

Association for Information Systems AIS Electronic Library (AISeL)

AMCIS 2007 Proceedings

Americas Conference on Information Systems
(AMCIS)

December 2007

Logic Agents and Conceptual Interoperability through Semantic Web Technologies

Tobias Lehmann
Universität der Bundeswehr

Andreas Karcher
Universität der Bundeswehr

Follow this and additional works at: <http://aisel.aisnet.org/amcis2007>

Recommended Citation

Lehmann, Tobias and Karcher, Andreas, "Logic Agents and Conceptual Interoperability through Semantic Web Technologies" (2007).
AMCIS 2007 Proceedings. 151.
<http://aisel.aisnet.org/amcis2007/151>

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2007 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

Logic Agents and Conceptual Interoperability through Semantic Web Technologies

Tobias Lehmann and Andreas Karcher
Department of Informatics, Universität der Bundeswehr München
85579 Neubiberg, Bavaria, Germany
Email: {tobias.lehmann|andreas.karcher}@unibw.de

Abstract. *This article describes one possible option on how the flexible and open world of Semantic Web Technologies such as the Resource Description Framework, the Web Ontology Language (OWL) or other languages compatible to the Jena-Model can be integrated with logic based agents implementing the so called Situation Calculus. By integrating both systems tightly all systems' features are technically made mutually available. Amongst others these are the web-based Modularity provided by OWL as well as similarity-recognition by Alignment-tools and Mapping-tools. Especially – but not exclusively – the latter methodology can bridge differences in interpretation of distributed Ontologies and herewith lead to a higher Level of Conceptual Interoperability. The paper outlines how Semantic Web Technologies can support Decision Makers within the EBO by Description Logic and Alignment along some examples. After this, synergetic effects from the mentioned integration are mentioned. Finally the technical integration based on Second Order Logic structures of complex Inferences represented by the Situation Calculus and Ontologies is described.*

Keywords: OWL, Ontologies, Logic Agents, Decision Support, Effects Based Approach to Operations.

1 Application Context: the Effects Based Approach to Operations

After the Warsaw Pact broke-down, western military forces faced a new class of threat such as international terrorism, arms-proliferation or asymmetric warfare. Therefore planning of operational and strategic missions shifts from an attrition-based towards an effects-based paradigm (Smith 2003). This change towards the Effects Based Approach to Operations (EBAO) opens up a broader and more far-sighted perspective for those involved. It strongly focuses on interagency-dependencies between actions taken and effects generated. By emphasizing the network nature of effects, complexity grows beyond human manageability (Klein 1998). In order to support a planner in coping with large amount of more or less structured information, technologies for integration of heterogeneous data quickly reach their limits (Wache, 2003). To support decision makers during their planning process, not only effective information-linking is needed but furthermore inferences using First Order Logic can provide helpful assistance. Compared to ontology-technology First Order Logic systems have a drawback in modeling features, interoperability and automated classification. The Web Ontology Language (OWL) is one representative of Description Logic (DL), and a W3C (World Wide Web Consortium) Standard since 2004. It provides mechanisms to build (combined with Classification-tools, so called Reasoners) implicit and flexible class structures (McGuinness and Harmelen 2004). In addition to this it natively supports distributed Knowledge Bases and can be used as a basis for Alignment-Processes between different models. Usage of Ontologies therefore means taking a step forward towards semantic comparison and a higher level of conceptual interoperability as described in Tolk (2006).

As a language for Knowledge Representation, OWL does not provide mechanisms for inference at instance level to support Planners but at structural level only (McGuinness and Harmelen 2004). If inference on instances is needed for Knowledge Processing, then an additional mechanism has to be integrated into the OWL-Knowledge Representation System. The article at hand describes one option to generically

integrate OWL- Knowledgebases with First Order Logic inferences (represented here by the Situation Calculus) and shows how instance inferences can benefit from technologies associated to Ontologies. This paper is not meant to analyze the applicability of Golog to certain problem-classes but to use it as a vehicle to describe synergetic integration of First Order Logic and Semantic Web Technologies.

