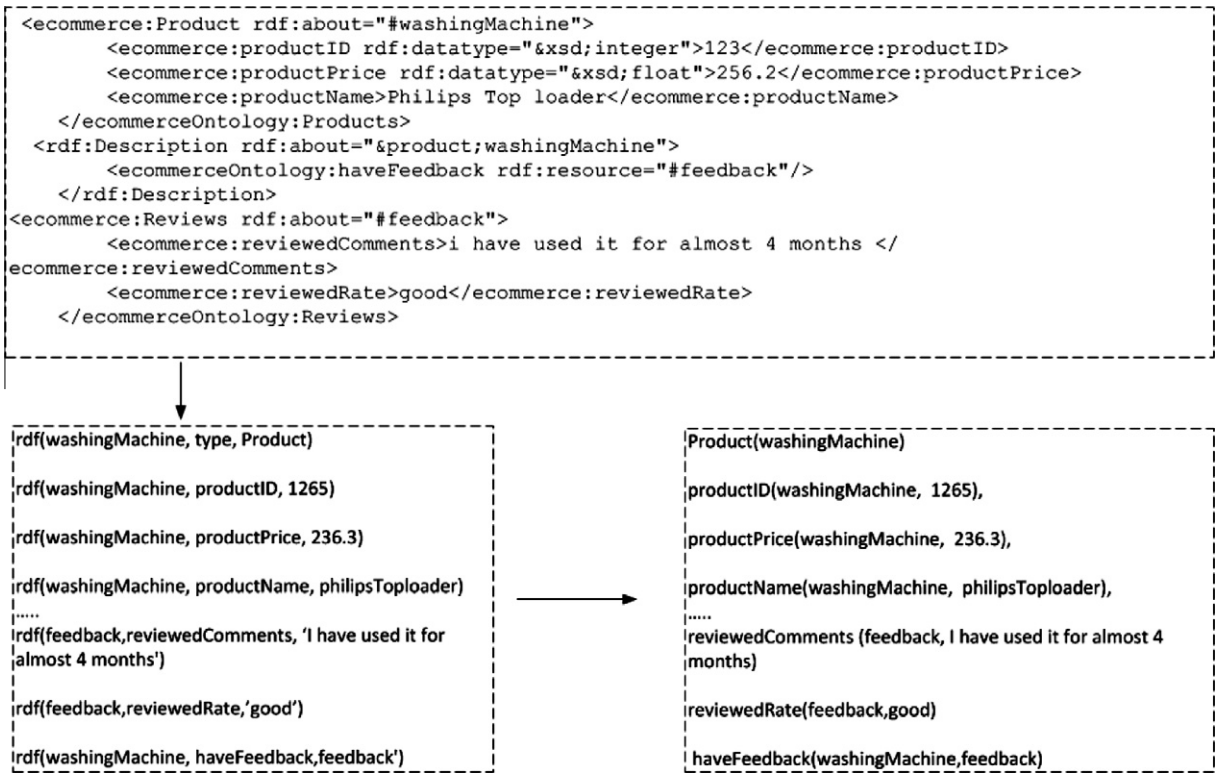


Fig. 9. Translation of data in RDF/XML format to DeLP facts.

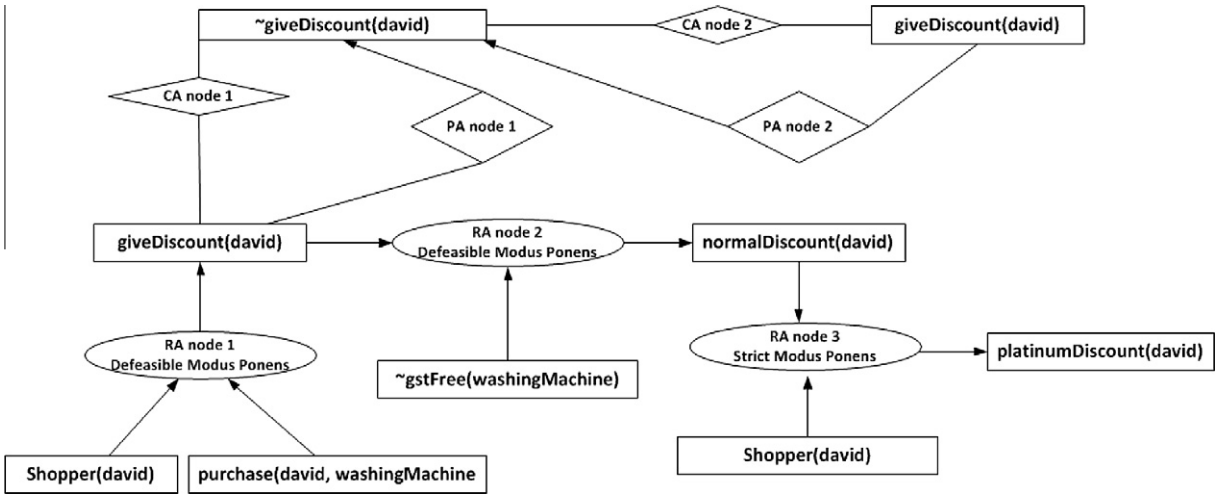


Fig. 10. Argument network representation in AIF.

i.e. *Override*, to explicitly represent the superiority relation among the rules.

In Web@IDSS, the defeasible rules defined in RuleML notation are translated to DeLP rules with the help of a Rule Translator module. Firstly, the RuleML parser the defeasible rules into an intermediate format and forwards the parsed rules to the DeLP translator to translate the defeasible rules to DeLP rules. The translator also saves certain information about the rules such as file URL, number of rules translated, owner/creator of rules, argument from expert, etc., in a database for their profiling. After translation, the rules are saved in a knowledge base. Fig. 8 represents the entire cycle of business rule representation in natural language and then its translation to RuleML format and ultimately into DeLP format.

7.2. Parsing of Web data defined in RDF/XML into DeLP facts

For execution of business rules defined in DeLP, we translate RDF/XML data into DeLP facts.

1. RDF/XML data is transformed with the help of the SWI-Prolog RDF Parser [67] into intermediate triple format i.e. *rdf (Subject, Predicate, Object).*
2. The intermediate triple format is further processed to transform the rdf statements into *Predicate (Subject, Object)*format.
3. The facts in *Predicate (Subject, Object)*format are then saved in the knowledge base. The type attribute is further translated using the following formula: type $(X, C) \rightarrow C\ (X)$.

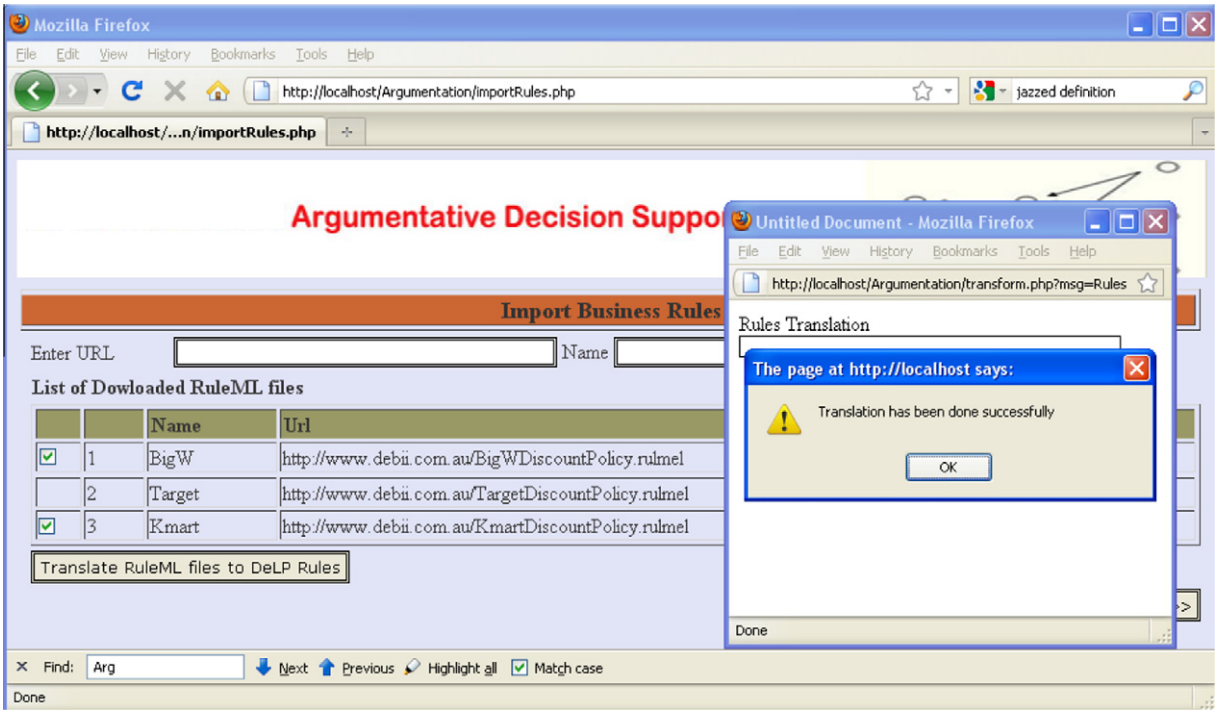


Fig. 11. Translation of business rules from RuleML to DeLP.

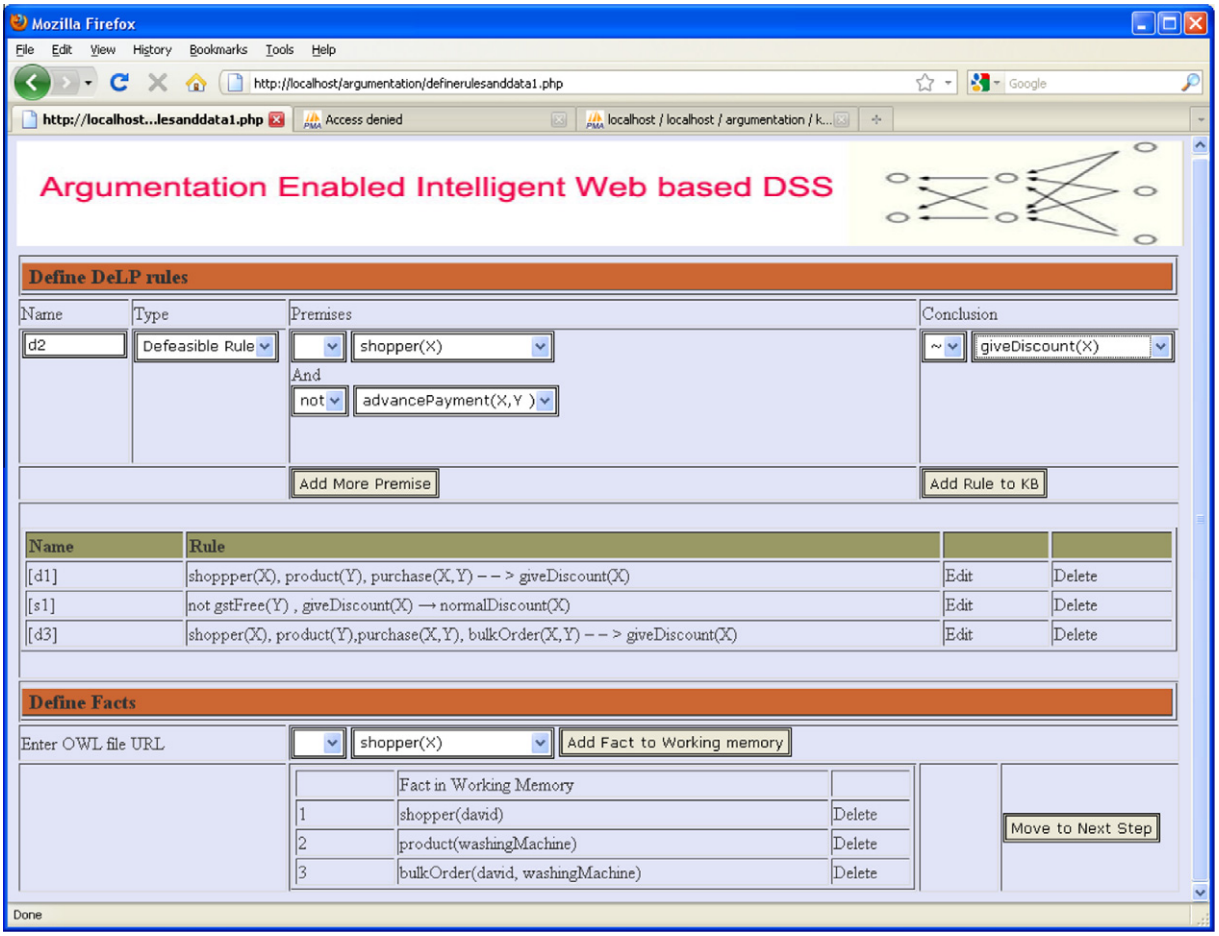


Fig. 12. Interface to define business rules and facts.

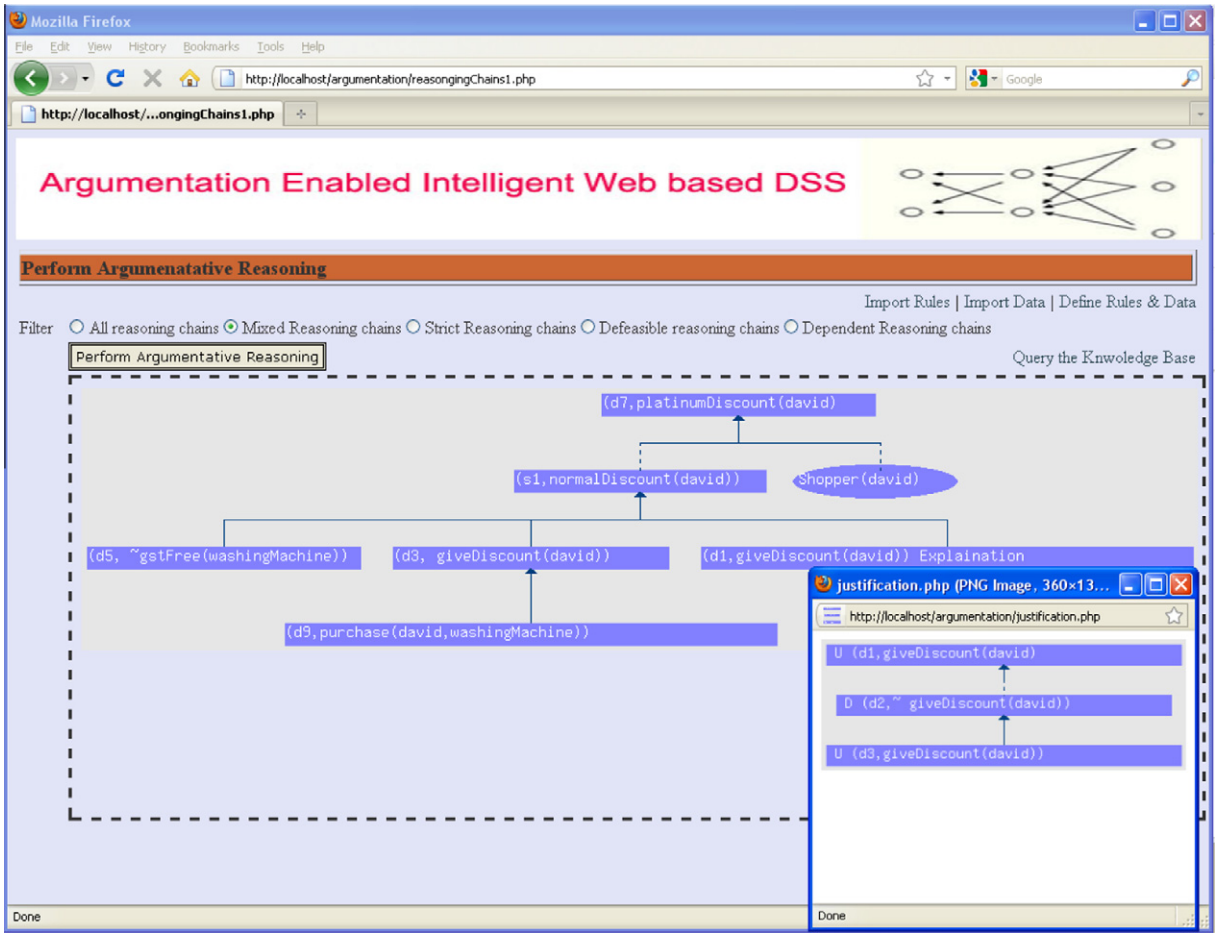


Fig. 13. Explanation of dynamic priority.

Fig. 9 represents the translation of data from RDF/XML format to DeLP facts.

7.3. Representation of a reasoning chain in AIF format

The Argument Interchange Format (AIF) is the result an international effort to develop a representational mechanism for exchanging argument resources between systems, research groups, tools, and domains using a semantically rich language. The output of the argumentative reasoner will be a justified conclusion based upon existing facts. This information is very useful and exporting this in AIF format will enable the system to merge it with the justified conclusions of other machines in the absence of complete or accurate information.

The ReasoningChain $\lambda_{(j,j)}$ in terms of AIF is defined as a graph with a set of nodes ($\mathcal{N}$ X $\mathcal{N}$) where X represents the binary relationships between nodes of the graph. The node $\mathcal{N}$ with only incoming edges, represents the'result' of the graph. The remaining nodes with either only outgoing edges or both incoming and outgoing edges, represents'support'for the result. Fig. 10 represents the mixed reasoning chain results in AIF compliant format.

8. Implementation and prototype development

This section provides the implementation details and working of Web@IDSS to represent and reason over incomplete and conflicting information spanning organizational boundaries. The development of the prototype system is carried out on a machine having Apache Web server version 2.2.11, PHP version 5.3.0, PHP Tree Graph Ext library[1] with certain extensions to differentiate between fact and claim of a rule, strict and defeasible inference, etc., MySql database version 5.1.36 and SWI-Prolog installed on it. After prototype development, the Web application is deployed using the local server.

The system provides an interface allowing user to import the public business rules or polices defined in RuleML format over the Web. The user can download the RuleML files by entering the file URL and name in the provided text fields and clicking the 'download' button as depicted in Fig. 11. Once the user has finished downloading the files, he can then translate these into DeLP format. Fig. 11 depicts the interface where the user can select the files by clicking the check-boxes and submitting the selected files for translation by clicking the 'Translate RuleML files to DeLP' rules button. Similarly, the systems interface allows the user to download RDF/XML files and translate them to DeLP facts, prior to the decision-making process.

Fig. 12 depicts an interface where a user can define his organizational business rules and facts to be stored in the knowledge base. The user can create a rule by assigning it a name, inference type, set of premises and a conclusion. The user can also view the list of rules created and can edit and delete them. The user can define certain facts to be saved in working memory which will be a source of rules activation.

[1] http://download.getabest.com/new/php-tree-graph-ext-222943.html.

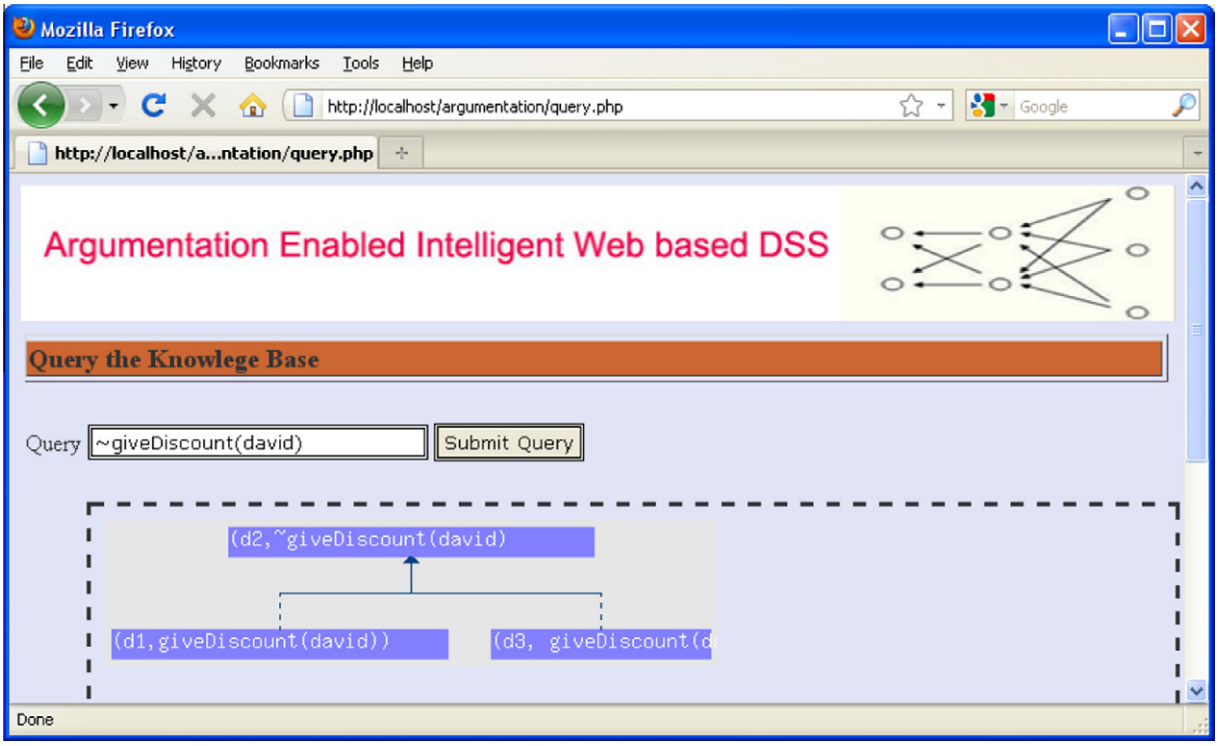


Fig. 14. Querying the knowledge base.

Once the user has finished defining his rules and facts and importing rules from other organizations, he proceeds to the next step, i.e. argumentative reasoning by clicking the 'Move to Next Step' button (shown in Fig. 12 above). Fig. 13 depicts the interface where the user can select a filter such as 'Mixed chain reasoning' and click the 'Perform Argumentative Reasoning' button to trigger the reasoning process.

Once the user clicks the 'Perform Argumentative Reasoning' button, the Web@IDSS takes into account all the rules present in the knowledge base and the data in the working memory to start performing argumentative reasoning and display the results to the user as depicted in Fig. 13. The oval shaped nodes are used to represent the facts and the rectangular nodes are used to represent the claim of the rules. The dotted line arising from the nodes represents the defeasible inference and the solid line represents strict inference.

If the reasoning chain contains some argument with justification, the user can click on the node labeled 'Explanation'. A pop-up window will appear showing a dialectical tree providing justification of dynamic priority as depicted in Fig. 13.

If the user wants to query the knowledge base, he can click on the 'Query Knowledge Base' link provided on the Argumentative Reasoning page and a pop-up window will appear where the user can enter the query. By clicking the 'submit' button, the Web@IDSS generates the dialectical tree against the query and displays it to the user as shown in Fig. 14.

9. Conclusion and future work

Advancements in Internet, WWW and AI technologies have provided tools and technologies which can be used by information system researchers to address the limitations of current DSS applications. The development of Web-based DSS is a big step forward in this regard. However, Web DSS are still subject to inherent limitations in that they still cannot handle incomplete and conflicting information.

In this paper, we determined that a major shortcoming of the existing IDSS applications is their inability to represent and handle incomplete and conflicting information spanning organizational boundaries. This is particularly important for organizations which take into consideration the data available on the Web for timely and accurate decision-making support. We identify that the argumentation is a pivotal methodology used by humans to overcome conflicts among participants belonging to diverse groups with diverse interests. Hence, we have proposed a conceptual framework for 'Argumentation Enabled Web IDSS' and described its working in detail. We have also defined formal argumentation semantics for the system. Finally, we discuss the implementation and prototype development with the help of screen shots of the Web@IDSS application. The major contributions of this paper is that it proposes Web@IDSS with the following research advances over other IDSS systems:

1. Formalization of data-driven and goal-driven reasoning process.
2. Proposal of a mechanism for reasoning over incomplete and conflicting information in the context of IDSS.
3. Web@IDSS can take into account both the internal as well as external data of an organization in the decision-making process.
4. Proof explanation of the reasoning results generated by our proposed Web@IDSS.
5. Representation of reasoning results in AIF format to make the output of Web@IDSS shareable and compatible with other systems.

Regarding future tasks, we will be working on the development of the following:

1. Integration of different reasoning chains, computed by different Web@IDSS, into consolidated argumentation framework with help of the Argumentation schemes [68].

2. Interactive graphical output visualizer so that non-technical users can interact with the reasoning chain to drill up and down the reasoning chains for better understanding of results.
3. Advanced query interface to handle different types of queries in Web@IDSS such as "Why" will lead to construction of justification, "What if" queries define addition facts and their impact and "How" queries will determine what should be done in order to achieve that objective.
4. Benchmarking Web@IDSS with other similar contemporary enterprise DSS.

Acknowledgement

We would like to express our appreciation to Dr. Alejandro J. Garcia and Mr. Sebastian Gottifredi from the "Artificial Intelligence Research and Development Laboratory (LIDIA)", Argentina, for providing DeLP Server v 0.5.0 for our research. We would also like to express our gratitude to the anonymous reviewers of our manuscript for their valuable suggestions to improve this article.

References

[1] S. Negash, P. Gray, Business intelligence, in: Americas Conference on Information Systems, 2003.
[2] D.J. Power, R. Sharda, Decision support systems, in: S.Y. Nof (Ed.), Springer Handbook of Automation, Springer, Berlin, Heidelberg, 2009, pp. 1539–1548. ISBN: 978-3-540-78831-7.
[3] E. Thomsen, BI's promised land, Intelligent Enterprise 6 (5) (2003) 20–25.
[4] J.P. Shim, M. Warkentin, J.F. Courtney, D.J. Power, R. Sharda, C. Carlsson, Past, present, and future of decision support technology, Decision Support Systems 33 (2) (2002) 111–126.
[5] Y. Xie, H. Wang, J. Efstathiou, A research framework for Web-based open decision support systems, Knowledge-Based Systems 18 (7) (2005) 309–319. ISSN 0950-7051.
[6] S. Eom, E. Kim, A survey of decision support system applications (1995–2001), Journal of the Operational Research Society 57 (2005) 1264–1278.
[7] K. Xu, S.S. Liao, J. Li, Y. Song, Mining comparative opinions from customer reviews for competitive intelligence, Decision Support Systems, in press, Corrected Proof, ISSN: 0167-9236.
[8] Y. Yao, N. Zhong, J. Liu, S. Ohsuga, Web intelligence (WI) research challenges and trends in the new information age, in: Web Intelligence: Research and Development, Lecture Notes in Computer Science, vol. 2198, Springer, Berlin, Heidelberg, 2001, pp. 1–17.
[9] J. Yao, Web-based Support Systems, first ed., Springer Publishing Company, Incorporated, 2010.
[10] W. Chung, H. Chen, E. Reid, Business stakeholder analyzer: an experiment of classifying stakeholders on the Web, Journal of the American Society for Information Science and Technology 60 (1) (2009) 59–74.
[11] H. Deng, S. Wibowo, A rule-based decision support system for evaluating and selecting IS projects, in: Proceedings of the International MultiConference of Engineers and Computer Scientists, Hong Kong, 2008.
[12] K. Cheung, M.-P. Cheong, Intelligent on-line decision support tools for market operators, in: International Conference on Intelligent Systems Applications to Power Systems, 2007, pp. 1 –6.
[13] J.P. Shim, M. Warkentin, J.F. Courtney, D.J. Power, R. Sharda, C. Carlsson, Past, present, and future of decision support technology, Decision Support Systems 33 (2) (2002) 111–126. ISSN 0167-9236.
[14] F.V. Assche, P. Layzell, P. Loucopoulos, G. Speltincx, Information systems development: a rule-based approach, Knowledge-Based Systems 1 (4) (1988) 227–234.
[15] W. Wen, Y. Chen, I. Chen, A knowledge-based decision support system for measuring enterprise performance, Knowledge-Based Systems 21 (2) (2008) 148–163.
[16] C. Carlsson, E. Turban, DSS: directions for the next decade, Decision Support Systems 33 (2) (2002) 105–110.
[17] G. Antoniou, A. Bikakis, DR-Prolog: a system for defeasible reasoning with rules and ontologies on the semantic Web, IEEE Transactions on Knowledge and Data Engineering 19 (2) (2007) 233.
[18] N. Bassiliades, G. Antoniou, I. Vlahavas, DR-DEVICE: A defeasible logic system for the semantic Web, Principles and Practice of Semantic Web Reasoning (2004) 134–148.
[19] B.N. Grosof, M.D. Gandhe, T.W. Finin, SweetJess: translating DAMLRuleML to JESS, in: Proceedings of the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web, 2002.
[20] A.J. Garcia, G.R. Simari, Defeasible logic programming: an argumentative approach, Theory and Practice of Logic Programming 4 (1+2) (2004) 95–138.
[21] J. Dix, S. Parsons, H. Prakken, G. Simari, Research challenges for argumentation, Computer Science – Research and Development 23 (1) (2009) 8.
[22] K.C. Lee, N. Chung, A web DSS approach to building an intelligent internet shopping mall by integrating virtual reality and avatar, Expert Systems with Applications 28 (2) (2005) 333–346. ISSN 0957-4174.
[23] E. Turban, E. McLean, J. Wetherbe, Information Technology for Strategic Advantage, John Wiley & Sons, Inc., 2001.
[24] M. Brodie, The end of the computing era: Hephaestus meets the olympians, in: Second IEEE International Conference on Digital Ecosystems and Technologies, 2008.
[25] M. Brodie, Understanding our digital universe: Unleashing natural forces, in: Second IEEE International Conference on Digital Ecosystems and Technologies, 2008. DEST 2008, 2009.
[26] D. Power, DSS Case Summaries, URL <http://dssresources.com/cases/DSScatalog.html>, Last visited (1/20/2011).
[27] D.J. Power, Decision Support Systems: Concepts and Resources for Managers, Greenwood Publishing Group, 2002.
[28] R. Vahidov, G.E. Kersten, Decision station: situating decision support systems, Decision Support Systems 38 (2) (2004) 283–303.
[29] B.G. Silverman, M. Bachann, K. Al-Akharas, Implications of buyer decision theory for design of e-commerce websites, International Journal of Human–Computer Studies 55 (5) (2001) 815–844.
[30] F. Toni, E-Business in ArguGRID, in: D. Veit, J. Altmann (Eds.), Grid Economics and Business Models, Lecture Notes in Computer Science, vol. 4685, Springer, Berlin, Heidelberg, 2007, pp. 164–169.
[31] H.K. Bhargava, D.J. Power, D. Sun, Progress in Web-based decision support technologies, Decision Support Systems 43(4) (2007) 1083–1095, special Issue Clusters.
[32] P. Subsorn, J. Xiao, K. Singh, A web-based application of group decision making in a fuzzy environment, in: 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, vol. 1, 2008, pp. 17–20.
[33] J. Ma, J. Lu, G. Zhang, Decider: A fuzzy multi-criteria group decision support system, Knowledge-Based Systems 23 (1)(2010) 23 – 31, special Issue on Intelligent Decision Support and Warning Systems.
[34] M. Noor-E-Alam, T.F. Lipi, M.A.A. Hasin, A. Ullah, Algorithms for fuzzy multi-expert multi-criteria decision making (ME-MCDM), Knowledge-Based Systems 24 (3) (2011) 367–377.
[35] Z. Yue, An extended TOPSIS for determining weights of decision makers with interval numbers, Knowledge-Based Systems 24 (1) (2011) 146–153.
[36] F. Cabrerizo, I. PTrez, E. Herrera-Viedma, Managing the consensus in group decision making in an unbalanced fuzzy linguistic context with incomplete information, Knowledge-Based Systems 23 (2) (2010) 169–181.
[37] T. Li, S. Feng, L.X. Li, Information visualization for intelligent decision support systems, Knowledge-Based Systems 14 (5-6) (2001) 259–262.
[38] D. Nute, Defeasible reasoning and decision support systems, Decision Support Systems 4 (1) (1988) 97–110.
[39] R.L. Causey, EVID: a system for interactive defeasible reasoning, Decision Support Systems 11 (2) (1994) 103–131.
[40] B. Johnston, G. Governatori, Induction of defeasible logic theories in the legal domain, in: Proceedings of the 9th International Conference on Artificial Intelligence and Law, ICAIL '03, ACM, New York, NY, USA, 2003, pp. 204–213.
[41] G. Antoniou, T. Skylogiannis, A. Bikakis, M. Doerr, N. Bassiliades, DR-Brokering: a semantic brokering system, Knowledge-Based Systems 20 (1) (2007) 61–72.
[42] E. Kontopoulos, N. Bassiliades, G. Antoniou, Visualizing Semantic Web proofs of defeasible logic in the DR-DEVICE system, Knowledge-Based Systems 24 (3) (2011) 406–419.
[43] P. Baroni, D. Fogli, G. Guida, Modeling argumentation in practical reasoning: a conceptual analysis of argument life cycle, in: 7th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Paris, 1998, pp. 1790–1797.
[44] D. Walton, Argumentation in Artificial Intelligence, Springer, 2009. chap. Argumentation Theory: A very short introduction, pp. 1–24.
[45] P.M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games, Artificial Intelligence 77 (2) (1995) 321. ISSN 0004-3702.
[46] S.H. Nielsen, S. Parsons, A generalization of Dungs abstract framework for argumentation: Arguing with sets of attacking arguments, Lecture Notes in Computer Science, vol. 4766/2007, Springer, Berlin, Heidelberg, 2007.
[47] S. Modgil, Reasoning about preferences in argumentation frameworks, Artificial Intelligence 173 (9–10) (2009) 901. ISSN 0004-3702.
[48] T.J.M. Bench-Capon, Persuasion in practical argument using value-based argumentation frameworks, Journal of Logic and Computation 13 (3) (2003) 429. ISSN 0955-792X.
[49] J. Bentahar, R. Alam, Z. Maamar, N.C. Narendra, Using argumentation to model and deploy agent-based applications, Knowledge-Based Systems, in press, ISSN 0950-7051.
[50] I. Rahwan, S.D. Ramchurn, N.R. Jennings, P. McBurney, S. Parsons, L. Sonenberg, Argumentation-based negotiation, The Knowledge Engineering Review 18 (04) (2004) 343, (ISSN 0269-8889).
[51] I. Rahwan, Guest editorial: argumentation in multi-agent systems, Autonomous Agents and Multi-agent Systems 11 (2) (2005) 115. ISSN 1387-2532.
[52] G.S. Mahalakshmi, T.V. Geetha, Argument-based learning communities, Knowledge-Based Systems 22 (4) (2009) 316–323. ISSN 0950-7051.
[53] J.L. Pollock, Rational cognition in OSCAR, Lecture Notes in Computer Science, vol. 1757/2000, Springer, Berlin, Heidelberg, 2000.
[54] G. Vreeswijk, IACAS: an implementation of Chisholm's principles of knowledge, in: In The Proceedings of the 2nd Dutch/German Workshop on Nonmonotonic Reasoning, Utrecht., 1995, pp. 225–234.

[55] C.I. Chesnevar, A.G. Maguitman, G.R. Simari, Argument-based critics and recommenders: a qualitative perspective on user support systems, Data and Knowledge Engineering 59 (2) (2006) 293. ISSN 0169-023X.
[56] M. Morge, The hedgehog and the fox: an argumentation-based decision support system, in: Proceedings of the 4th International Conference on Argumentation in Multi-agent Systems, ArgMAS'07, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 114–131. ISBN 3-540-78914-6, 978-3-540-78914-7.
[57] L.R.D. Vesine, A Confidence System for Solving Real- World Problems with Argumentation, International Journal of Artificial Intelligence 4 (2010) 78–89.
[58] M. Williams, A. Hunter, Harnessing ontologies for argument-based decision-making in breast cancer, in: 19th IEEE International Conference on Tools with Artificial Intelligence, 2007, vol. 2, ISSN 1082-3409, pp. 254 –261, doi:10.1109/ICTAI.2007.110, 2007.
[59] S. Alejandro Gomez, C. Ivan Chesnevar, G.R. Simari, Reasoning with inconsistent ontologies through argumentation, Applied Artificial Intelligence 24 (1–2) (2010) 102–148. ISSN 0883-9514.
[60] N.K. Janjua, F.K. Hussain, Development of a logic layer in the semantic Web: Research issues, Semantics, Knowledge and Grid, International Conference (2010) 367–370.
[61] C.L. Forgy, Rete: A fast algorithm for the many pattern/many object pattern match problem, Artificial Intelligence 19 (1) (1982) 17–37. ISSN 0004-3702.
[62] I. Rahwan, F. Zablitha, C. Reed, Laying the foundations for a world wide argument web, Artificial Intelligence 171 (10–15) (2007) 897. ISSN 0004-3702.
[63] H. Cirstea, C. Kirchner, M. Moossen, P.-E. Moreau, Production Systems and Rete Algorithm Formalisation, Tech. Rep., ILOG, INRIA Lorraine, INRIA Rocquencourt. URL <http://hal.inria.fr/docs/00/28/09/38/PDF/rete.formalisation.pdf>, 2004.
[64] G.A.W. Vreeswijk, Abstract argumentation systems, Artificial Intelligence 90 (1–2) (1997) 225–279. ISSN 0004-370.
[65] N.D. Rotstein, M.O. Moguillansky, A.J. Garcia, G.R. Simari, A dynamic argumentation framework, in: P. Baroni, F. Cerutti, M. Giacomin, G.R. Simari (Eds.), Frontiers in Artificial Intelligence and Applications, Computational Models of Argument – Proceedings of Comma 2010, vol. 216, IOS Press, 2010.
[66] D. Pham, G. Governatori, S. Raboczi, A. Newman, S. Thakur, On extending RuleML for modal defeasible logic, rule representation, Interchange and Reasoning on the Web (2008) 89–103.
[67] J. Wielemaker, SWI-Prolog RDF parser, URL <http://www.swi-prolog.org/pldoc/package/rdf2pl.html>, Last accessed May, 2011.
[68] I. Rahwan, C. Reed, F. Zablith, On building argumentation schemes using the argument interchange format, in: Proceedings of the IJCAI Workshop on Computational Models of Natural Argument (CMNA), 2007.

Inf Syst Front
DOI 10.1007/s10796-012-9365-x

Semantic information and knowledge integration through argumentative reasoning to support intelligent decision making

Naeem Khalid Janjua · Farookh Khadeer Hussain · Omar Khadeer Hussain

Abstract The availability of integrated, high quality information is a pre-requisite for a decision support system (DSS) to aid in the decision-making process. The introduction of semantic web ensures the seamless integration of information derived from diverse sources and transforms the DSS to an adoptable and flexible Semantic Web-DSS (Web-DSS). However, due to the monotonic nature of the layered development of semantic web, it lacks the capability to represent, reason and integrate incomplete and conflicting information. This, in turn, renders an enterprise incapable of *knowledge integration*; that is, integration of information about a subject that could potentially be incomplete, inconsistent and distributed among different Web-DSS within or across enterprises. In this article, we address the issues of incomplete and inconsistent semantic information and knowledge integration by using argumentation and argumentation schemes. We discuss the Argumentation-enabled Information Integration Web-DSS (Web@IDSS) along with its syntax and semantics for semantic information integration, and devise a methodology for sharing the results of Web@IDSS in Argument Interchange Format (AIF) format. We also discuss Argumentation-enabled Knowledge Integration Web-DSS (Web@KIDSS) for semantic knowledge integration. We provide formal syntax and semantics for the Web@KIDSS, propose a conceptual framework, and describe it in detail. We present the algorithms for knowledge integration and the prototype application for validation of results.

Keywords Semantic web · Information integration · Argumentation · Argumentation schemes · Web based DSS

N. K. Janjua · O. K. Hussain
School of Information Systems, Curtin Business School, Curtin University, Perth, WA, Australia

N. K. Janjua
e-mail: naeem.janjua@gmail.com

O. K. Hussain
e-mail: o.hussain@cbs.curtin.edu.au

F. K. Hussain (✉)
Decision Support and e-Service Intelligence (DeSI) Lab, Quantum Computation and Intelligent Systems (QCIS), School of Software, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia
e-mail: Farookh.Hussain@uts.edu.au

1 Introduction

Decision support systems (DSS) are a broad category of interactive computer-based information systems designed and developed over the last forty years for a wide range of domains with the objective of constructing the reasons with which a decision maker will convince himself and other actors involved in the decision-making process (Power 2002; Power and Sharda 2009; Shim et al. 2002). If we examine the research conducted over the past few years, the synergies among technologies of the Internet, World Wide Web (WWW) and Artificial Intelligence (AI) have enabled the single user-focused DSS to evolve into an intelligent and complex Web-based DSS (Web-DSS) (March and Hevner 2007). The current Web-DSS are compatible with new technologies for business intelligence and provide a more transparent interaction between system and decision maker to improve the efficiency and

Published online: 19 July 2012

effectiveness of intelligent decision making (Liu et al. 2010).

With the current proliferation and widespread adoption of e-business, manufacturing and business are extending beyond enterprise boundaries (Norta and Eshuis 2010; Alaranta and Henningsson 2008). Moreover, enterprises are involved in collaboration and mergers with other enterprises on a global scale. This has created a demand for Intelligent Information Integration (III) to integrate, on demand, information from heterogeneous data sources, applications and environments, in order to facilitate inter-operation and collaboration, and provide intelligent decision support through Web-DSS, leading to a more efficient decision-making process at enterprise level (March and Hevner 2007; Seng and Kong 2009; Osei-Bryson and Ngwenyama 2008). It is important to note here that when we refer to 'information' we mean the following:

- the business polices or rules governing the inference mechanism, and
- the data to which the inference mechanism is being applied.