1.1 The challenge of distributed modeling

As depicted in Figure 1 current Knowledge Bases (in the context of military Knowledge Representation) are created and maintained by a large number of distributed Analysts, who focus on specific categories such as politics, military, economic, social, information, and infrastructure. These so called PMESII categories lead to a separation of concerns which harbors the risk of redundancy that can easily turn into inconsistency. Although the endeavor of Knowledge Representation is international, especially in Europe national interests often protect own built systems at the expense of system interoperability, integration and common platforms.

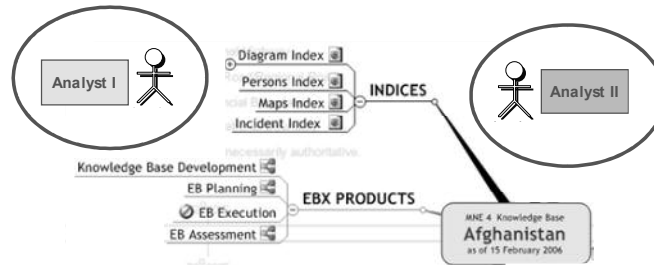


Figure 1. Distributed Modeling

Additionally, the less formalized explicit knowledge is (such as text files, Word documents, Mind Maps etc.) the more costly it becomes to detect overlapping. Intention of this paper is to show means for integration of formalized heterogeneous Ontologies, how their projection into other logic-based systems and to show, formalized Ontologies' capacity for Decision Support within distributed Knowledge Representation by Description Logic and Alignment.

1.2 A brief introduction to the Situation Calculus

The Situation Calculus is an abstract algorithm which tries to accomplish defined objectives by combining formalized actions and thereby converting one calculable situation into another (McCarthy and Hayes 1969). Courses of actions can be inferred by defining actions containing pre- and post- conditions and combining them sequentially (Levesque et al. 1997). These courses – if inferable – then lead to states which are described by objectives. This combinatory way of solving problems is, without further modifications, limited to low-to-medium complexity scenarios containing mostly certain and complete knowledge (Thielscher 1997). A interpreter called Golog can be considered a Prolog-based implementation of the Situation Calculus (Schiffel and Thielscher 2005).

Compared to classical First-Order-Logic, Golog is enriched by control structures (*if & while macros*) which reduce complexity when searching for courses of actions (McCarthy and Hayes 1969). Golog makes use of so called *primitive actions* which can be assembled to complex actions (such as: to drive a car means that one enters it, starts the engine and uses different paddles and a transmission). The question whether an action is applicable is defined within a KB – in this case an action *holds* within a situation. After the application of an action its consequences are formalized as so called *poss*-predicates, which actively change the inferred KB and therefore create new situations. Its resolving algorithm is strictly mechanically though and foresees no option for high-level inference-strategies.

The Situation Calculus could usefully be applied to scenarios whereby the planner knows little of the problem domain, but has knowledge of the appropriate actions and processes (Brachman and Levesque 2004).

1.3 The Web Ontology Language OWL

OWL is a flexible mechanism to define classes and individuals and describe the relations between them. It is platform-independent and extends the Resource Description Framework (RDF) (Klyne and Carroll 2004) semantics and serialisations (McGuinness and Harmelen 2004). An OWL-ontology can natively import other OWL-Ontologies or concepts contained in these by using URIs.

Therefore the reuse (see e.g. <http://swoogle.umbc.edu/>) of Ontologies is possible and desired. OWL Ontologies' class structures are defined by Description Logic statements. These statements express Set-Theory based relations between classes and qualify and quantify properties (Smith et al. 2004). One downside of OWL for classical inferences is the almost complete negligence of individuals. Although it allows the definition of instances, it exclusively focuses on class-structure-reasoning and requires external applications to give answers to questions on instance-inferences.

By the mechanisms mentioned above, OWL can be considered means of integrative Knowledge Representation. By associated Alignment tools similar structures in different models can be detected semi-automatically, and then be aligned and mapped, which leads to a more

dynamic integration of systems within a common context. The applicability and validity of alignment in a particular domain strongly depends on its language. The more static and the less ambiguous a domain language is, the better machine-formalizable it is a consistent way. In order to bind a Situation Calculus implementation to OWL we chose a representative of First Order Logic which deals with individual reasoning named SWI-Prolog. The following paragraph outlines how these two systems can be combined consistently and synergetic in order to give comprehensive decision support for EBAO planners.

2 Decision Support for the EBAO

This section describes what Support mechanisms Ontologies and its associated technologies can provide for distributed modeling and the Effects Based Approach to Operations in particular to ease semantic (not primarily technical) problems. Firstly some Ontology features are outlined. Secondly prerequisites for the application of complex First Order Logic inferences – represented by the Situation Calculus here – are elaborated. Thirdly and finally some positive side effects for of the prerequisites are described, which have been detected by now.

2.1 Description Logic as a base for Situational Understanding and Situational Awareness

As described earlier, when a KB for a Focus Area (an Area, of interest for current or future missions such as Iraq or Afghanistan) is prepared or maintained, distributed Ontologies can be used in different agencies and nations. As a typical result for the plurality of a KB, Figure 2 shows four different Analysts interpreting an identical thing from four different standpoints. The same person within Afghanistan is understood as a dangerous Warlord by the Military Analyst (I), as Anti-Governmental force by the Political Analyst (II), as a Merchant by the Economy Analyst (III) and as a Powerbroker by the Social Analyst (IV).

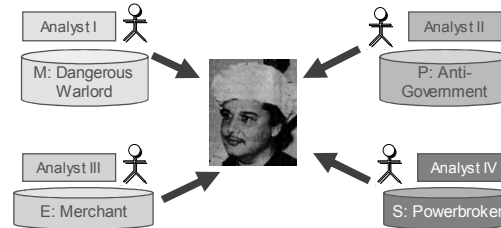


Figure 2. Different perspectives in modeling

But all have in common, that they model the identical person. In classical Knowledge Bases such as used in former Multi National Experiments of NATO, this Person would most likely have occurred four times as defined concepts, whereas the Description Logic would allow the inferred classification according to the descriptions from different analysts. Without going into too specific details on the Knowledge Base used for this paper, if each analyst contributed not an own instance but a description for this person, consistency would be preserved and the classification depending on these would lead to the insights the Analysts needed from Figure 2. As an example for Description Logics (shown in Figure 3) the S-Analyst could define, that *Person_M*, as he will be called furtheron, is a Person (a trivial statement, common in all systems supporting inheritance functionality). The E-Analyst could contribute, that *Person_M* conducts an activity, namely drug trade. And the M-Analyst would describe *Person_M* as a Person leading a big militia (that is a specialization of a simple militia).

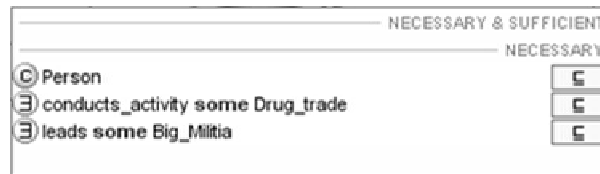


Figure 3. Example Description Logic statements

From these statements – and some other rules within the KB defined in (Lehmann 2007), the statements depicted in Figure 2 are inferable. This is shown in Figure 4. All arcs used in this figure represent inheritance relations, all ellipses represent classes. *Person_M* is an Anti-Government concept, because drug trade is considered an illegal activity and all concept conduction illegal activities are Anti-Government concepts. He is a Trader, because a Trader is every concept that conducts an activity that is of class Trade (which is given for drug trade). And because he leads a big Militia he is not only a simple warlord (which would be everybody, leading a Militia) but a dangerous Warlord too. All Leaders are Powerbrokers - that is why the S-Analyst would see the *Person_M* as such.