The semantic web, an extension of the current World Wide Web (WWW), is a step towards addressing the aforementioned challenge, i.e. Intelligent Information Integration (III), pertinent to Web-based DSS. The semantic web is seeking a universal medium for data exchange, i.e. classifying, packaging and semantically enriching information for support of data automation, integration, and reuse across various applications (Suguri et al. 2008; Torroni et al. 2009). The core of the semantic web, i.e. ontologies, meta-data and relations for performing inference with rules, is a source of seamless information integration of heterogeneous information sources. However, the initially proposed single stack architecture (SSA) of semantic web by Tim Berners-Lee assumed that the semantic web stack is composed of a main language and every new development should be built on top of existing layers (Berners-Lee 2000; Lee 2003). In response to criticisms that this proposal is unrealistic and unsustainable, Berners-Lee then proposed an alternative multi-stack architecture (MSA) to overcome the limitation of SSA (Lee 2005, 2006). Although the MSA is more realistic in the long run, it provides a basis for only monotonic logic-based knowledge representation and reasoning; it does not provide a basis for representing, reasoning and integrating incomplete and/or contradictory information. In monotonic logic, once a conclusion has been drawn using currently available information, the availability of new information may negate or invalidate existing information. This is a significant problem with monotonic logics and requires re-examination of the entire reasoning cycles when new information, especially that which contradicts existing knowledge, is added to the knowledge base (Hurt 1998). Hence, the current generation of Web-DSS is not able to represent, reason and integrate incomplete and inconsistent information, irrespective of whether this information emerges from within the organization or outside it (Antoniou et al. 2004; Carlsson and Turban 2002). There has been much discussion in the literature on the development of defeasible or non-monotonic logic based Web-DSS systems that can integrate incomplete and conflicting information through reasoning and defining priorities among conflicting rules at compile time (Antoniou and Bikakis 2007; Bassiliades et al. 2004; Grosof et al. 2002). Reasoning is called 'defeasible' or 'non-monotonic' if a rule that supports a conclusion can be defeated by new information. However, such Web-DSS systems have two main limitations:

- They provide a formalism to represent and handle only individual preferences in the form of priorities in order to handle incomplete and conflicting information. However, Web-DSS are subject to inconsistencies deriving from multiple data/information sources and multiple users; therefore, it is not possible to define priorities in advance in order to resolve conflicts among business rules derived from multiple sources/users.
- The use of these priorities is usually embedded in the derivation mechanism and competing rules are compared individually during the derivation process. Therefore, the derivation notion is bound to one single comparison criterion. In such a scenario, the explanation of the results is based on a single criterion only and fails to take into account the multiple factors important for decision-making.

To overcome the limitations of current Web-DSS, we propose and develop a framework for semantic information and knowledge integration at the enterprise level for intelligent decision support. Figure 1 depicts a layered architecture for Intelligent Information Integration (III) spanning the enterprise. It starts with the semantic information integration phase followed by the semantic knowledge integration phase. Let us discuss each phase of III along with the contributions of this article.

1. **Semantic Information Integration (SII)**
   We define SII as the integration of semantic information elicited in the form of business rules and data, residing in different Web-based decision support systems. Our previous work focused on the de-

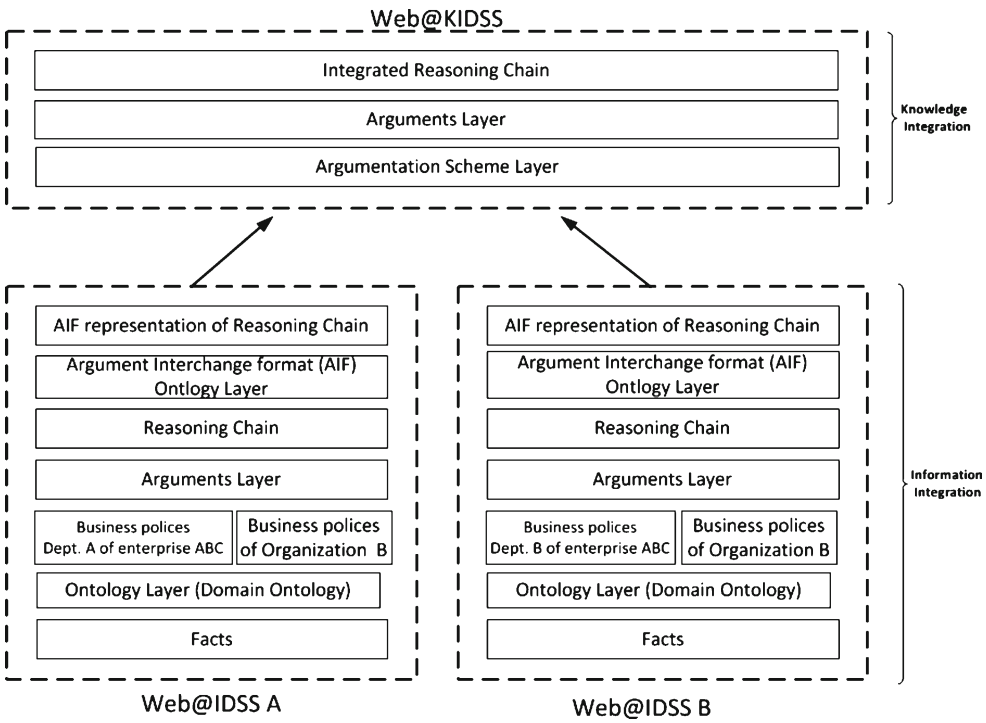


Fig. 1 Layered-view architecture for intelligent information integration (III)

sign and development of an argumentation-enabled intelligent decision support system (Web@IDSS) capable of providing decision support by integrating semantic information which potentially could be incomplete and inconsistent. Sections 4 and 5 provide detailed information about Argumentation and Web@IDSS respectively.

In this article, we extend the functionality of Web@IDSS with AIF reification in order to share the results of Web@IDSS over the enterprise intranet using Argument Interchange Format (AIF) ontology.

2. **Semantic Knowledge Integration(SKI)**

We define SKI as the integration of decisions or results about a subject that are potentially incomplete, inconsistent and distributed amongst different Web-DSS. To integrate the results of different Web-DSS published over the web or enterprise intranet in AIF compliant format, we propose and demonstrate extensions to Web@IDSS to address the issue of knowledge integration at enterprise level with the help of argumentation schemes known as Web-based Intelligent Knowledge Integration DSS (Web@KIDSS). The contributions of this paper to the existing literature body in terms of SKI are as follows:

(a) We propose and demonstrate the integration of different AIF-compatible reasoning chains using a user-defined argumentation scheme.
(b) We provide formal syntax and semantics for argumentative reasoning for semantic knowledge integration.
(c) We also discuss the system architecture in detail followed by algorithms used for semantic knowledge integration.
(d) We illustrate the use of Web@KIDSS with a case study, discuss prototype development, and indicate future directions.

The rest of the paper is structured as follows: Section 2 presents a review of the literature, elaborating and focusing on Semantic Web-DSS and Defeasible logic-based implementations of Web-DSS along with their limitations. Section 3 describes a case study related to information and knowledge integration in an enterprise. Section 4 discusses argumentation and argumentation schemes. Section 5 introduces semantic information integration using Web@IDSS. It describes the working of We@IDSS and AIF reification. This is followed by Section 6 which elaborates upon the knowledge integration process using Web@KIDSS. In this section, we define the formal syntax and semantics for knowledge integration. This section also elaborates on the proposed conceptual framework for semantic knowledge integration and discusses the algorithms in detail. This section is followed by implementation and prototype development in Section 7. Section 8 presents the conclusion and future directions.

2 Review of literature

In an open computing environment, such as the World Wide Web or an enterprise intranet, various decision support systems are expected to work together to support information exchange, processing, and integration. However, DSS are usually built by different people, at different times, to fulfil different requirements and goals leading to (Xue et al. 2009):

1. different supporting infrastructures
2. different syntactic representations of information
3. different schematic designs of information models
4. different semantics of information models.
5. conflicts among information, and the presence of incomplete information hinders its integration into information systems and afterwards knowledge integration at enterprise level.

Mostly, integration efforts have focused largely on the first four issues (Zhou et al. 2010; Seng and Kong 2009; Nguyen et al. 2011). In this paper, we discuss the fifth issue which has received little attention in existing literature: semantic information and knowledge integration in the presence of incomplete and conflicting information.

Recently, knowledge engineers have realized that they need to agree on a shared conceptualization of an application domain, known as an 'ontology' when developing two or more information systems which are syntactically and semantically interoperable (Muthaiyah and Kerschberg 2007). A number of researchers are working on different aspects of ontologies such ontology construction (Kim et al. 2011), ontology mapping (Chua and Goh 2010), ontology tailoring (Flahive et al. 2005, 2009) and materialization of ontological views (Bhatt et al. 2006) etc. Using ontologies, the integration of information distributed among different applications gives the information integration a new level of automation and flexibility ultimately leading to better description, explanation, conjunction, integration and reasoning on some related data, thereby leading to better decision making.

Currently, the use of ontologies for semantic information integration can be viewed from two perspectives. Firstly, ontologies were introduced as a shared, explicit specification of a conceptualization of a domain. Therefore, ontologies lead to integration tasks to describe the semantics of information sources and to make the content explicit. This also focuses on the design and development of common ontologies that can be extended for more specific application domain specification. However, this will exacerbate the integration problem (Noy 2004; Benkö et al. 2003; Buccella et al. 2009; Xue et al. 2009). Secondly, ontologies with extended rules are used for reasoning purposes. This involves an extension of ontologies with rules, where inference and reasoning are central to the process. Here, rules are defined on top of ontologies to infer new knowledge. The proposals for integration of rules languages and ontology languages can be classified by the degree of integration (Antoniou et al. 2005). Firstly, the hybrid approach is one where there is strict separation between the rule predicates and ontology predicates and reasoning is done by interfacing the existing rule reasoner with the ontology reasoned; whereas, with the homogeneous approach, both rules and ontologies are embedded in the same logical language $\mathcal{L}$ without making a prior distinction between the rule predicates and ontology predicates, and the reasoning single reasoner can be used for reasoning purposes.

Many DSS applications are built using the second approach, i.e. ontology with extended rules, to ensure the availability of integrated, high quality information for decision support. Broadly speaking, such approaches fall into two categories:

1. Semantic Web-DSS
2. Defeasible logic-based Web-DSS

In the following sections, we discuss these categories in detail.

2.1 Semantic Web-DSS

The importance of Semantic-based Web DSS in business applications has been identified by a number of researchers over a period of time (Vahidov and Kersten 2004; Silverman et al. 2001; Toni 2007). Kartha and Novstrup (2009) have proposed a combination of ontologies and decision rules for building a decision support application for time sensitive targeting. They have represented knowledge with the help of rules known as decision rules which: (a) include primitives from multiple ontologies and primitives that are defined by algorithms that are outside of the rule framework; (b) are time-dependent; and (c) incorporate default assumptions. They have developed what is known as the Sentinel system, which is general enough to support a wide variety of DSS tasks.

Ceccaroni et al. (2004), present an environmental decision support system (called OntoWEDSS) for waste

water treatment to improve the diagnosis of faults in a treatment plant, which provides support for complex problem-solving and facilitates knowledge modelling and reuse. The system is based on the integration of case-based and rule-based reasoning with an ontology, i.e. Waste-Water Ontology (WaWO), for the representation of the domain and for reasoning. Nicolicin-Georgescu et al. (2010) present an approach to managing data warehouse cache allocations via decision support systems, by using autonomic computing and semantic web technologies. He has presented heuristics for autonomic computing adoption, using ontologies for DSS system modelling and ontology-based rules for heuristics implementation.

Similarly, Salam (2007) presents a supplier performance contract monitoring and execution DSS, using OWL-DL[1] for knowledge representation SWRL[2] to express rules on top of OWL-DL ontologies. Cheung and Cheong (2007) address the challenges of market operations using a rule-based approach in mission-critical decisions and Garcia-Crespo et al. (2011) propose a semantic model for knowledge representation in e-business. Yang et al. (2009) have proposed a Semantic Web-DSS and provide semantics for defining static and dynamic semantics representation based on ontology and quantitative decision making comprising three steps: publishing decision requirement, bidding and then role-based collaboration among decision peers (each Semantic Web-DSS is a peer) to negotiate for decision models.

In all these attempts, the systems integrate information through reasoning with the help of ontologies under certain assumptions including:

1. The given problem can be fully addressed with available information (solution to the problem lies within the available information). In order to elucidate it, let us consider an example. A department in an enterprise wants to improve its product and would like to make use of all the information it holds internally in order to adequately identify issues regarding product quality and improve the product's quality. The department ignores any information outside its own boundaries.
2. The information or specification of business rules for decision-making is consistent. In other words, it is assumed that no contradictory rules will emerge during the decision-making process
3. New information will be consistent with the already available information or specifications.
4. New information does not lead to retraction of previous conclusions.

In the existing literature, there is no research on enterprise-wide Web-DSS that addresses the aforementioned issues of information integration for intelligent decision-making.

2.2 Defeasible logic-based Semantic Web-DSS

This is the second category of Web-DSS having the capability of integrating information which could be incomplete and inconsistent. In this type of Web-DSS, the special types of rules known as defeasible rules are deployed to incorporate defeasible or non-monotonic behaviour in the system.

Dr Prolog (Antoniou and Bikakis 2007) is a Prolog-based implementation for carrying out defeasible reasoning on the semantic web. It provides declarative system support rules, facts, ontologies, RuleML, and both monotonic and non-monotonic rules. The system provides a number of variants such as ambiguity blocking, ambiguity propagation and conflicting literals.

Dr-Device (Kontopoulos et al. 2011; Bassiliades et al. 2004) is CLISP-based defeasible reasoning implementation for information integration provided with a VDR-Device reasoning system. Compared to Prolog, Dr-Device supports only one variant for information integration, i.e. ambiguity blocking.

Sweetjess (Grosof et al. 2002) is another defeasible reasoning system based on Jess and closely resembles courteous logic programs. It allows for procedural attachment and it implements only one reasoning variant. Moreover, it imposes a number of restrictions on the programs so that it can map on Jess.

Table 1 compares different defeasible logic-based semantic web implementations. In the context of semantic Web-DSS, these implementations have various limitations. Firstly, they provide either data-driven or goal-driven reasoning. Data-driven moves from current facts to a certain conclusion, whereas goal-driven reasoning is used to validate the conclusion with supporting facts and answer the user queries. However, in the case of semantic Web-DSS, both types of reasoning are needed for information integration. The existing proposed approaches cannot handle both types of reasoning for information integration. Secondly, they define

[1] http://www.w3.org/TR/owl-guide/

[2] http://www.w3.org/Submission/SWRL/

Table 1 Comparison of defeasible logic based semantic Web-DSS

	Dr-Prolog	Dr-Device	Situated courteous logic
Language	Prolog	JESS	JESS
Logic	Defeasible logic	Defeasible logic	Situated courteous logic
Semantic data	RDFS/OWL	RDF	DAML+OIL
RuleML	Yes	Yes	Yes
Incomplete knowledge representation	Yes	Yes	Yes
Conflict representation	Yes	Yes	Yes
Data-driven reasoning	No	Yes	Yes
Goal-driven reasoning	Yes	No	No
Conflict resolution	User defined individual preferences	User defined individual preferences	User defined individual preferences
Explanation	Textual	Textual	Textual
AIF reification	No	No	No
Information integration	Limited	Limited	Limited
Knowledge integration	No	No	No

explicit (user- defined) individual preferences among conflicting rules at compile time to resolve conflicts between them. The use of these priorities is usually embedded in the derivation mechanism and conflicting rules are compared individually during the derivation process. In such formalisms, the derivation notion is bound to one single comparison criterion. However, the semantic Web-DSS is a source of defeasible knowledge as it is open by nature and subject to inconsistencies deriving from multiple sources; therefore, it is not possible to define priorities in advance among conflicting rules and even if priorities exist, it is not appropriate to compare rules individually during the derivation process. As a result, these systems provide limited information integration and no knowledge integration at all. Additionally, all of them provide to the end user only a textual explanation about the integrated information, and the integrated information results are not exportable in Argument Interchange Format (AIF).

3 Case study

Let us assume that ABC is an enterprise comprised of different departments such as IT, Marketing, and Human Resources. Due to certain unavoidable circumstances, the enterprise has decided to relocate its departments to a new site. Higher authorities have instructed the managers of each department to give their recommendations along with justifications about the suitability of using the XYZ relocation service provider for relocation purposes. The XYZ relocation service provider has its business-related information published on the web, giving all the necessary information required potential customer such as ABC enterprise. Figure 2 depicts the interaction between the internal environment of enterprise ABC and the external environment.

In order to generate recommendations for XYZ, the managers need an automated information integration system which could automatically access, reason and integrate XYZ business information along with their departmental information (requirements). It is important to note here that the information from service provider XYZ could be potentially incomplete and contradict departmental information of enterprise ABC. Additionally, the managers also want to incorporate user feedback provided by existing XYZ clients via a feedback forum, in the decision-making process. On successful information integration, managers need to forward their recommendations to higher authorities in a standard format.

Once each department within the ABC enterprise forwards its recommendation to higher authorities, the

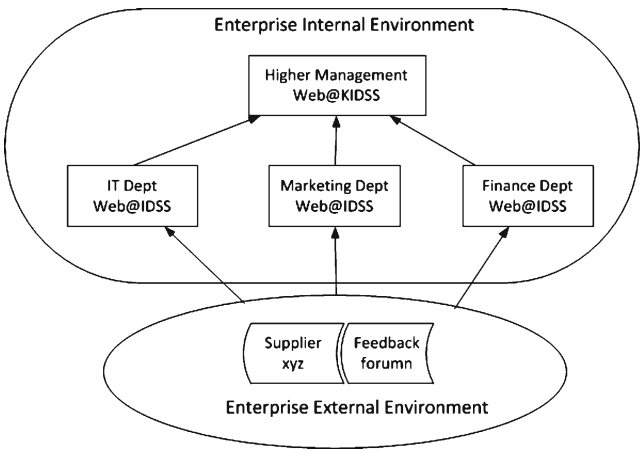


Fig. 2 Interaction of enterprise ABC with external environment

latter will need to integrate the knowledge or recommendations about XYZ obtained from each department into a coherent information model which could help them to reach a final decision, i.e. whether or not to engage the services of the XYZ relocation service provider.

Here we formalize the requirements for enterprise ABC to successfully achieve the aforementioned tasks.

- A declarative language for specifying the business requirements of an organization
- Language with the capability of representing incomplete and conflicting information (i.e. business rules and data)
- Information integration via an inference mechanism that can perform reasoning pertaining to incomplete and conflicting information coming from different sources.
- Justifiable explanation of the information integration results
- Ability to export results to other software systems
- Knowledge integration through a user-defined scheme that allows the user to define constraints pertaining to knowledge to be integrated.

Assumption

- Organizations ABC, XYZ and Feedback forum share a common vocabulary defined in OWL/RDF format and the predicates defined in the vocabulary are used for the specification of business rules.

4 Argumentation and argumentation schemes

In everyday life communication, argumentation often has negative connotations, suggesting quarrelsomeness and unpleasantness; this is a misconception. In fact, argumentation in the classical sense is the study of effective reasoning which is a key to the way that humans deal with conflicting information by taking into account arguments and counter-arguments relevant to a certain issue (Zarefsky 2009). Argumentation is inherently a process rather than an instant picture, and the building blocks of argumentation are the arguments and the relationship between those arguments (Loui 1998). According to Walton (2009) and Palau and Moens (2009), an argument is a set of statements (propositions) consisting of a conclusion, a set of premises, and inference from premises to conclusion. During the process of argumentation, relationships among the arguments are linked with each other in a certain pattern to support the ultimate conclusion. Such linking patterns are called 'argumentation schemes' and allow reasoning to be performed using a set of premises and a conclusion. These argumentation schemes have emerged from informal logic (Godden and Walton 2007). The schemes help to categorize the way that arguments are built. They bridge the gap between logic-based application and human reasoning by capturing stereotypical patterns of human reasoning. An example is an argument from an expert opinion scheme. Formally, an argumentation scheme is composed of a set of premises A_i, a conclusion denoted as S, and a set of critical questions CQ_i aimed at defeating the derivation of the consequent (Letia and Groza 2008; Rahwan et al. 2007a).

Argumentation formalism has been used in the past by different researchers in DSS for practical reasoning. Morge (2008) proposed a DSS based on abductive reasoning which helps the decision maker to select a business location after evaluating different alternatives; it suggests several solutions and provides an interactive and intelligible explanation of those choices. To develop the DSS, Morge (2008) used a logic language to represent knowledge, goals and actions with quantitative priorities attached to them to represent likelihood of knowledge, preferences between goals and expected utilities of actions respectively. Similarly, Chesnevar et al. (2006b) identified that the current critic recommender systems are incapable of dealing with the defeasible nature of information. They present a novel approach to the integration of DSS, such as critics and recommender systems with a defeasible argumentation framework, to enhance the practical reasoning capabilities of such systems.

5 Argumentation enabled semantic information integration

The use of ontologies have helped to improve the integration of information derived from different sources to a new level of automation and flexibility. This has ultimately led to better description, explanation, conjunction, integration and reasoning on some related data, resulting in better decision-making. However, the current generation of Web-DSS is not able to represent, reason and integrate incomplete and inconsistent information for information integration purposes, irrespective of whether this information emerges from within the organization or outside it (Carlsson and Turban 2002).

We have addressed this challenge (representation and reasoning over incomplete and inconsistent semantic information) by using Web@IDSS (Janjua and

Hussain 2011). The Web@IDSS uses Defeasible logic-based argumentation formalism (DeLP)(Garcia and Simari 2004) as knowledge representation and reasoning language with certain extensions.

$$\mathcal{WM} = \left\{ \begin{array}{c} relocationService(xyz), client(it), useService(xyz), efficient(xyz), safeDelivery(xyz) \\ language(english), languageProblem(xyz, english) \end{array} \right\} illustration(1)$$

$$\mathcal{R} = \left\{ \begin{array}{l} [x.d1] relocationService(X), client(Y), useService(X) \dashrightarrow \sim giveDiscount(X) \\ [x.d2] client(Y), relocationService(X), reuseService(Y, X) \dashrightarrow giveDiscount(Y) \\ [x.d3] efficient(X), safeDelivery(X) \dashrightarrow reliableService(X) \\ [x.s1] giveDicount(Y), advancmentPayment(Y) \rightarrow normalDiscount(Y) \\ [x.s2] normalDiscount(Y), bulkOrder(Y) \rightarrow platinumDiscount(Y) \\ [a.it.d1] client(Y), happy(Y, X), relocationService(X) \dashrightarrow resuseService(Y, X) \\ [a.it.d3] ontimeDelivery(X) \dashrightarrow efficient(X). \\ [rc1.a.it.d4] not\ dmanageProduct(X) \dashrightarrow safeDelivery(X) \\ [a.it.d5] largeTruck(X), reuseService(Y, X), reliableService(X), normalDiscount(Y) \\ \dashrightarrow goodRelocationService(X) \\ [a.it.d6] language(ENG), languageProblem(X, ENG) \dashrightarrow \sim clearCriteria(X) \\ [a.it.d7] demandCash(X), demandTip(X) \dashrightarrow \sim convienent(xyz) \\ [a.it.d8] goodRelocationService(X), not\ convienent(X), not\ clearCriteria(X) \\ \dashrightarrow recommendService(X) \end{array} \right\} illustration(2)$$

$$Args = \{\} \ldots\ldots illustration(3)$$

The system is capable of translating business rules defined in RuleML[3] syntax into DeLP rules. The system also translates the RDF/XML data into DeLP facts and RDF(S) and part of OWL Ontologies into DeLP facts and DeLP rules. Once the information such as business rules and data translation is complete, the system starts an argumentation construction cycle.

Lets us assume that illustrations 1, 2 and 3 depicts the initial state of the argumentative production system of the IT department of enterprise ABC. Illustration 1 represents current information saved in the working memory ($\mathcal{WM}$) and illustration 2 represents business policies i.e. business rules ($\mathcal{R}$) of the IT department and XYZ service provider, each identified by their respective label, and illustration 3 depicts an empty active argument set ($Args$) before the arguments construction cycle.

$$\mathcal{WM}' = \left\{ \begin{array}{c} relocationService(xyz), client(it), useService(xyz), efficient(xyz), safeDelivery(xyz) \\ language(english), languageProblem(xyz, english), resuseService(it, xyz), giveDiscount(it), \\ normalDiscount(it), reliableService(xyz), goodRelocationService(xyz), \sim clearCriteria(xyz), \\ \sim convienent(xyz), recommendService(xyz) \end{array} \right\} illustration(4)$$

$$Args = \left\{ \begin{array}{l} [rc1.a.it.d1] client(it), happy(it, xyz), relocationService(xyz) \dashrightarrow reuseService(it, xyz) \\ [rc1.x.d2] client(it), relocationService(xyz), reuseService(it, xyz) \dashrightarrow giveDiscount(it); \\ [rc1.x.s1] giveDicount(it), advancmentPayment(it) \rightarrow normalDiscount(it) \\ [rc1.a.it.d3] ontimeDelivery(xyz) \dashrightarrow efficient(xyz). \\ [rc1.a.it.d4] not\ dmanageProduct(xyz) \dashrightarrow safeDelivery(xyz). \\ [rc1.x.d3] efficient(xyz), safeDelivery(xyz) \dashrightarrow reliableService(xyz) \\ [rc1.a.it.d5] largeTruck(xyz), reuseService(it, xyz), reliableService(xyz) \\ , normalDiscount(xyz) \dashrightarrow goodRelocationService(xyz) \\ [rc1.a.it.d6] language(english), languageProblem(xyz, English) \dashrightarrow \\ \sim clearCriteria(xyz) \\ [rc1.a.it.d7] demandCash(xyz), demandTip(xyz) \dashrightarrow \sim convienent(xyz) \\ [rc1.a.it.d8] goodRelocationService(xyz), not\ convienent(xyz), \\ not\ clearCriteria(xyz) \dashrightarrow recommendService(xyz) \end{array} \right\} illustration(5)$$

[3] http://ruleml.org

The argument construction is a recursive process which involves interpretation of business rules. The system searches for business rules from a knowledge base whose pattern matches the facts present in working memory ($\mathcal{WM}$) and, on a successful match, executes the business rule which then adds the rules conclusion, i.e. ground predicate, to the working memory. The argument construction process continues until all the matched business rules in the knowledge base have been processed. This interpretation of a business rule is also known as 'firing of a rule'. Illustrations 4 and 5 depicts the updated argumentative production system of the IT department with a populated active argument set and updated working memory($\mathcal{WM}\prime$).

The argument construction followed by the argumentation phase consists of conflict detection, resolution and justified explanation. The conflict between an argument and its counter-argument is resolved either by static static priority establishment or dynamic priority establishment. For dynamic priority establishment, the process of argumentation starts where an argument may be defeated by other arguments. Since counter-arguments are also arguments which in turn may be defeated, this process results in the construction of dialectical trees. This is an interesting property of the argumentation approach which involves dialectical proof procedures that are quite close to the process used by humans when discussing an issue. This similarity to human-style discussions gives argumentation an advantage that can be useful in many contexts. As a result of argumentation, the development of dialectical analysis for conflict resolution amongst arguments leads to the establishment of dynamic priorities between conflicting arguments.

The last step in the argumentation process is the construction of reasoning chains. During this process, all the sub-arguments of an argument including arguments in conflict with undefeated dialectical trees are linked together as a reasoning chain. This process continues until all possible arguments are linked up into reasoning chains. The top argument i.e. conclusion, of the reasoning chain is known as 'the 'result' of the reasoning chain', and the chain of sub-arguments supporting the top argument is called the 'support' of the result. Figure 3 is a graphical representation of a reasoning chain of the argumentative production system depicted in illustrations 4 and 5. The arguments in Fig. 3 are represented in short form e.g. $[rc1.a.it.d8]recommendService(xyz)$ where [rc1.a.it.d8] is the label of the argument and $recommendService(xyz)$ is the claim of the argument.

5.1 Argument interchange format (AIF) reification

The AIF is an international effort to develop a representational mechanism for exchanging argument resources between research groups, tools, and domains using a semantically rich language (Chesnevar et al. 2006a; Iyad Rahwan 2009; Rahwan et al. 2007b). The AIF was developed as a commonly agreed upon core ontology i.e. AIF ontology that specifies the basic concepts used to express arguments and the relationship between arguments. Figure 4 depicts the AIF core ontology. The upper ontology describes the arguments in the form of a connected network of nodes. The nodes are of two types, namely, information nodes (I-Node) and scheme nodes (S-nodes). The I-Node captures the information in the form of premise, conclusion, exception or presumption. The scheme nodes

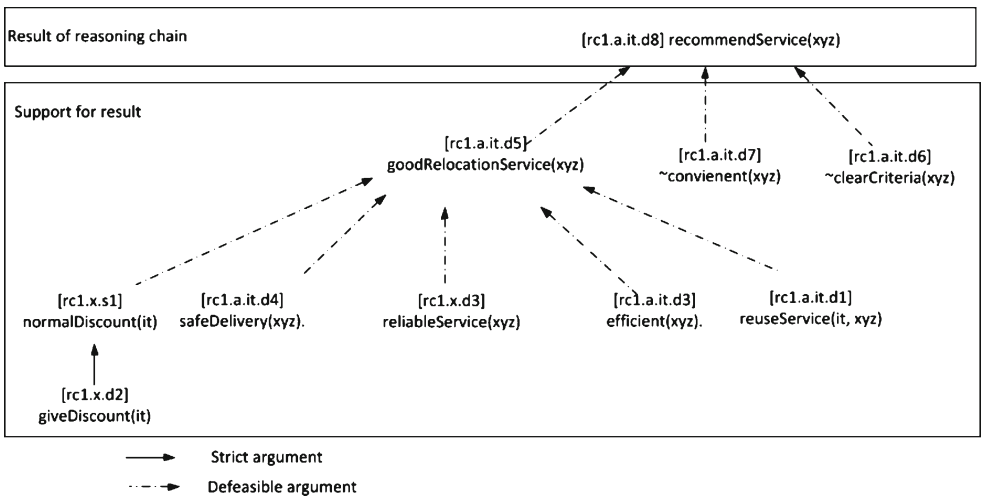


Fig. 3 Graphical representation of a reasoning chain

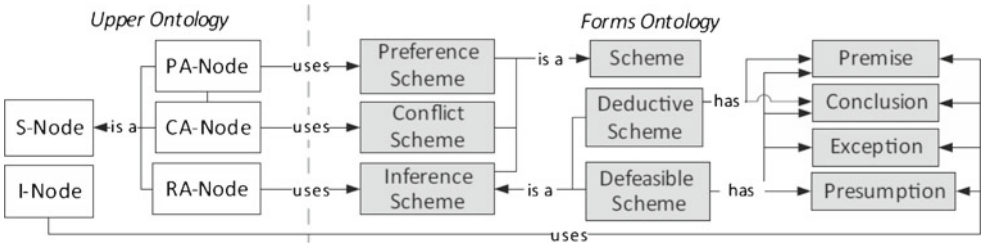


Fig. 4 The upper and forms ontologies of the AIF (Bex et al. 2010)

provide the relationship between two I-Nodes and are further classified as rule application nodes (RA-Node) that correspond to inference from premises to claim, conflict nodes (CA-Node) that correspond to conflict between two nodes and preference application nodes (PA-node) that correspond to preference ordering between conflicting nodes. These different kinds of nodes are used to build the AIF argument graph defined by Rahwan et al. (2007b) as follows:

Definition 1 (Argument Network) An argument network Φ is a graph G consisting of

- a set $\mathcal{N}$ of vertices (or nodes) comprises of I-Nodes and S-Nodes; and
- a binary relation $\xrightarrow{\text{edge}}$: $\mathcal{N}$ x $\mathcal{N}$ representing edges among nodes

such that $\nexists\ (i, j) \in \xrightarrow{\text{edge}}$ where both $i \in \mathcal{N}_1$ and $j \in \mathcal{N}_1$.

The I-Node can be connected to other I-Nodes only through S-Nodes to represent a rationale behind the relationship between I-Nodes. However, the S-Node can be directly connected to other S-Nodes to represent some kind of meta-reasoning. For instance, RA-to-RA and RA-to-PA edges might indicate some kind of meta-justification for the application of an inference rule or a particular criterion for defining preference. Additionally, the ontology does not use typed edges in a graph; instead, the semantics for edges can be inferred node types that they connect. The argumentation schemes are represented as forms of arguments known as 'forms ontology' in AIF core ontology as depicted in Fig. 4. Those argumentation schemes are preference scheme, conflict scheme and inference scheme. The inference scheme is again divided into a deductive scheme which represents deductive inference from premises to a claim, and a defeasible scheme which represents defeasible inference from premises to a claim.

The AIF reification in the Web@IDSS will help to express arguments in more concrete language, and with better representation and handling of conflicts in an argument network and better evaluation of complex arguments and reasoning chains. The export of results provided by Web@IDSS in AIF-compliant format will enable the system to merge it with the justified conclusions of other machines in the absence of complete or accurate information.

In the rest of this section, we provide formalisms that make use of an 'Argument Network' to represent integrated information as argumentative reasoning chains.

Definition 2 (Web@IDSS argument network) Given an argument graph G and set of forms F in a argument network Φ, a Web@IDSS argument network AG is defined as follows: ($\mathcal{WM}$, $\mathcal{R}$, $Args$) Where

- $\mathcal{WM}$: a set of information nodes i.e. $\mathcal{N}^I_{i,\ldots,n}$, where I represents information node and i represents index of the node.
- $\mathcal{R}$: a set of user-defined rules or specifications to establish links between $\mathcal{N}^I_i$ nodes through S node such that $\nexists\ (i, j) \in \xrightarrow{\text{edge}}$ where both $i \in \mathcal{N}_1$ and $j \in \mathcal{N}_1$
- $Args$ a set of arguments derived from $\mathcal{R}$, where each argument establishes a linked set of premise ($\mathcal{N}^I_i$) to a claim($\mathcal{N}^I_j$) through S node. Based upon the forms of ontology, the strict argument and defeasible argument are defined as follows:

(Strict argument) :

$$\mathcal{N}^I_i, \ldots\ldots, \mathcal{N}^I_j \xrightarrow{\text{Uses(RA,deductiveScheme)}} \mathcal{N}^I_k$$

(Defeasible argument) :

$$\mathcal{N}^I_m, \ldots\ldots, \mathcal{N}^I_n \overset{\text{Uses(RA,defeasibleScheme)}}{\dashrightarrow} \mathcal{N}_o$$

The binary relation $\xrightarrow{\text{edge}}$: $\mathcal{N}$ x $\mathcal{N}$representing edge among nodes in Web@IDSS can be categorized as follows:

- Counter-argument: $\mathcal{N}^I_i \overset{\text{Uses(CA-Node)}}{\dashrightarrow} \sim \mathcal{N}^I_j$ such that $\mathcal{N}^I_i$ is counter-argue $\mathcal{N}^I_j$

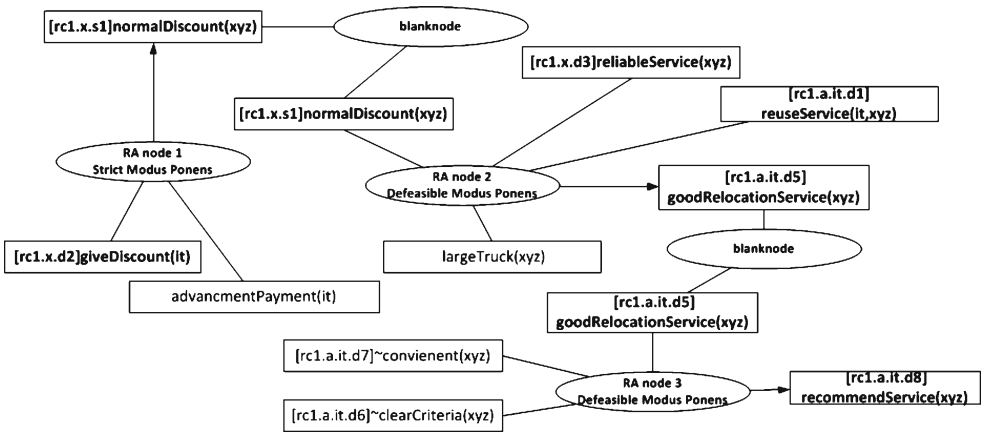


Fig. 5 Pictorial representation of a AIF compliant reasoning chain

- Static defeat: $\mathcal{N}_i^I \overset{\text{Uses(PA-Node)}}{\dashrightarrow} \mathcal{N}_j^I$ such that $\mathcal{N}_i^I$ is has priority over $\mathcal{N}_j^I$
- Dynamic defeat: $\mathcal{N}_i^I \overset{\text{Uses(PA-Node)}}{\dashrightarrow} \mathcal{N}_j^I$ such that $\mathcal{N}_i^I$ has priority over $\mathcal{N}_j^I$
- Sub-argument: For representation of sub-argument relationship in AIF format, we added a blank-node into argument network i.e. $\mathcal{N}_i^I \overset{\text{Uses(Blank-Node)}}{\dashrightarrow} \mathcal{N}_j^I$ such that $\mathcal{N}_i^I$ (claim of an argument) is sub-argument of $\mathcal{N}_j^I$ (premise of an argument).

Definition 3 (Predecessor and Successor Nodes) Given a graph AG consisting of a set of nodes $\mathcal{N}$ and a relation $\mathcal{S} \subseteq \mathcal{N} \times \mathcal{N}$ defining the set of edges between the nodes. For each node n $\in \mathcal{N}$, we define the set of its predecessor and successor nodes as follows:

- A Predecessor node: $\{x \in \mathcal{N} \mid (x, n) \in \mathcal{S}\}$,
- A Successor node: $\{x \in \mathcal{N} \mid (n, x) \in \mathcal{S}\}$.

Using the above definitions, we represent the reasoning chains produced by Web@IDSS in AIF format. The node $\mathcal{N}_i^I$ with no successor and have predecessor nodes

Fig. 6 Serialization of AIF compliant reasoning chain in turtle format

```
@prefix RC: <http://www.abc.org/it/RC.owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl2xml: <http://www.w3.org/2006/12/owl2-xml#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix : <http://www.w3.org/2006/12/owl2-xml#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix kb: <http://protege.stanford.edu/kb#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://www.w3.org/2002/07/owl#> .
[ rdf:type owl:Ontology ;
  owl:imports <file:/D:/data/Downloads/ArgDFProtegeOntology/ArgDFProtegeOntology/ArgOnt.rdfs>
] .
###  http://www.abc.org/it/RC.owl#DefeasibleInference
RC:DefeasibleInference rdf:type owl:Class ;rdfs:subClassOf kb:RuleScheme .
###  http://www.abc.org/it/RC.owl#recommendations
RC:recommendations rdf:type owl:Class ; rdfs:subClassOf owl:Thing .
###  http://protege.stanford.edu/kb#hasPremise
kb:hasPremise rdf:type owl:NamedIndividual .
###  http://www.abc.org/it/RC.owl#RAnode1
RC:RAnode1 rdf:type kb:RA-Node ,  owl:NamedIndividual ;
           kb:hasPremiseDescription RC:advancePayment(it) ,
                                    RC:giveDiscount(it) ;
           kb:hasConclusionDescription RC:normalDiscount(it) ;
           kb:edgeFromSNode RC:normalDiscount(it) .
###  http://www.abc.org/it/RC.owl#advancePayment(it)
RC:advancePayment(it) rdf:type kb:I-Node , owl:NamedIndividual ;
                      kb:hasPremiseDescription kb:hasPremise .
###  http://www.abc.org/it/RC.owl#giveDiscount(it)
RC:giveDiscount(it) rdf:type kb:I-Node , owl:NamedIndividual ;
                    kb:edgeFromINode RC:RAnode1 .
###  http://www.abc.org/it/RC.owl#normalDiscount(it)
RC:normalDiscount(it) rdf:type kb:I-Node , owl:NamedIndividual .
...........
```

is called the 'result' of reasoning chain. The remaining nodes are known as 'support' for the result. Figure 5 depicts the graphical representation of a reasoning chain in AIF format.[4]

5.2 Publication of reasoning chains

The purpose of AIF reification is to publish and share the results of a Web-DSS over the web, enterprise intranet or with other applications in order to provide better decision-making support. For annotation of a reasoning chain, we developed a 'reasoning chain ontology' on top of the ArgDF ontology[5] and serialized the AIF-compliant reasoning chain in RDF/XML format. Figure 6 depicts the serialization of a reasoning chain in turtle format.

6 Argumentation-scheme-enabled, argumentation-driven semantic knowledge integration

Today, the decision-making environment has become very complex and decentralized, exacerbated by WWW. Results produced by one Web-DSS might need to be integrated with other Web-DSS to obtain a comprehensive picture of the problem at enterprise level to enable higher authorities to gain business insights and make better decisions. We call such information integration that is related to one subject and distributed among different information sources 'knowledge integration'. Let us consider the case study discussed in Section 3 where each department needs to formulate and forward its recommendations about the relocation service provider XYZ to higher authorities. During this process, each department, with the help of Web@IDSS or AIF-compliant DSS system, produces recommendations in the form of a reasoning chain. Let us assume that according to the IT department, although the relocation service provider is not convenient and not good at formalising the clients' criteria, still we assume it is a good relocation service provider and we recommend it. Whereas, other departments have a different opinion. According to illustrations 6, 7 and 8 which depict the recommendations produced by the IT, Marketing and Human Resources departments respectively, it is quite evident that each department has some valuable information about relocation service supplier XYZ, which could help the higher authorities to make the final decision about this supplier. But the biggest challenge is how to automate the integration of this knowledge which is derived from different sources and could be incomplete and inconsistent, to facilitate the decision making process at enterprise level.

$$IT = \left\{ \begin{array}{l} [rc1.a.it.d1]client(it), happy(it, xyz), relocationService(xyz) \dashrightarrow reuseService(it, xyz) \\ [rc1.x.d2]client(it), relocationService(xyz), reuseService(it, xyz) \dashrightarrow giveDiscount(it); \\ [rc1.x.s1]giveDicount(it), advancmentPayment(it) \rightarrow normalDiscount(it) \\ [rc1.a.it.d3]ontimeDelivery(xyz) \dashrightarrow efficient(xyz). \\ [rc1.a.it.d4]not\ dmanageProduct(xyz) \dashrightarrow safeDelivery(xyz). \\ [rc1.x.d3]efficient(xyz), safeDelivery(xyz) \dashrightarrow reliableService(xyz) \\ [rc1.a.it.d5]largeTruck(xyz), reuseService(it, xyz), reliableService(xyz), \\ normalDiscount(xyz) \dashrightarrow goodRelocationService(xyz) \\ [rc1.a.it.d6]language(english), languageProblem(xyz, english) \dashrightarrow\sim clearCriteria(xyz) \\ [rc1.a.it.d7]demandCash(xyz), demandTip(xyz) \dashrightarrow\sim convienent(xyz) \\ [rc1.a.it.d8]goodRelocationService(xyz), not\ convienent(xyz), not\ clearCriteria(xyz) \\ \dashrightarrow recommendService(xyz) \end{array} \right\} illustration(6)$$

[4]The directed arrow are just to emulate the edge from S-node to N-node claim of the argument.