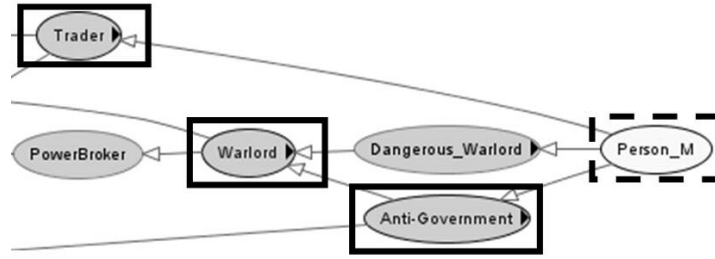


Figure 4. An inferred DL-structure

This simple example shows how implicit relations at structure level can be expressed by Description Logic and inferred by so called Reasoner (Sattler 2005). On the other hand DL also allows classification of currently “unknown” concepts. That is a Concept “without a class” could be described and the Reasoner would determine all its inferable super classes. This enables queries such as “I have a concept here, from which I know, that I conducts activities of trade”, with results like then this concept is a Person, a Trader, and an *Anti-Government* element.

But Description Logic can provide more implicit information, when it comes to properties. A further (implicit) classification for the separation of concerns for PMESII-Analysts that makes use of property inheritance is described in the following. Each property inherits from a general property called *general_Prop*. Classification as per the PMESII-categories can then be done by statements such as that every concept that is related to some Militia or to some Forces via a *general_Prop* is a Military concept. Therefore all warlords (who are implicitly defined as well) become Military concepts.

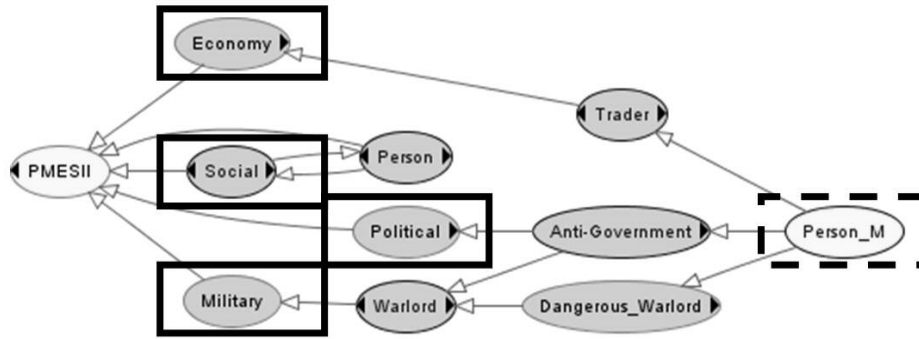


Figure 5. Implicit PMESII- classification using property inheritance

Results for these definitions could also be imaginable that classes are subclasses for PMESII-concepts. As done so in case of the Social concepts that include all Persons within the KB (heavily simplified). It is to mention here, the overlapping of categories is wanted in order to show areas where coordination of several agencies may be required. Each Anti-Government for example concept is both Military concept and Political concept.

2.2 Objectives and actions: concepts for the Situation Calculus

This section describes domain unspecific concepts used in the context of the logical inferences. The definition of all following concepts is done in OWL. By taking Situation Calculus-specific requirements into account at structural level, all instances can be transformed as described in 3.1 and used by an interpreter. Due to the distributed nature and other features (as those in 2.1) of Ontologies and their related applications interoperability can be carried into the Logic Language. They can be applied in different domains likewise and their functions and relations are as follows:

Pointer (Helper Class)

A Pointer carries information about a set of classes, a set of individuals and one property (as shown in Figure 5).