[5]http://www.argdf.org/source/ArgDFProtegeOntology.zip

$$Mar = \left\{ \begin{array}{l} [rc2.a.mk.d1] not\ happy(marketing, xyz), relocationService(xyz) \dashrightarrow\sim reuseService(marketing, xyz) \\ [rc2.x.d1] relocationService(xyz), client(marketing), useService(xyz) \dashrightarrow \\ \sim giveDiscount(marketing) \\ [rc2.a.mk.d3] ontimeDelivery(xyz), largeTruck(xyz), \dashrightarrow efficient(xyz) \\ [rc2.a.mk.d5] dmanageProduct(xyz) \dashrightarrow\sim safeDeliver(xyz) \\ [rc2.a.mk.d4] not\ efficent(xyz), not\ reuseService(xyz), not\ giveDiscount(marketing), \\ not\ safeDeliver(xyz) \dashrightarrow\sim goodRelocationService(xyz) \\ [rc2.a.mk.d6] not\ goodRelocationService(xyz) \dashrightarrow\sim recommendService(xyz) \end{array} \right\} illustration(7)$$

$$HR = \left\{ \begin{array}{l} [rc3.a.hr.d1] language(english), languageProblem(xyz, english) \dashrightarrow\sim clearCriteria(xyz) \\ [rc3.x.d2] client(hr), relocationService(xyz), reuseService(hr, xyz) \dashrightarrow giveDiscount(hr) \\ [rc3.a.hr.d2] not\ ontimeDelivery(xyz) \dashrightarrow\sim efficient(xyz) \\ [rc3.a.hr.d3] not\ efficient(xyz), not\ giveDiscount(xyz) \dashrightarrow\sim goodRelocationService(xyz). \\ [rc3.a.hr.d4] not\ goodRelocationService(xyz), not\ clearCriteria(xyz), \dashrightarrow \\ \sim recommendService(xyz) \end{array} \right\} illustration(8)$$

In the following section, we define the formal syntax and semantics for Argumentation scheme enabled Argumentative Knowledge Integration Web-DSS (Web@KIDSS). The proposed system is capable of integrating knowledge regarding one subject derived from different Web-based DSS into a coherent, consolidated form so that higher authorities can have better business insights and make the optimal decisions or decisions that take a broader perspective.

6.1 Formal syntax and semantics

Definition 4 (Recommendation space) A collection of recommendations, each in the form of a reasoning chain $\lambda_{(\text{identifer,result})}$ contributed by a source 'i' is known as a 'recommendation space'. Mathematically, recommendation space is defined as follows:

$$\Theta = \sum_{i=0}^{n} \left\{ [i]\lambda_{(\text{identifer,result})} \right\} \quad (1)$$

The recommendation space' for enterprise ABC is depicted in Fig. 7 can be mathematically represented as follows:

$$\Theta = \left\{ [rc1]\lambda_{(\text{A,recommend})}, [rc2]\lambda_{(\text{B,}\sim\text{recommend})}, [rc3]\lambda_{(\text{C,recommend})} \right\} \quad (2)$$

Where

- $[rc1]\lambda_{(A,a)}$ represents the recommendation in the form of a reasoning chain by department IT department identified as rc1.
- Similarly, $[rc2]\lambda_{(B,\sim a)}$ is a recommendation from the HR department identified as rc2 and $[rc3]\lambda_{(C,a)}$ is a recommendation from the marketing department identified as rc3.

Definition 5 (Integration scheme) An Integration scheme, a user-defined argumentation scheme (Katie Atkinson 2008), is a tuple having the following form:

$$\mathcal{IS} = \{\{name, (premise_i, ...premise_n), conclusion, criticalquestions, variant\} \quad (3)$$

Where

- name is the label of the scheme which identifies the scheme
- premise is a set of facts to be matched
- sConclusion is a result of the scheme
- sCriticalquestion is a set of queries
- svaraint is a boolean flag for conflicts blocking. If svariant is true, the conflicts are blocked and the reasoning chain will not considered for any further processing; whereas, if the flag is false, then the reasoning chains with conflicts are still considered for further processing.

The critical questions can be categorized as exceptions and assumptions. The premises provide reasons for accepting the conclusion only if the assumptions are true and there are no exceptions. If either an assumption is false or an exception is true, unless premises provide reasons for accepting the conclusion, the conclusion would not be valid (Katie Atkinson 2008). Thus, both assumption and exceptions attack the conclusion of the scheme.

In the context of the case study discussed above, the integration scheme represents the criteria that are used by higher authorities to evaluate each department's recommendation before their integration. If the recommendation meets the integration criteria, then the reasoning chain is included in the knowledge integration process.

Definition 6 (Valuation operator) The application of the integration scheme to a reasoning chain is termed 'valuation of a reasoning chain'. Mathematically, we define the valuation operator $\lozenge$ as a binary operator such as

$$[rc1]\lambda^{\text{val}}_{(A,a)} = \left\{[rc1]\lambda_{(A,a)} \lozenge \mathcal{IS}\right\} \tag{4}$$

During valuation of a reasoning chain, all the premises and critical questions originating from the integration scheme are executed on the corresponding reasoning chain. If the premises match the reasoning chain and queries return true on execution over the reasoning chain, then the reasoning chain is considered to be a valued reasoning chain. The reasoning chain is still considered valued if the reasoning chain premise does not match or queries return false, but the conflict blocking flag, i.e. svariant is false.

Definition 7 (Focus operator) $\otimes$ is a binary operator, such that

$$[rc1]\lambda^{\text{val}}_{(A,a)} \otimes [rc2]\lambda^{\text{val}}_{(B,a)} \tag{5}$$

is called a 'focus operator'. This corresponds to AND operator. If two arguments, belonging to different reasoning chains, have the same claim, the application of the focus operator produces those arguments in a resultant set. Let us consider the recommendation space depicted in Fig. 7, the application of the focus operator to reasoning chains rc2 and rc3 results in the following set of common claims $\{[mk.d4, hr.d4] \sim recommendService(xyz), [mk.d6, hr.d6] \sim goodRelationService(xyz)\}$.

Definition 8 (Merge operator (Fan et al. 2010)) $\boxplus$ is a binary operator, such that

$$[ar1]a, b, c \dashrightarrow d \boxplus [ar2]e, b, c \dashrightarrow d \tag{6}$$

is called a 'merge operator'. This corresponds to the OR operator. Let us consider the recommendation space depicted in Fig. 7, containing two argument 'hr.d6' argument and 'mk.d6' argument belonging to reasoning chain rc2 and rc3 respectively. The application of the merge operator to these arguments results in the construction of a new argument which would look like:

[hr.d6][mk.d6]$\sim goodRelocationService(xyz), \sim clearCriteriac(xyz) \dashrightarrow \sim recommendService(xyz)$.

The merge operator applies to the arguments with the same inference type.

Definition 9 (Unique operator) $\odot$ is a binary operator, such that

$$[rc1]\lambda^{\text{val}}_{(A,a)} \odot [rc2]\lambda^{\text{val}}_{(B,a)} \tag{7}$$

is called a 'unique operator'. The application of unique operator on reasoning chains results in all those arguments whose claim is unique between the reasoning chains. Let us consider the recommendation space depicted in Fig. 7, the application of unique operator on reasoning chains rc2 and rc3 results in following set of arguments $\{\sim clearCriteria(xyz), giveDiscount(mk),$

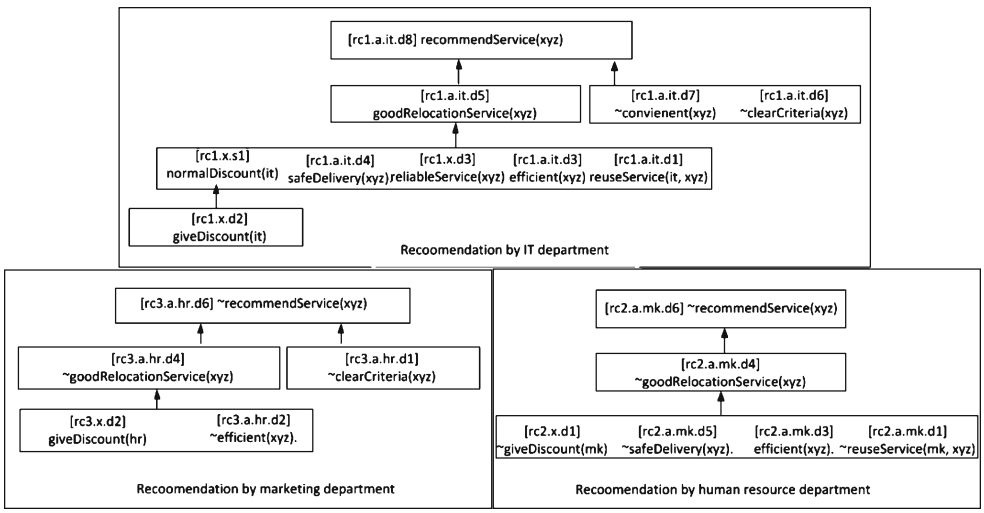


Fig. 7 Recommendation space

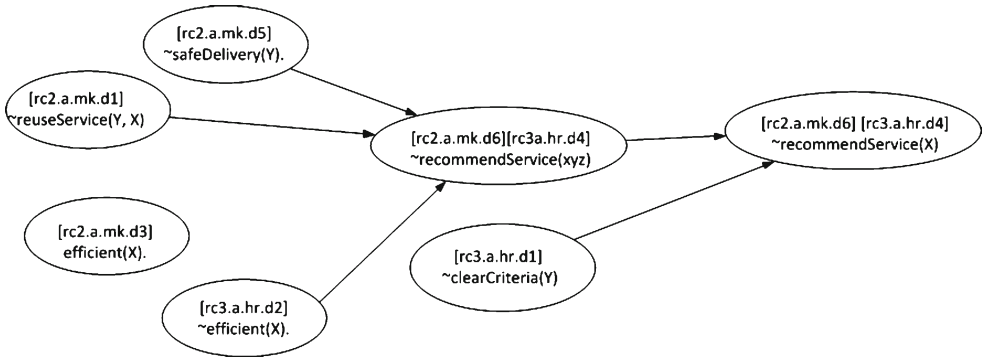


Fig. 8 Graphical representation of integration of rc2 and rc3

$\sim giveDiscount(hr)$, $\sim safeDelivery(xyz)$, $\sim efficent(xyz)$, $\sim reuseService(xyz)\}$.[6]

Definition 10 (Conflict operator) is a binary operator, such that

$$[rc1]\lambda^{\text{val}}_{(A,a)} \oslash [rc2]\lambda^{\text{val}}_{(B,a)} \quad (8)$$

is called a 'conflict operator'. The application of this operator to reasoning chains will return the set of arguments along with their counter-argument and undefeated or blocking dialectical trees.

Let us consider the recommendation space illustrated in Fig. 7, the application of conflict operator on reasoning chains rc2 and rc3 results in the following set of claims $\{efficient(xyz), \sim efficient(xyz)\}$

Definition 11 (Preference operator) is a binary operator such that

$$[a]giveDiscount(XYZ) > [b] \sim giveDiscount(XYZ) \quad (9)$$

is known as a 'preference operator'. The end user can define a preference relation explicitly for an argument and its counter-arguments.

Definition 12 (Knowledge integration) The result of a reasoning chain $\lambda_{(A,a)}$ supported by a chain of sub-arguments belonging to valued recommendation set produces an Integrated reasoning chain $\Theta_a = (\lambda_i, ..., \lambda_n)$ where $0 < i < n$. Mathematically, we define an integrated reasoning chain as follows:

$$\forall r, s \in valuated\ argument\ set\{if(s \xi r) then \Theta_j = \Theta_j \cup s \quad (10)$$

where ξ is used to represent the sub-argument relationship and $\Theta_j$is used to represent an integrated reasoning chain for result j. Taking into consideration the recommendation space depicted in Fig. 7, the integration of rc2 and rc3 results are shown in Fig. 8 considering conflict blocking flag is false.

[6]For simplicity we have not mentioned the identifers of the arguments.

7 Proposed conceptual framework

In this section, we elaborate in detail the proposed conceptual framework of Web@KIDSS for knowledge integration spanning across different Web-DSS. The proposed framework takes into consideration the reasoning chains (recommendations) published on the web or enterprise intranet for enabling decision-making based on enterprise-wise information integration. The task of Web@KIDSS is to integrate them into a consolidated reasoning chain depicting an enterprise-wise picture to the decision maker and facilitate the decision-making process. To achieve this task, the system user defines an integration scheme to perform a primary evaluation of a reasoning chain to determine whether it is suitable for integration with the rest of the reasoning chains. The proposed framework uses DeLP as knowledge representation and reasoning language with certain extensions (Janjua and Hussain 2011). Figure 9 depicts the proposed conceptual framework. The key components of our proposed framework are as follows:

7.1 Translation of the reasoning chains

The reasoning chains published on intranet by different Web-DSS in AIF compliant format are imported by the system user and system translates the imported reasoning chains in DeLP compliant reasoning chains. During this process, the translation of nodes and the relationship between nodes in an AIF formatted reasoning chain are translated to DeLP construct. The information nodes are translated as either premise of an argument or claim, whereas scheme nodes are used to build the types of arguments and relationship between arguments. For example, if there is an RA-node (defeasible or strict inference) the predecessor of scheme nodes will be the premise and successor of the RA-node and will be claim of the argument. Similarly, CA-nodes and PA-nodes are translated into counter-argument and defeat the relationship between arguments respectively. The blank-nodes are translated as sub-argument

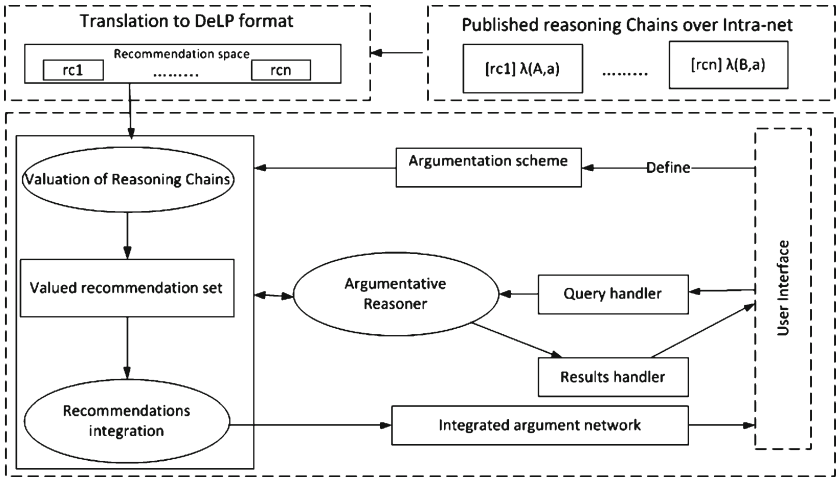


Fig. 9 Conceptual framework of Web@KIDSS

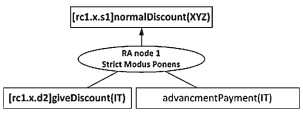
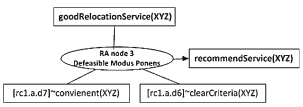
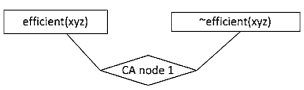
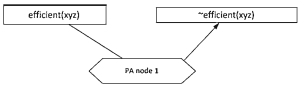
relationship between arguments. Some examples of the translation are depicted in Table 2 below.

7.2 Definition of integration scheme

Once the task of translating the reasoning chains has been accomplished, the next step is to define the integration scheme. The integration scheme, derived from the concept of argumentation scheme, corresponds to our daily life pattern of reasoning. To further explain it, let us consider the case study discussed in Section 3 where higher authorities has a set of recommendations and wants to select only those recommendations that satisfy certain specific criteria. For example, higher authorities specify criteria that the recommendation must be provided for relocation service provider XYZ. Therefore, only recommendations for XYZ are considered for the final decision-making process. The scope of application of an integration scheme ranges from valuation of reasoning chains and their integration during the decision-making process. In the proposed framework, DeLP language is used to create an integration scheme using the following steps:

1. Enter name of integration scheme.
2. Define set of premises.
3. Define set of critical questions. The critical questions are queries to be executed on a reasoning chain. The critical questions are further categorised as follows:
   (a) Set of assumptions
   (b) Set of exceptions

Table 2 Translation of AIF compliant reasoning chain in Web@KIDSS

Scheme Node	AIF representation	Translation
Strict inference		If the RA-Nodes use strict modus ponens, then all the incoming edges to RA-Node are considered as premises and the successor node is considered as claim of strict argument.
Defeasible inference		If the RA-Nodes use defeasible modus ponens, then all the incoming edges to RA-Node are considered as premises and the successor node is considered as claim of defeasible argument
CA-node		No translation, as proposed system has a built-in mechanism to identify conflicting arguments
PA-node		No translation, as proposed system has a built-in mechanism to identify priority among conflicting arguments.

4. Set conflict handling variant i.e. conflict blocking either true or false. The scope of conflict handling can be defined at valuation of reasoning chains or their integration or at both levels.
   (a) During valuation of a reasoning chain, if there exists any conflict between the a critical question and the premise, then in the case of conflict blocking variant true, the reasoning chain is not considered suitable for knowledge integration and vice versa.
   (b) During knowledge integration, if there exists a conflict between two arguments coming from different reasoning chains, then in the case of conflict blocking variant true, those arguments are not considered in the final decision-making process and vice versa.

Table 3 below depicts the definition of an integration scheme for knowledge integration.

7.3 Valued recommendation set

After defining the integration scheme, now the system applies the user-defined integration scheme to each reasoning chain. This process requires the following two steps:

1. Modelling of reasoning chains
2. Application of integration scheme to reasoning chains

7.3.1 Modelling of reasoning chains

The system first models the reasoning chain by identifying its basic elements as determined by Toulmin (2003). A reasoning chain is modelled as follows:

1. **Back-up evidence**: The initial working memory describing the current situation, from which the argumentative reasoner starts its derivation activity. In a reasoning chain, these nodes have no incoming edge (no predecessor nodes) and only an outer edge, or successor nodes, are considered as back-up evidence.
2. **Claim**: The result of the reasoning chain corresponds to claim.
3. **Warrant**: The support for the result of a reasoning chain is called a warrant. It is a set of arguments linked up to form a reasoning chain link as back-up evidence for a claim.

Such modelling of a reasoning chain has a significant relevance for correctly modelling a practical argumentation activity and helps to categorize the various ways by which arguments can be analysed and defeated and therefore the following strategies could have significant value as identified by Baroni et al. (1998). If conflict exists between a critical question and data, then the entire conclusion drawn from them is undermined. Similarly, it could help to point out flaws in the reasoning chain that relate data to the conclusion. Additionally, if conflict exists between claim and critical question, then the decision maker has to see the warrant and data in order to defeat the claim.

7.3.2 Application of integration scheme on reasoning chains

After the modelling of reasoning chains, the system applies the integration scheme defined by the system user to each and every reasoning chain. This involves executing all commands against the selected reasoning chain as depicted in Table 3. During this process, if any conflict exists either between data and premise, or conflict between critical question and warrant, then the system stores those results and depending upon conflict blocking variable value, the reasoning chain will be considered for the knowledge integration phase. The system also displays the results to the system user so that conflicts can be resolved if possible.

7.4 Knowledge integration

Once the valuation of reasoning chains has been completed, the next step is knowledge integration. This step involves integration of the diverse valued reasoning chains into a single consolidated reasoning chain to provide a complete picture to the decision maker to

Table 3 Integration scheme mapping to DeLP

	Integration scheme	DeLP construct
Scheme name	SupplierIntegrationScheme	
Premise	– The recommendations are against Relocation Service provider XYZ	– Execute(relocationService(xyz))
Critical questions	– The XYZ is good at formalising the clients' criteria	– Execute(clearCriteria(xyz))
Variant	– Conflict blocking is true for valuation of reasoning chains	– Conflict-blocking=true

support the decision-making process. This step comprises the following tasks:

1. Identification of conflicts among arguments belonging to different valued reasoning chains in a valued recommendation set.
2. Automated resolution of conflicts between arguments with the help of static and dynamic defeat. In case of blocking arguments, the system needs human intervention to resolve the conflict between them.
3. Construction of new arguments. If two arguments from a valued recommendation set have the same claim, then combine the premises of those arguments to produce a new argument.
4. Building up of reasoning chains and providing an interface to system user to making a final judgement as depicted in Fig. 10

7.5 Query the valuated set

The system also provides an interface to query the valuated reasoning chains.

Definition 13 (Query) A query 'q' , consists of a predicate, and can be executed on the argument set Args with the help of function executeQuery(q) $\in \mathcal{F}$ to check the support for the predicate in the recommendation space.

There are four possible answers to a query, as follows:

- If the answer is 'yes', then the result will be an undefeated dialectical tree. Mathematically, it is presented as follows:

$$\Sigma_U(\mathcal{A}, h) = executeQuery(q) \ldots\ldots\ldots \text{Equation (24)}$$

- If the answer is 'no', then the result will be a defeated dialectical tree. Mathematically, it is presented as follows:

$$\Sigma_D(\mathcal{A}, h) = executeQuery(q) \ldots\ldots\ldots \text{Equation (25)}$$

- If the answer is 'undecided', then the result will be a blocked dialectical tree. Mathematically, it is presented as follows:

$$\Sigma_B(\mathcal{A}, h) = executeQuery(q) \ldots\ldots\ldots \text{Equation (26)}$$

- Unknown, if the predicate in the query is not in the language of the program. Mathematically, it is presented as follows:

$$unknown = executeQuery(q) \ldots\ldots\ldots \text{Equation (27)}$$

7.6 User interface

The user interface is the graphical representation of a reasoning engine output for the end user. The user interface component will be responsible for representing the argumentation process and justifies conclusions to the user in the form of an inverted tree-like structure, and the user will be able to interact with and query the results.

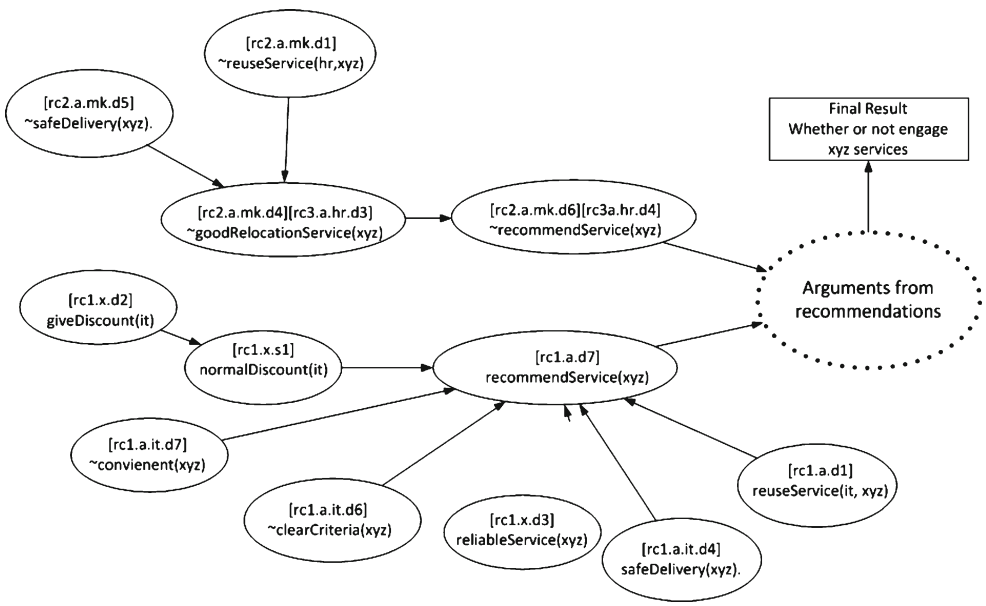


Fig. 10 Graphical representation of integrated knowledge for decision support

7.7 Algorithms for knowledge integration

In this section, we describe the working of our knowledge integration algorithms. Algorithm 1 (knowledge integration) invokes Algorithm 2 (valuation of a reasoning chain) and Algorithm 3 (combine reasoning chains). Algorithm 1 takes into account a set of AIF-compliant reasoning chains and sets their valuation flag to false. Then, the system applies the user-defined integration scheme to each of the reasoning chains one by one; this is known as the valuation of a reasoning chain. This step is carried out by invoking Algorithm 2. Algorithm 2 takes into account a single reasoning chain and a user-defined integration scheme such as *supplierIntegrationScheme* shown in Table 3. During valuation, all the queries generated through the integration scheme are executed on a reasoning chain. If the result of a query execution is false, this will establish a conflict between the reasoning chain content and the integration scheme content. The algorithm returns true if there is no conflict between the integration scheme and reasoning chain or the conflict blocking flag has a value of false. Otherwise, this algorithm will return false. The process of valuation applies to all the reasoning chains. After valuation, reasoning chains are ready for the next step: knowledge integration. All those reasoning chains with valuation flags true are considered in the knowledge integration phase. For knowledge integration, the system first integrates all those reasoning chains whose

Algorithm 1: Knowledge integration

Data: Recommendation space {rc1,rc2,rc3}
Result: Integrated arguments network

```
Array rc []= {rc1, rc2, rc3};
int i=0;
supplierIngtegrationScheme si;
System user initialize si;
foreach rc.length do
    rc[i].valued=false;
    i++;
end
foreach rchain in rc do
    boolean a = Valuation_of_reasoning_chain(rchain,si);
    if a is true then
        rc.valued = true;
    else
        rc.remove(rchain);
    end
end
i =0;
rc = combineReasoningChain(rc, sameResults);
displayResult(rc);
```

Algorithm 2: Valuation of a reasoning chain

Data: a reasoning chains(rc) and integration scheme(is)
Result: boolean

```
foreach premise in is do
    if result = execute(premise) on rc then
        if result= false then
            addconflict(premise, rc);
        end
    end
end
foreach cq in is do
    if result = execute(cq) on is then
        if result= false then
            addconflict(cq, rc);
        end
    end
end
if addconflict[] !=null or is.conflictProgation=true then
    return true;
else
    return false;
end
```

Algorithm 3: Combine reasoning chains

Data: Valued recommendation set, claimflag
Result: Array of combined reasoning chain

```
Array commonClaims [ ];
Array conflictingArguments [ ];
for i=0;i <rc.length(); i++ do
    for j=i;j <rc.length(); j++ do
        if flag==sameClaims then
            condition=rc[i].result ==rc[j].result
        else
            condition=rc[i].result !=rc[j].result
        end
        if condition then
            commonArgument = rc[i] ⊗ rc[j];
            activeArgumentSet= rc[i] ⊙ rc[j];
            conflictingArguments = rc[i] ⊘ rc[j];
            foreach c in commonClaims do
                argRc1 = getArgument(c, rc1);
                argRc2 = getArguments(c, rc2);
                r = argRc1 ⊞ argRc2;
                activeArgumentSet=r;
                i++;
            end
            foreach arg in conflictingArguments do
                argRc1 = getArgument(arg, rc1);
                argRc2 = getArguments(arg, rc2);
                if conflictblocking==true then

                    activeArgumentSet = argRc1;
                    activeArgumentSet = argRc2;
                    static or dynamic defeat(argRc1, argRc2);
                    userPreference=getPreferences(argRc1,argRc2);
                    if userPreference(argRc1,argRc2) then
                    preferenceSet = argRc1 > argRc2;
                end
            end
        end
        rc [i]=BuildupReasoningChain(activeArgumentSet);
    end
end
return rc;
```

Algorithm 4: Buildup a reasoning chain

Data: activeArgumentSet
Result: $\lambda_{(\mathcal{A},h)}$
Let S=subArgumentsOf($(\mathcal{A}, h)$);
if $S \neq \emptyset$ **then**
  foreach *argument* $(\mathcal{A}_i, h_i) \in S$ **do**
    if *noCounterArgument*$(\mathcal{A}_i, h_i)$ **then**
      BuildReasoningChain($(\mathcal{A}_i, h_i)$) ;
      Put $\lambda((\mathcal{A}_i, h_i))$as a immediate supporting argument of $(\mathcal{A}, h)$;
    end
  end
else
  $\lambda_{(\mathcal{A},h)} = (\mathcal{A}, h)$;
end

results are the same or support the same point of view. This task is performed by invoking Algorithm 3 with a set of valuated reasoning chains. Algorithm 3 first loops through a set of reasoning chains and compares the result of a reasoning chain with the result of the remaining reasoning chains; if the results match, then those reasoning chains are integrated. Three kinds of operators are used during this integration process. With the help of a focus operator ($\otimes$), the new arguments are constructed and then loaded into a valued recommendation set. With the help of a unique operator ($\odot$), unique arguments from both reasoning chains are loaded into an valued recommendation set. With the help of the conflict operator ($\oslash$) the conflicting arguments are taken into account for conflict resolution. If the conflict blocking flag for knowledge integration is false, then the system tries to resolve conflicts with the help of static or dynamic defeat. Otherwise, the system asks the end user to choose between the conflicting arguments. Finally, Algorithm 3 invokes Algorithm 4 (Buildup a reasoning chain) in order to establish a reasoning chain from the argument loaded in the valued recommendation set. The important thing to note here is that conflicts may exist in a valued recommendation set if the conflict blocking flag is true. The display function in Algorithm 1 displays the integrated reasoning chains to the user as depicted in Fig. 10. The integrated reasoning chains depict the different points of view supported by the set of arguments. The end user can make a decision based on the integrated information. The system saves the decision and makes it available for future reference.

8 Implementation and prototype development

This section provides the implementation details and working of Web@KIDSS to represent and reason over incomplete and conflicting recommendations coming from different departments. The development of the prototype system is carried out on a machine having an Apache Web server version 2.2.11, PHP version 5.3.0, PHP Tree Graph Ext library[7] with certain extensions to differentiate between fact and claim of a rule, strict and defeasible inference etc, MySql database version 5.1.36 and SWI-Prolog installed on it. After prototype development, the Web application is deployed on the DEBII server.

Figure 11 shows the interface provided to the system user to import the recommendations published in the form of reasoning chains in AIF format over the enterprise intranet. The user can download the file by entering the URL and name in the provided text fields and click the 'Download button'. The interface also shows the list of downloaded AIF compliant reasoning chain files and the user is able to either view or remove them from the Web@KIDSS. Once the user has finished downloading the recommendation files, s/he can then translate the downloaded files into DeLP format. Figure 11 depicts the interface where the user can select the files by clicking the check-boxes and submitting the selected files for translation by clicking the 'Translate AIF format files to DeLP' format button.

Once the user has finished importing the AIF files, s/he can then define an integration scheme for the valuation of reasoning chains. Figure 12 depicts an interface where a user can define premises that need to be matched, queries to be executed, and conflict blocking variant at valuation of a reasoning chain and knowledge integration levels. The end user also gives the integration scheme a name. Once the user has finished the integration scheme, s/he proceeds to the next step which is the valuation of reasoning chains by clicking the 'Next' link (shown in Fig. 12 above). Figure 13 depicts the interface where the user can select reasoning chains and click the 'Apply Integration Scheme' button to trigger the valuation process.

Once the user clicks the Apply Integration Scheme button, the Web@KIDSS applies all the premises that need to be matched and queries to be executed on reasoning chains. The user can view the outcome of the valuation process by clicking on the 'View' link against the valuated reasoning chain as depicted in Fig. 14. The text in red shows the conflict between the integration scheme and the contents of a reasoning chain.

After the valuation of reasoning chains, the next step is knowledge integration whereby all the reasoning chains are integrated to depict the overall problem. Figure 15 depicts the knowledge integration under two arguments: recommend the supplier XYZ as preferred

[7] http://download.getabest.com/new/php-tree-graph-ext-222943.html

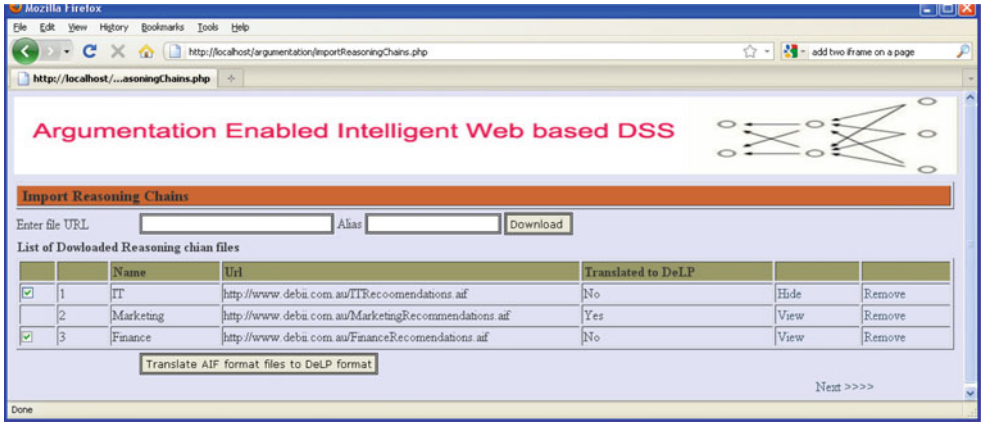


Fig. 11 Interface to import reasoning chains

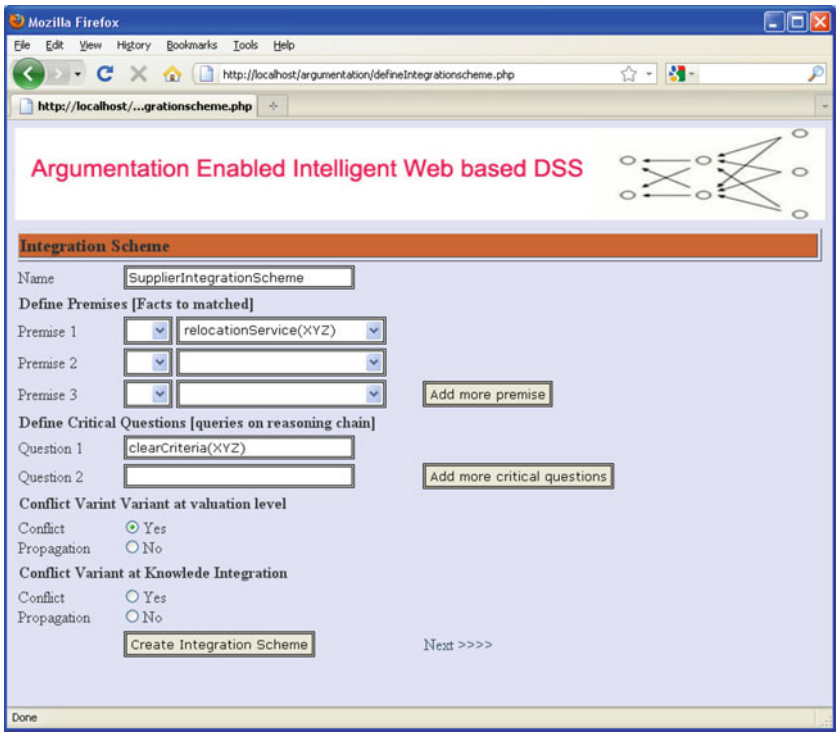


Fig. 12 Interface to define integration scheme

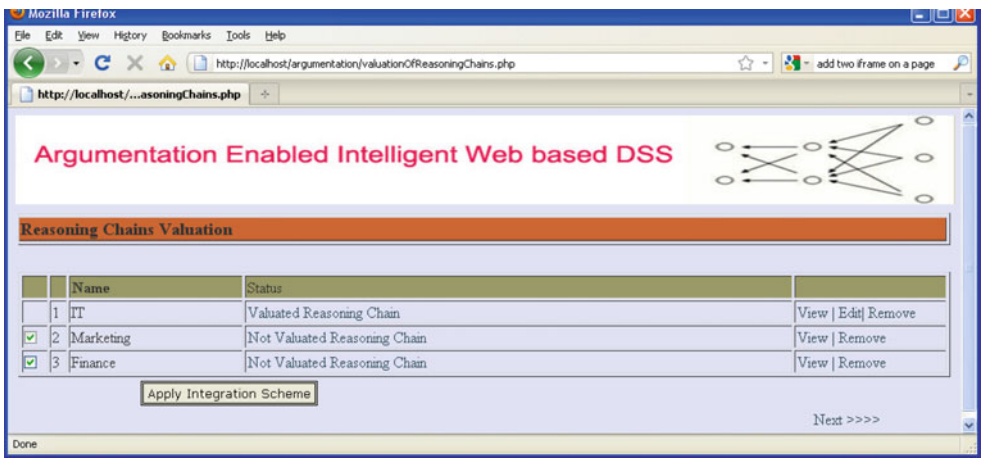


Fig. 13 Interface to select reasoning chains and apply integration scheme

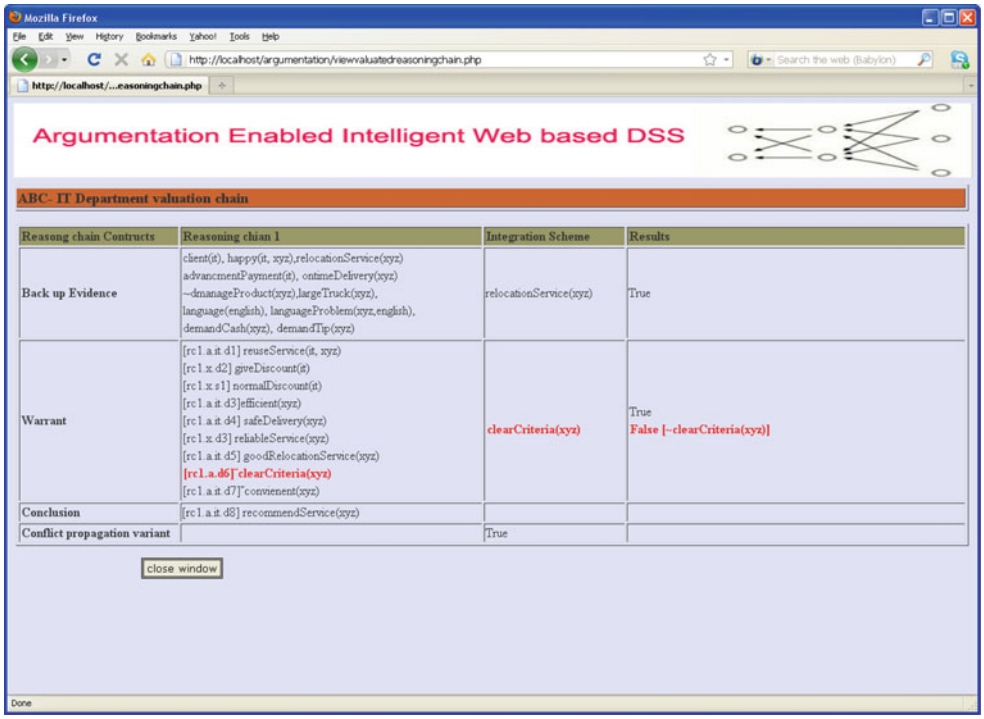


Fig. 14 Interface depicting the result of a reasoning chain valuation

client or not recommend it as preferred client. The final decision needs to be made by the end user who selects the result from the drop-down menu and clicks the 'Final Decision' button. This will save the system user's preference in the knowledge base.

9 Conclusion and future directions

In this article, we have presented a solution for enterprise-wide information and knowledge integration for intelligent decision making. We pointed out

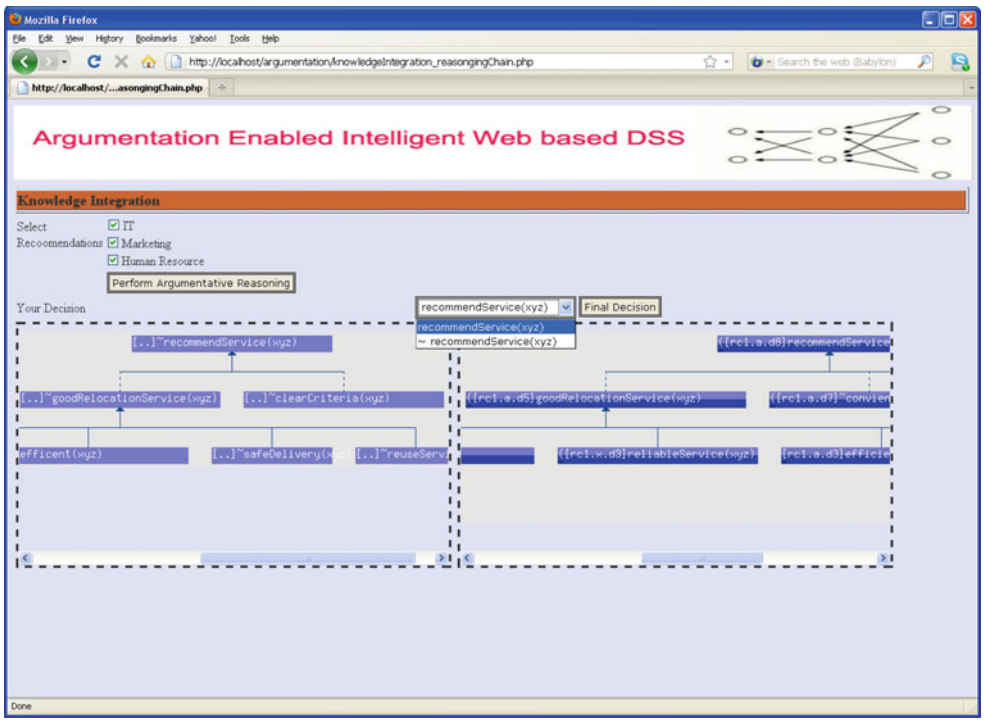


Fig. 15 Interface presenting integrated knowledge to facilitate final decision

that the semantic web, while it addresses the issues of syntactical and semantical heterogeneity of information to integrate and benefit the decision support system, nevertheless does not address the issue of incomplete and conflicting information integration. Several researchers have attempted to address this issue, but their efforts have provided a formalism to represent and handle only individual preferences in the form of priorities among the conflicting rules. However, DSS systems are subject to inconsistencies deriving from multiple sources and multiple users; therefore, it is not possible to define priorities in advance in order to resolve conflict among rules derived from multiple sources/users. This limitation of the current Web-DSS also prevents enterprises from being able to integrate knowledge.