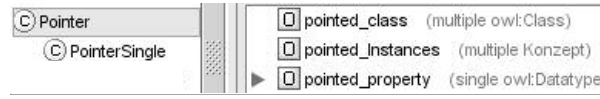


Figure 6. Pointer Class

A Pointer itself is without any functionality in OWL but has to be interpreted in applications using the OWL KB. In this case Prolog resolves a Pointer instance to relations (Subject, Predicate, Object) containing the property's values. These relations are restricted to instances listed explicitly and all instances of the given classes including their subclasses' instances, where all inferred instances occur as subjects within the relation. By addressing both classes and individuals the pointer class bridges structure and instance level. This leads to – as depicted in Figure 7 to a Focus Area independent definition A Pointer in one Focus Area can have very different instance and subclasses in another.

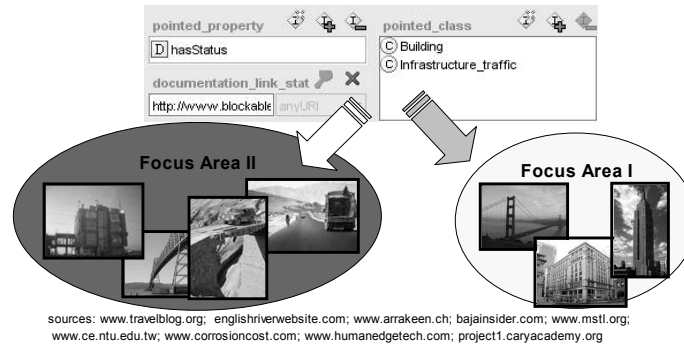


Figure 7. Example for dynamic addressing

While “Focus Area I” could be located within the US, “Focus Area II” could be located in Afghanistan and contain other elements of infrastructure blockable by certain means. But both play a similar role for transportation, traffic or accessibility of certain regions.

Objectives

Objectives define end-states for determined sets of concepts. This end-state can be described by objects or XML Schema data-type values (Fallside and Walmsley 2004). In order to achieve higher flexibility and a wider range of objectives not depending on concrete individuals the helper class Pointer is used to address the concepts targeted by an objective. Keeping in mind that class-relation – in sense of subclass and superclass – are to be inferred from DL expressions, these pointers are a flexible collection of classes and individuals.

Actions

Actions describe value-changes within a KB. If applied, Actions can change references or values of data-type properties. The most striking difference between an Effect within the EBAO and an Action is their granularity – an Effect is a result or conclusion of changes within systems after all Actions have been taken, whereas an action is a single change of a set of objects. In addition to this actions' ranges can be limited to a set of classes – an analogy to an OOP's polymorphic method's signature. The Actions' structure is defined as follows:

An Action's applicability to concepts is restricted to properties of certain sets of individuals. These sets of individuals are described by a set of Pointers (as shown in Figure 7).

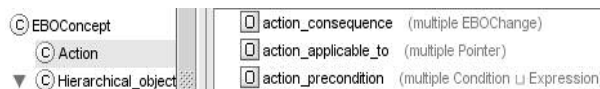


Figure 8. Actions

In order to quantify the change committed by an action another helper class is used (which shall be described briefly here only). The *EBOChange* class defines in which way concepts are modified by an action. It differentiates between relative and absolute change and contains a formula to calculate a modification. Finally Actions carry information about under which circumstances they can be executed. A

Condition in this case is a set-comparison (using Existential- or Universal Quantifiers) of two Pointer or formula (including constant values and Pointers).

2.3 Defining Situation Calculus actions

The transformation's objective mentioned above is to define primitive Actions within the OWL KB and automatically transform them into a Golog system. Figure 9 depicts the structure of a Golog Action named 'block' filled with example data. It shows that only instances of class infrastructure can be blocked and that the instance must not be blocked already.

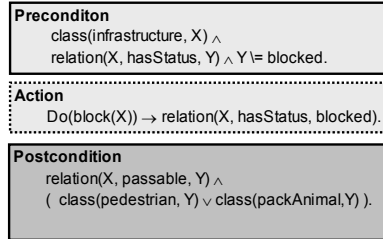


Figure 9. Schema of a Golog action

If this precondition (*hold*- aspect mentioned in 1.2) is fulfilled then the status of the infrastructure is set to 'blocked' which means (formalized in postcondition) that the infrastructure is passable for instances of *Pedestrian* and *PackAnimal* only (*poss*-aspect mentioned in 1.2).