In this article, we extend our previous work on Web@IDSS to make its results shareable in AIF format. We also provide formal syntax and semantics for Web@KIDSS. The Web@KIDSS is equipped with argumentative reasoning and an argumentation scheme for knowledge integration. Therefore, Web@KIDSS is capable of handling incomplete and contradictory knowledge in the form of reasoning chains published over the web or enterprise intranet by diverse Web-DSS or Web@IDSS. The major contributions of this article are as follows:

1. Extension to Web@IDSS with AIF reification resulting in sharing of results in the form of AIF-compliant reasoning chains.
2. Formalization of syntax and semantics for knowledge integration in Web@KIDSS.
3. A proposed conceptual framework for representing, reasoning and integrating incomplete and conflicting reasoning chains for knowledge integration in Web@KIDSS.
4. Design and development of algorithms for knowledge integration and their validation through prototype development.

Our future work will be primarily along the following lines:

1. Enterprise environments are becoming increasingly complex, competitive and dynamic. The business policies change dynamically and frequently to keep pace with the competitive nature of business environments. However the actual processes carried out in day-to-day business environments are not always in consonance with the new business policies (Wang et al. 2009). This situation is more profound in the case of managing dynamic processes where environment changes rapidly (Pesic and van der Aalst 2006). This demands for an enterprise business process modeling methodology that automatically builds models and executes task specific models in response to user queries (Ba et al. 1997). Such an approach should be flexible enough for e-Collaboration for business process modeling amongst different participants to address new challenges such as business process mergers. To address above mentioned challenge, we aim to design and develop policy-centric information system by extending the argumentation based intelligent decision making techniques proposed in this paper. We also aim at to introduce a graphical language to represent different process constructs and their linkages in a process model.
2. In the past decade or so, numerous machine learning methods have been used to automatically learn and recognize complex patterns and make intelligent decisions based on enterprise data. One of the common attributes of these machine leaning methods is that their working and functionality is constrained by the amount of input data. However, the scale of the enterprise data has increased mani-fold (leading to the concept of Big Data), thereby in many cases rendering the underlying machine learning algorithms either incapable of managing such large and ever increasing data, or too slow for decision making. In our further work we intend to enhance the current generation of machine learning techniques with argumentation formalisms described in this paper. In such cases, the arguments from experts are considered during mining of enterprise data.[8] Such work will lay down foundations for performing large-scale analytics on big data in an enterprise. Such an approach would make use of cloud platforms.

References

Alaranta, M., & Henningsson S. (2008). An approach to analyzing and planning post-merger is integration: insights from two field studies. *Information Systems Frontiers, 10*, 307–319.

Antoniou, G., & Bikakis, A. (2007). Dr-prolog: a system for defeasible reasoning with rules and ontologies on the semantic web. *IEEE Transactions on Knowledge and Data Engineering, 19*(2), 233.

Antoniou, G., Baldoni, M., Bonatti, P.A., Nejdl, W., Olmedilla, D. (2004). Rule-based policy specification. In Yu, T., Jajodia, S. (Eds.), *Secure data management in decentralized systems* (pp. 169–216, Vol. 33). US: Springer

Antoniou, G., Damasio, C.V., Grosof, B., Horrocks, I., Kifer, M., Maluszynski, J., Patel-Schneider, P.F. (2005).

[8] http://www.ailab.si/martin/abml/

Combining rules and ontologies: A survey. Tech. Rep. IST-2004-506779 REWERSE Deliverable I3-D3, Technical Report IST506779/Linköping/I3-D3/D/PU/a1. Linköping University

Ba, S., Lang, K.R., Whinston, A.B. (1997). Enterprise decision support using intranet technology. *Decision Support Systems, 20*(2), 99–134. doi:10.1016/S0167-9236(96)00068-1, http://www.sciencedirect.com/science/article/pii/S0167923696000681.

Baroni, P., Fogli, D., Guida, G. (1998). Modeling argumentation in practical reasoning: a conceptual analysis of argument life cycle. In *7th international conference on information processing and management of uncertainty in knowledge-based systems* (pp. 1790–1797). Paris.

Bassiliades, N., Antoniou, G., Vlahavas, I. (2004). Dr-device: A defeasible logic system for the semantic web. *Principles and practice of semantic web reasoning* (pp 134–148).

Benkö, T., Lukácsy, G., Fokt, A., Szeredi, P. (2003). Information integration through reasoning on meta-data. In *AI moves to IA: Workshop on artificial intelligence, information access, and mobile computing*.

Berners-Lee, T. (2000). Semantic web - xml2000, W3C. http://www.w3.org/2000/Talks/1206-xml2k-tbl/. Accessed 1 March 2012

Bex, F., Prakken, H., Reed, C. (2010). A formal analysis of the aif in terms of the aspic framework. In *3rd international conference on computational models of argument*.

Bhatt, M., Flahive, A., Wouters, C., Rahayu, W., Taniar, D. (2006). Move: a distributed framework for materialized ontology view extraction. *Algorithmica, 45*, 457–481. doi:10.1007/s00453-006-1221-2, http://portal.acm.org/citation.cfm?id=1165166.1165175.

Buccella, A., Cechich, A., Fillottrani, P. (2009). Ontology-driven geographic information integration: a survey of current approaches. *Computers & Geosciences 35*(4), 710–723. Geoscience Knowledge Representation in Cyberinfrastructure.

Carlsson, C, & Turban, E. (2002). Dss: directions for the next decade. *Decision Support Systems, 33*(2), 105–110.

Ceccaroni, L., Cortés, U., Sànchez-Marrè, M. (2004). Ontowedss: augmenting environmental decision-support systems with ontologies. *Environmental Modelling & Software, 19*(9), 785–797. Environmental Sciences and Artificial Intelligence.

Chesnevar, C., McGinnis, J., Modgil, S., Rahwan, I., Reed, C., Simari, G., South, M., Vreeswijk, G., Willmott, S. (2006a). Towards an argument interchange format. *The Knowledge Engineering Review, 21*(4), 293.

Chesnevar, C.I., Maguitman, A.G., Simari, G.R. (2006b). Argument-based critics and recommenders: a qualitative perspective on user support systems. *Data & Knowledge Engineering, 59*(2), 293.

Cheung, K., & Cheong, M.P. (2007). Intelligent on-line decision support tools for market operators. In *International conference on intelligent systems applications to power systems, 2007* (pp. 1–6).

Chua, W.W.K., & Goh, A.E.S. (2010). Techniques for discovering correspondences between ontologies. *International Journal of Web and Grid Services Archive, 6*(3), 213–243. doi:10.1504/IJWGS.2010.035090.

Fan, X., Toni, F., Hussain, A. (2010). Two-agent conflict resolution with assumption-based argumentation. In *Proceeding of the 2010 conference on computational models of argument: Proceedings of COMMA 2010* (pp. 231–242). Amsterdam, The Netherlands: IOS Press.

Flahive, A., Rahayu, W., Taniar, D., Apduhan, B. (2005). A distributed ontology framework in the semantic grid environment. In *19th international conference on Advanced Information Networking and Applications, 2005. AINA 2005* (pp. 193–196, Vol. 2). doi:10.1109/AINA.2005.19.

Flahive, A., Taniar, D., Rahayu, W., Apduhan, B.O. (2009). Ontology tailoring in the semantic grid. *Computer Standards & Interfaces, 31*(5), 870–885. doi:10.1016/j.csi.2008.03.016, http://www.sciencedirect.com/science/article/pii/S0920548908000330. Specification, Standards and Information Management for Distributed Systems.

Garcia, A.J., & Simari, G.R. (2004). Defeasible logic programming: an argumentative approach. *Theory and Practice of Logic Programming, 4*(1+2), 95–138.

Garcia-Crespo, A., Ruiz-Mezcua, B., Lopez-Cuadrado, J.L., Gonzalez-Carrasco, I. (2011). Semantic model for knowledge representation in e-business. *Knowledge-Based Systems, 24*(2), 282–296. doi:10.1016/j.knosys.2010.09.006.

Godden, D.J., & Walton, D. (2007) Advances in the theory of argumentation schemes and critical questions. *Informal Logic, 27*, 267–292.

Grosof BN, Gandhe MD, Finin TW (2002) Sweetjess: Translating damlruleml to jess. In *Proceedings of the international workshop on rule markup languages for business rules on the semantic web*.

Hurt, C.D. (1998). Nonmonotonic logic for use in information retrieval: an exploratory paper. *Information processing & management, 34*(1), 35.

Iyad Rahwan, C.R. (2009). *The argument interchange format, argumentation in artifical intelligence*. Springer.

Janjua, N.K., & Hussain, F.K. (2011). Web@idss : argumentation-enabled web-based idss for reasoning over incomplete and conflicting information. *Knowledge-Based Systems, 32*, 9–27. doi:10.1016/j.knosys.2011.09.009, http://www.sciencedirect.com/science/article/pii/S0950705111002103.

Kartha, N., & Novstrup, A. (2009). Ontology and rule based knowledge representation for situation management and decision support. In Mott, S., Buford, J.F., Jakobson, G., Mendenhall, J.M. (Eds.), *Intelligent sensing, situation management, impact assessment, and cyber-sensing*. SPIE.

Katie Atkinson, T.B.C. (2008). *Abstract argumentation scheme frameworks, artificial intelligence: Methodology, systems, and applications* (Vol. 5253/2008). Berlin/Heidelberg: Springer.

Kim, J., Kim, P., Chung, H. (2011). Ontology construction using online ontologies based on selection, mapping and merging. *International Journal of Web and Grid Services, 7*, 170–189.

Kontopoulos, E., Bassiliades, N., Antoniou, G. (2011). Visualizing semantic web proofs of defeasible logic in the DR-DEVICE system. *Knowledge-Based Systems, 24*(3), 406–419. doi:10.1016/j.knosys.2010.12.001.

Lee, T.B. (2003). The semantic web and challenges. W3C. http://www.w3.org/2003/Talks/01-sweb-tbl/. Accessed 1 March 2012.

Lee, T.B. (2005). Www 2005 keynote. W3C. http://www.w3.org/2005/Talks/0511-keynote-tbl/. Accessed 1 March 2012.

Lee, T.B. (2006). Artificial intelligence and the semantic web: Aaai 2006 keynote. W3C. http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html. Accessed 1 March 2012.

Letia, I., & Groza, A. (2008). A planning-based approach for enacting world wide argument web. In Badica, C., Mangioni, G., Carchiolo, V., Burdescu, D. (Eds.), *Intelligent distributed computing, systems and applications, studies in computational intelligence* (pp. 137–146, Vol. 162). Berlin/Heidelberg: Springer.

Liu, S., Duffy, A., Whitfield, R., Boyle, I. (2010). Integration of decision support systems to improve decision support performance. *Knowledge and Information Systems, 22*, 261–286.

Loui, R.P. (1998). Process and policy: Resource-bounded non-demonstrative reasoning. *Computational intelligence, 14*(1), 1.

March, S.T., & Hevner, A.R. (2007). Integrated decision support systems: a data warehousing perspective. *Decision Support Systems, 43*(3), 1031–1043. Integrated Decision Support.

Morge, M. (2008). The hedgehog and the fox: An argumentation-based decision support system. In: *Proceedings of the 4th international conference on Argumentation in Multi-agent Systems, ArgMAS'07* (pp. 114–131). Berlin, Heidelberg: Springer-Verlag.

Muthaiyah, S., & Kerschberg, L. (2007). Virtual organization security policies: an ontology-based integration approach. *Information Systems Frontiers, 9*, 505–514. doi:10.1007/s10796-007-9050-7.

Nguyen, H.Q., Taniar, D., Rahayu, J.W., Nguyen, K. (2011). Double-layered schema integration of heterogeneous xml sources. *Journal of Systems and Software, 84*(1), 63–76. doi:10.1016/j.jss.2010.07.055. Information Networking and Software Services.

Nicolicin-Georgescu, V., Benatier, V., Lehn, R., Briand, H. (2010). Ontology-based autonomic computing for decision support systems management: Shared ressources allocation between groups of data warehouses. In *2010 3rd international conference on Communication Theory, Reliability, and Quality of Service (CTRQ)* (pp. 233–236). doi:10.1109/CTRQ.2010.46.

Norta, A., & Eshuis, R. (2010). Specification and verification of harmonized business-process collaborations. *Information Systems Frontiers, 12*, 457–479. doi:10.1007/s10796-009-9164-1.

Noy, N.F. (2004). Semantic integration: a survey of ontology-based approaches. *SIGMOD Record, 33*, 65–70.

Osei-Bryson, K.M., & Ngwenyama, O. (2008). Decision models for information systems management. *Information Systems Frontiers, 10*, 277–279.

Palau, R.M., & Moens, M.F. (2009). Argumentation mining: the detection, classification and structure of arguments in text. In *ICAIL '09: Proceedings of the 12th international conference on artificial intelligence and law* (pp. 98–107). New York, NY: ACM.

Pesic, M., & van der Aalst, W. (2006). A declarative approach for flexible business processes management. In Eder, J., & Dustdar, S. (Eds.), *Business process management workshops, lecture notes in computer science* (Vol. 4103, pp 169–180). Berlin/Heidelberg: Springer. doi:10.1007/11837862_18.

Power, D.J. (2002). *Decision support systems: Concepts and resources for managers*. Greenwood Publishing Group.

Power, D.J., & Sharda, R. (2009). Decision support systems. In Nof, S.Y. (Ed.), *Springer handbook of automation* (pp. 1539–1548). Berlin/Heidelberg: Springer.

Rahwan, I., Zablith, F., Reed, C. (2007a). Towards large scale argumentation support on the semantic web. In *AAAI'07: Proceedings of the 22nd national conference on artificial intelligence* (pp. 1446–1451). AAAI Press.

Rahwan, I., Zablitha, F., Reed, C. (2007b). Laying the foundations for a world wide argument web. *Artificial intelligence, 171*(10–15), 897.

Salam, A. (2007). Design and implementation of semantic decision support system for supplier performance contract monitoring and execution: Integrating description logics, semantic web rules and service-oriented computing in the context of the extended enterprise. In *Americas conference on information systems*.

Seng, J.L., & Kong, I.L. (2009). A schema and ontology-aided intelligent information integration. *Expert Systems with Applications, 36*, 10,538–10,550.

Shim, J.P., Warkentin, M., Courtney, J.F., Power, D.J., Sharda, R., Carlsson, C. (2002). Past, present, and future of decision support technology. *Decision Support Systems, 33*(2), 111–126.

Silverman, B.G., Bachann, M., Al-Akharas, K. (2001). Implications of buyer decision theory for design of e-commerce websites. *International Journal of Human-Computer Studies, 55*(5), 815–844.

Suguri, H., Ahmad, H.F., Pasha, M., Khalid, N. (2008). *Foundation for autonomous semantic grid*. USA: Nova Science Publications.

Toni, F. (2007). E-business in argugrid. In Veit, D., & Altmann, J. (Eds.), Grid economics and business models, lecture notes in computer science (Vol. 4685, pp. 164–169). Berlin/Heidelberg: Springer.

Torroni, P., Gavanelli, M., Chesani, F. (2009). *Arguing on the semantic grid*. USA: Springer.

Toulmin, S.E. (2003). *The uses of argument*. Cambridge University Press.

Vahidov, R., Kersten, G.E. (2004). Decision station: situating decision support systems. *Decision Support Systems, 38*(2), 283–303.

Walton, D. (2009). *Argumentation in artificial intelligence, chap argumentation theory: A very short introduction* (pp. 1–24). Springer.

Wang, H.J., Zhao, J.L., Zhang, L.J. (2009). Policy-driven process mapping (pdpm): discovering process models from business policies. *Decision Support Systems, 48*(1), 267–281. doi:10.1016/j.dss.2009.08.006, http://www.sciencedirect.com/science/article/pii/S0167923609002012, Information Product Markets.

Xue, Y., Ghenniwa, H., Shen, W. (2009). Ontological view-driven semantic integration in collaborative networks. In Camarinha-Matos, L., Paraskakis, I., Afsarmanesh, H. (Eds.), *Leveraging knowledge for innovation in collaborative networks*, *IFIP advances in information and communication technology* (Vol. 307, pp. 311–318). Boston: Springer.

Yang, X., Bo, Z., Bei, Z. (2009). Research on semantic decision support system. In *WRI World congress on computer science and information engineering, 2009* (Vol. 5, pp. 687–691). doi:10.1109/CSIE.2009.364.

Zarefsky, D. (2009). *Argumentation: The study of effective reasoning* (2nd Edn., Vol. 2009). Northwestern University. URL: http://www.teach12.com/ttcx/CourseDescLong2.aspx?cid=4294.

Zhou, J., Yang, H., Wang, M., Zhang, R., Yue, T., Zhang, S., Mo, R. (2010). A survey of semantic enterprise information integration. In: *2010 3rd International Conference on Information Sciences and Interaction Sciences (ICIS)* (pp. 234–239).

Naeem Khalid Janjua is a PhD student at School of Information Systems, Curtin University, Perth, WA. He received his Master Degree in Information Technology from NUST School of Electrical Engineering and Computer Science, Islamabad, Pakistan. His areas of interests are Web-based decision support systems, Argumentation based applications, Semantic Web and Artificial Intelligence technologies.

Farookh Khadeer Hussain received the Bachelor of Technology degree in computer science and computer engineering; the M.S. degree in Information Yechnology from the La Trobe University, Melbourne, Australia; and the Ph.D. degree in Information Systems from Curtin University of Technology, Perth, Australia, in 2006. He is currently a Faculty member at School of Software, Faculty of Engineering and Information Technology, University of Technology, Sydney, Ultimo, NSW. His areas of active research are trust, reputation, trust ontologies, data modeling of

public and private trust data, semantic web technologies and industrial informatics. He works actively in the domain of making informed business decisions (business intelligence) through the use of trust and reputation technology. He is interested in the application of trust and reputation as a technology, as a business analysis and intelligence tool, and the applications of trust and reputation to various domains.

Omar Khadeer Hussain received his PhD degree in Computer Science in 2008 from Curtin University. Since 2008, he is currently a Research Fellow with the School of Information Systems at Curtin University. His research interests are in the area of Risk Assessment and Management, Trusted Computing, Informed Decision Support Systems. He is a member of the IEEE and the IEEE Computer Society.

Defeasible Reasoning based Argumentative Web-IDSS for Virtual Teams (VTs)

Naeem Khalid Janjua
Digital Ecosystems and Business Intelligence Institute
Curtin University of Technology
Perth, WA
Email: naeemkhalid.janjua@curtin.edu.au

Farookh Khadeer Hussain
Digital Ecosystems and Business Intelligence Institute
Curtin University of Technology
Perth, WA
Email: farookh.hussain@cbs.curtin.edu.au

***Abstract*—The Web-based intelligent decision support system (Web-IDSS) is pivotal for a Virtual Team (VT) to successfully execute business-related tasks. The current generation of Web-IDSS built on top of semantic web technologies for VTs lacks the capability to provide decision support when underlying information is incomplete and/or contradictory. In this article, we address this limitation of current Web-IDSS through defeasible logic based argumentation formalism. The proposed Web-IDSS uses a hybrid reasoning approach: forward chaining (data-driven) for the construction of arguments over incomplete information, and backward chaining (goal-driven) for conflict identification and resolution with explanation. The proposed Web-IDSS adheres to web standards and publishes the outcome of argumentative reasoning in Argument Interchange Format (AIF).**

I. Introduction and Related Work

Today, business environments are becoming more complex, competitive and dynamic, demanding that organizations be more flexible and responsive to environmental changes. As a result, organizations are establishing Information Technology as a primary enabler in order to adapt to an ever changing competitive environment. One of the outcomes of such efforts is the establishment of Virtual Team (VT) to accomplish one or more organizational tasks [1]. Technological support for these virtual teams for collaboration in distributed environments is now a topic of great interest.

The VT deals with two types of knowledge. The first is static knowledge composed of facts and ontologies which remain static over a period of time; this requires monotonic reasoning based upon open world assumptions to guarantee the correct propagation of truth. The second category, which needs special attention, is dynamic knowledge such as business polices, business contracts etc. that often change according to business needs and strategies, leading to conflicts among business polices. Such dynamic knowledge that is potentially incomplete and inconsistent needs non-monotonic or defeasible reasoning [2]. The term 'defeasible reasoning' was coined to mean 'convincing' although not rigorous reasoning as a concept introduced in the philosophy of law. Defeasible reasoning is a simple rule-based approach to perform reasoning about uncertain information where a rule supporting a conclusion may be negated or invalidated with the emergence of new information.

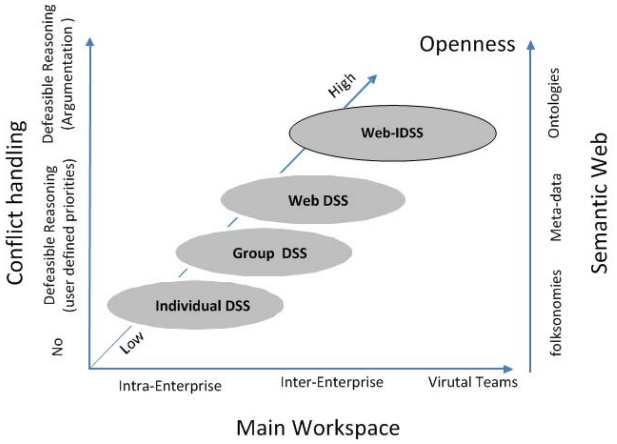


Figure 1. Trend towards defeasible logic based argumentative Web-IDSS for VTs

Business policies or contracts are also rule-based statements that are used by organizations to run their business activities. The importance of explicit, declarative and automatically executable business rules has been identified by several researchers [3, 4]. Attempts have been made to apply defeasible logic based approaches to represent and reason over dynamic knowledge that potentially is incomplete and conflicting as depicted in Table I. Although these attempts are very promising, they define explicit priorities among business rules at compile time, in order to resolve conflicts among business rules in advance. Whereas, VT is a source of defeasible knowledge as it is open by nature and subject to inconsistencies deriving from multiple sources; therefore, it is not possible to define priorities in advance among conflicting rules. Additionally, in current implementations, the use of these priorities is usually embedded in the derivation mechanism and competing rules are compared individually during the derivation process. In such formalisms, the derivation notion is bound to one single comparison criterion. In such a scenario, the explanation of the results is based on a single criterion only and fails to take into account the multiple factors important for decision-making in VT.

978-0-7695-4513-4/11 $26.00 © 2011 IEEE
DOI 10.1109/WI-IAT.2011.168

	Dr-Prolog [7]	Dr-Device [8]	Situated Courteous logic [9]
Language	Prolog	JESS	JESS
Logic	Defeasible logic	Defeasible logic	Situated Courteous logic
Semantic data	RDFS/OWL	RDF	DAML+OIL
Rules representation	RuleML	RuleML	RuleML
Incomplete knowledge representation	Yes	Yes	Yes
Conflict representation	Yes	Yes	Yes
Data-driven reasoning	No	Yes	Yes
Goal-driven reasoning	Yes	No	No
Conflict resolution	User defined priorities	User defined priorities	User defined priorities
Explanation	Textual	Textual	Textual
AIF compatibility	No	No	No

Table I
COMPARISON OF DEFEASIBLE LOGIC BASED IMPLEMENTATIONS.

In contrast, if we look at Artificial Intelligence research, the challenge of incomplete and conflicting knowledge representation and reasoning over it in software agents has been addressed using logic-based argumentation formalism, i.e. defeasible logic programming (DeLP) [5]. Argumentation formalisms are defeasible reasoning systems which work by considering the reasons that lead to a given conclusion (claim) through a piece of reasoning (the supporting arguments) and potential challenges (counter arguments) for accepting the conclusion [6]. Argumentation plays a pivotal role in identifying and organizing what can be justifiably concluded, and presenting it systematically to human users or merging it with the justified conclusions of other machines in the absence of complete or accurate information.

In this article, we identify the shortcomings of DeLP in the context of providing decision support to a VT and extend it with data-driven reasoning and make it interoperable with semantic web standards. We propose defeasible logic based argumentative Web-IDSS for a VT. The novelty of the system is its hybrid reasoning approach: forward chaining (data-driven) for the construction of arguments, and backward chaining (goal-driven) for the evaluation of conclusions. The proposed system provides graphical explanation to the members of a VT for better understanding and traceability of results. From the Web-IDSS perspective, such powerful support systems would be able to carry out reasoning on data across members of a VT. This is the potential area of growth and research in Intelligent Web-DSS as depicted in Figure 1.

II. DEFEASIBLE LOGIC PROGRAMMING

Defeasible logic programming (DeLP)[5] is a general-purpose defeasible argumentation formalism based on logic programming, intended to model inconsistent and potentially contradictory knowledge. A defeasible logic program has the form ψ= (Π, Δ), where Π and Δ stand for strict knowledge and defeasible knowledge, respectively. The set Π involves strict rules of the form $P \leftarrow Q_1 \ldots Q_n$ and facts (strict rules with empty body), and it is assumed to be non-contradictory (i.e., no complementary literals P and $\sim P$ can be inferred, where $\sim P$ denotes the contrary of P). The set $\triangle$ involves defeasible rules of the form $P \leftarrow\!\!\prec Q_1 \ldots Q_n$ which stand for $Q_1 \ldots Q_n$ provide a tentative reason to believe P". Rules in DeLP are defined in terms of literals. A literal is an atom A or the strict negation ($\sim A$) of an atom. Default negation (denoted not A) is also allowed in the body of defeasible rules. For more details about syntax and semantics of DeLP interested readers are referred to [5] .

DeLP uses the argumentation formalism for the treatment of contradictory information by identifying conflicting information in the knowledge base and applying the dialectical process to decide which information prevails leading to conclusion. Therefore in DeLP, in order to decide between competing conclusions, the arguments that support the conclusions are compared. Thus, the comparison criterion is independent of the derivation process, and could be replaced in a modular way. Although DeLP addresses the challenge of incomplete and conflicting knowledge representation and reasoning, it cannot be used for the development of Web-IDSS for VT because of following limitations:

1) DeLP uses backward chaining or goal-driven reasoning only, whereas most of the reasoning in Web-IDSS for VTs is primarily data-driven requiring forward chain reasoning.
2) There is no tool available for VTs to translate business rules defined in RuleML[1] format into DeLP rule format.
3) There is no tool that provides proof explanation by graphical representation of reasoning steps and conclusion generated by an argumentative reasoner for traversal by VT members.

III. PROPOSED SYSTEM ARCHITECTURE

In this section, we discuss the proposed system architecture in detail as depicted in Figure 1. Let us assume that a virtual team for the Olympic Games is comprised of the Olympic International Committee (OIC), Organising Committees for the Olympic Games (OCOG) and host city. The objective of the virtual team is to make important decisions about sports activities. These three organizations have their own particular goals and expectations which impact on the overall organisation of the sports events. Let us further assume that the current task of VT members is to decide "whether or not a scheduled match will be played in rainy conditions". To accomplish this task, each member of a VT provides his/her views in form of rules about the stated task, and with the help of our proposed system, they reach to a justifiable conclusion.

[1] http://ruleml.org/

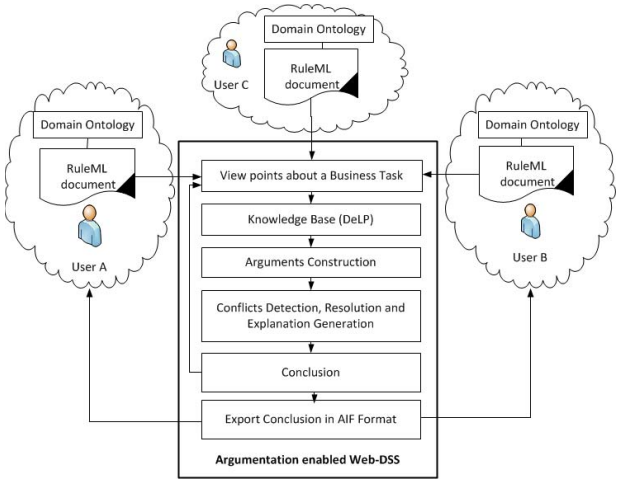


Figure 2. Proposed system architecture of argumentative Web-IDSS for a VT

The key components of the proposed system architecture are as follows:

A. View points against a business task

RuleML is an important step to provide a uniform format for the representation of business rules for a virtual team. RuleML supports different business rule types via the pre-defined $implies$ element and allows them to be named using the pre-defined oid element. RuleML syntax has been extended to express defeasible rules [10]. All members of a virtual team define their business rules in RuleML format and submit them through a graphical user interface of our proposed system as their point of view against a certain business task. The important thing to note here is the assumption that all members of a VT use the same domain ontology concepts to specify their business rules.

B. Knowledge base

A collection of facts is known as working memory and collection of rules is known as rule base. A working memory and a rule base are collectively known as knowledge base. In the proposed framework, the rule base comprises of strict rules and defeasible rules. Figure 3 depicts a knowledge base for a VT. In rule base a business rule takes the following form [rule identifier] [rule body] [type of rule] [head], where:

- rule identifier : is an identifier or name of the business rule;
- rule body : is a tuple of predicates. Each predicate is called as Premise;
- head: a predicate whose instances could be intuitively considered to be added to the working memory when the rule is activated during argument construction (defined later on);
- type of rule : system supports two types of rules: strict rules and defeasible rules. The strict rule is used to represent non-negotiable information represented by a solid arrow such as $\rightarrow$. Whereas, a defeasible rule is used to represent tentative, negotiable information and is represented by dotted arrow such as $\dashrightarrow$.

Rule 1	type(X,C)→C(X)	Class
Rule 2	subClassof(Sc, C), Sc(X)→C(X)	Subclass
Rule 3	objectProperty(X), domain(X, Y), range(X,Z) →X (Y, Z)	Object Property
Rule 4	objectProperty(X), X(Z, V), subProperty(X, Y)→Y(Z, X)	subProperty
Rule 5	dataProperty(X), domain(X, Y), range(X, Z) →X(Y, Z)	Data Property
Rule 6	dataproperty(X), X(Z, V), subProperty(X, Y)→Y(Z, X)	SubProperty

Table II
RULES FOR TRANSLATION OF OWL/RDF TO DELP FACTS

C. Translation of viewpoints and domain ontology

The proposed system parses and translates the business rules defined in RuleML notation to DeLP rule format and saves them in a knowledge base. Table II represents the ontology schema translation rules which also reside in the knowledge base to bring semantic interoperability among the business rules. Let us consider that 'ground' and 'stadium' are two ontological concepts from the domain ontology such that ground is subclass of stadium. The working memory contains 'ground(perth)'. On execution of Rule 2 depicted in Table II i.e., $ground(X) \rightarrow stadium(X)$ results in addition the of new fact i.e. stadium(perth), in the working memory.

D. Construction of arguments from view points

To construct arguments from a set of rules and working memory, forward chain reasoning is carried out which involves a rule engine matching the conditions of the rules in the rule base against the facts in working memory. For each match, a rule instance is created and put into the Active rule set. Once the matching phase is completed, the instances of all the rules in Active rule are fired. Firing the rule instance will result in the addition of a new fact to the working memory and the addition of an instance of rule known as argument in the Argument Set.

- A label is used to identify the argument.
- A conclusion is known as its claim.
- Ground predicate is known as the premise, supporting the claim.

A key issue to be noted here is that the new inferred facts may conflict with the existing knowledge base. The purpose is to retain conflicting information instead of eliminating it, in order to obtain a better insight when deciding on business strategies. Figure 4 represents the updated working memory and argument set.

E. Conflict detection, resolution and justifiable explanation

Once the argument construction process is complete, the process of conflict detection, resolution with explanation

Working memory

$$\left\{\begin{array}{l} ground(perth), rain(monday), \text{day(monday)}, \text{stadium(perth)}, \\ conditionOfLights(perth, good), \text{team}(aus\ ), \\ team(eng), matchSchedule(aus, eng, monday) \end{array}\right\}$$

Rule Base

$$\left\{\begin{array}{l} [hc1] ground(X), not\ rain(Y) \dashrightarrow \sim groundReady(X) \\ \\ [ocog1] stadium(X), drainage(X, good), rain(Y) \dashrightarrow \\ groundReady(X) \\ \\ [hc2] ground(X), conditionOfLights(X, bad) \dashrightarrow \\ \sim groundReady(X) \\ \\ [\text{oic1}] team(A), team(B), \text{day(Y)}, \text{matchSchedule}(A, B, Y) \dashrightarrow \\ printTickets(Y) \\ \\ [ocog2]\ not\ stadiumReady(X), \text{day(Y)}, printTicket(Y) \dashrightarrow \\ \sim rescheduleMatch(Y). \\ \\ [oic2]\ not\ rescheduleMatch(Y), day(Y) \rightarrow playMatch(Y) \end{array}\right\}$$

Figure 3. Knowledge base of argumentative Web-IDSS.

Updated Working Memory

$$\left\{\begin{array}{l} groundReady(perth), rain(monday), day(monday), \\ conditionOfLights(perth, good), team(aus), team(eng), \\ matchSchedule(aus, eng, monday), stadium(perth), \\ groundReady(perth), \\ \sim groundReady(perth), printTicket(monday), \\ rescheduleMatch(monday), playMatch(monday) \end{array}\right\}$$

Arguments Set

$$\left\{\begin{array}{l} [hc1] ground(perth), not\ rain(monday) \\ \dashrightarrow \sim groundReady(perth) \\ \\ [ocog1] ground(perth), drainage(perth, good), \\ rain(monday) \dashrightarrow groundReady(perth) \\ \\ [hc2] ground(perth), conditionOfLights(perth, bad) \\ \dashrightarrow \sim groundReady(perth) \\ \\ [\text{oic1}] team(aus), team(eng), day(monday), \\ matchSchedule(aus, eng, monday) \dashrightarrow \\ printTickets(monday) \\ \\ [ocog2]\ not\ groundReady(perth), \text{day(monday)}, \\ printTicket(monday) \dashrightarrow \\ \sim rescheduleMatch(monday). \\ \\ [oci2]\ not\ rescheduleMatch(monday), \\ day(monday) \rightarrow playMatch(monday) \end{array}\right\}$$

Figure 4. Updated knowledge base of argumentative Web-IDSS.

phase is initiated. For this purpose, we used a built-in mechanism of DeLP. We first identify the argument having a counter-argument. We define counter-argument as:

- Given an argument set, an argument 'r' counter-argues argument 's' if and only if claim of argument 'r' is inconsistent with claim of 's' or claim of argument 'r' is inconsistent with the premises of argument 's'.

In Figure 4, an argument 'hc1' is counter-argument to argument 'cocg1'. The system forwards the claim of the argument to the DeLP engine to construct its dialectical trees or justification trees as depicted in Figure 5 (Left) where graphically an argument is represented as $(hc1, \sim groundReady(perth))$ where hc1 is argument identifier and $\sim groundReady(perth)$ is claim of argument. Once the dialectical tree construction is complete, the nodes in dialectical trees are marked as either defeated or un-defeated as shown in Figure 5 (Right) and the DeLP engine return the results to the system. If the marked dialectical tree is defeated, then the counter-argument has priority over the argument and vice versa. Similarly, the system resolves the remaining conflicts between arguments and their counter-arguments by passing the argument's claim to DeLP engine to acquire its support from the knowledge base. In case, the marked dialectical tree of both, the argument and its counter-argument are undefeated, then those arguments are considered as blocking arguments and the system needs human intervention to resolve the conflict between them.

F. Conclusion

The last step of an argumentation process is the construction of a conclusion in the form of a reasoning chains. We define a supporting argument or sub-argument as:

- Given an argument set, an argument 'r' is a sub-argument of argument 's' if and only if claim of argument 'r' belongs to premise of argument 's'.

In Figure 4 an argument 'oic1' is supporting argument to argument 'ocog2'.

During this process, all sub-arguments with undefeated dialectical trees are linked together as a reasoning chain. This process will continue until all possible arguments are linked up into a reasoning chain. The top argument i.e. conclusion, of the reasoning chain is called the 'result' of the reasoning chain, and the chain of arguments supporting the top argument are called to support the conclusion as depicted in Figure 7. The reasoning chain is always consistent (i.e., there is no contradiction in the result and support for the result). Therefore, for example, $groundReady(perth)$ and $\sim groundReady(perth)$ will not belong to one reasoning chain, but each one of them can belong to different reasoning chains and those reasoning chains represent alternative paths or choices.

G. Publish conclusion

The Argument Interchange Format (AIF) [11] is an international effort to develop a representational mechanism for exchanging argument resources between research groups, tools, and domains using a semantically rich language. The system make use of Argument Interchange Format (AIF) to export the output of the argumentation process i.e. reasoning chains, over the web to ensure its interoperability with other argumentative systems. For annotation of a reasoning chain, we developed 'reasoning chain ontology' on top of the ArgDF ontology[2] and serialized the AIF compliant reasoning chain in RDF/XML format.

[2]http://www.argdf.org/source/ArgDF Protege Ontology.zip

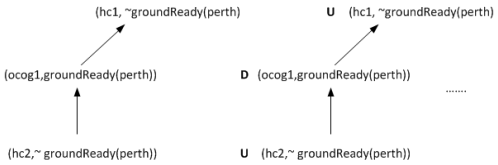


Figure 5. Dialectical tree (Left) and marked dialectical tree(Right)

H. Query the knowledge base

The members of a VT can query the knowledge base. The query consists of a predicate, and can be executed on the argument set to check the support for the predicate in the argument set. There are four possible answers to a query. If the answer is 'yes', then the result will be an undefeated dialectical tree. If the answer is 'no', then the result will be a defeated dialectical tree. If the answer is 'undecided', then the result will be a blocked dialectical tree. If unknown, then the predicate in the query is not in the language of the program.

IV. Prototype Development and Conclusion

The prototype development of the system is carried on a machine having Apache Web server version 2.2.11, PHP version 5.3.0, PHP Tree Graph Ext library[3] , MYSQL database version 5.1.36 and SWI-Prolog installed on it. We extended the PHP Tree Graph Ext library with certain extensions to differentiate between fact and claim of a rule, strict and defeasible inference etc,. The proposed system provides an interface to members of a VT to submit their rules against a business task defined in RuleML format. After translation of rules, the proposed system performs argumentative reasoning as discussed in Section III.

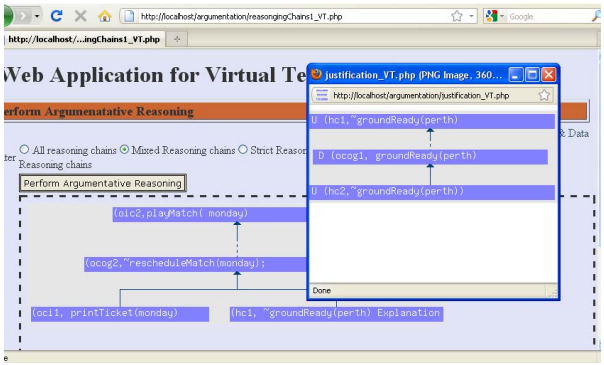


Figure 7. Graphical representation of a reasoning chain

Figure 7 depicts results of argumentative reasoning in the form of a graphical reasoning chain and a small pop window which shows justification for an argument($hc1, \sim groundReady(perth)$). We conclude here, by saying that, applications built on top of the proposed work, will provide more practical, understandable results to members of a VT for intelligent decision making.

[3]http://download.getabest.com/new/php-tree-graph-ext-222943.html

References

[1] A. Powell, G. Piccoli, and B. Ives, "Virtual teams: a review of current literature and directions for future research," *SIGMIS Database*, vol. 35, pp. 6–36, February 2004.

[2] N. K. Janjua and F. K. Hussain, "Development of a logic layer in the semantic web: Research issues," *Semantics, Knowledge and Grid, International Conference*, pp. 367–370, 2010.

[3] G. Antoniou and M. Arief, "Executable declarative business rules and their use in electronic commerce," in *Proceedings of the 2002 ACM Symposium on Applied Computing*, SAC '02, (New York, NY, USA), pp. 6–10, ACM, 2002.

[4] H. Boley, "Are your rules online? four web rule essentials," in *Advances in Rule Interchange and Applications* (A. Paschke and Y. Biletskiy, eds.), vol. 4824 of *Lecture Notes in Computer Science*, pp. 7–24, Springer Berlin / Heidelberg, 2007.

[5] A. J. Garcia and G. R. Simari, "Defeasible logic programming: an argumentative approach," *Theory and Practice of Logic Programming*, vol. 4, no. 1+2, pp. 95–138, 2004.

[6] J. Dix, S. Parsons, H. Prakken, and G. Simari, "Research challenges for argumentation," *Computer Science - Research and Development*, vol. 23, p. 8, Mar 2009.

[7] G. Antoniou and A. Bikakis, "Dr-prolog: A system for defeasible reasoning with rules and ontologies on the semantic web," *IEEE transactions on knowledge and data engineering*, vol. 19, no. 2, p. 233, 2007.

[8] E. Kontopoulos, N. Bassiliades, and G. Antoniou, "Deploying defeasible logic rule bases for the semantic web," *Data & Knowledge Engineering*, pp. 116–146, 2008.

[9] B. N. Grosof, M. D. Gandhe, and T. W. Finin, "Sweetjess: translating damlruleml to jess," in *Proceedings of the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web*, 2002.

[10] N. Bassiliades, G. Antoniou, and I. Vlahavas, "Dr-device: A defeasible logic system for the semantic web," *Principles and Practice of Semantic Web Reasoning*, pp. 134–148, 2004.

[11] C. Chesnevar, J. McGinnis, S. Modgil, I. Rahwan, C. Reed, G. Simari, M. South, G. Vreeswijk, and S. Willmott, "Towards an argument interchange format," *The Knowledge Engineering Review*, vol. 21, no. 4, p. 293, 2006.