A crucial question arises now how the Golog interpreter can combine actions. This task can – to a certain extend – be defined as a procedure (complex action) and its success highly depends on the task given. The application of Golog is not focus of this paper. It is considered representative for complex inferences and a vehicle to demonstrate integration into Ontologies as well as (side-) effects generated by this

2.4 SOL- inference beyond the Situation Calculus using its Actions

As a positive side effect, Actions defined in OWL can be inferred to a PMESII-category using the inherited property mechanism in section 2.1. Each Action that refers to a class of a Military class as a range is of Military concerns. All instances of this class and all its subclasses too. Herewith a separation of concerns can be defined implicitly and potential overlapping becomes visible. A more general side effect is that all instances or classes fulfilling an Action's prerequisites is a Leverage Point, if this Action can be executed by own forces or stuff.

3 Transferring an OWL KB to Second Order logic

In order to make First-Order-Logic as transparent as possible for "common" users such as Analysts, representing its fact base in OWL and converting it to a logic-language is considered an effective means. By proceeding so, analysts can concentrate on modeling benefitting from the support of a large Ontology community and a large number of convenient tools for creation and maintenance of these models.

3.1 Options for transformation

Since implementations of the predicate based Situation Calculus exist in Prolog this section shows how to transfer an OWL- Knowledge Base to a FOL system that complies the Situation Calculus formalization. Technically OWL can be converted to Prolog or imported by Prolog in numerous ways. Although OWL documents can be parsed (e.g. Vassiliadis 2005) by Prolog-predicates, most of the OWL document enrichments (in the sense of inferred KB structure or third party application plug-ins) get lost during serialization. As shown in Figure 10, this way of converting OWL knowledge to a Prolog Engine (depicted as (1)) leaves the inference engine and the model representation untouched – which can be an advantage if one aims at a small number (and therefore low complexity) of systems involved.

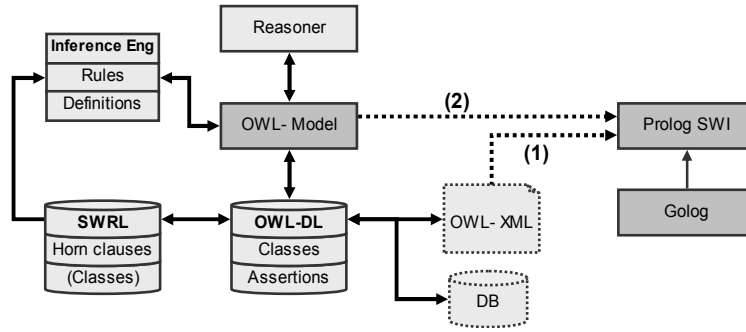


Figure 10. Two options to convert an OWL KB to FOL

Another method of transferring OWL to Prolog (specialization of (2)) aiming at neutral exchange of information is the process described in (Huang 2003). It converts data formalized in neutral exchange protocols (Bechhofer 2006) to Prolog-syntax. But applications using these interfaces neglect certain features (e.g. property descriptions such as transitivity or symmetry) if not using latest versions.

The third way of transferring knowledge to the Prolog world is shown above as (2) and uses the Ontology's Model directly. In this context, the Jena Model (Dickinson 2006) is used to represent the OWL- W3C specification (McGuinness and Harmelen 2004) as a complex data structure. Jena supports additional Ontology-languages besides OWL and can communicate with Reasoners to resolve logical expressions at structure level as well as some other inference engines. It is a Java implementation, therefore platform-independent and provides a well documented and open API for programmatic access. In the context of this work the third option is preferred.

3.2 An Transformation-Architecture

Basis for all transformations in this context an OWL-Ontology represented as Jena Knowledgebase. OWL is used as basis since it is semantically richest compared to other modeling Languages (such as XML-related technologies or the Unified Modeling Language). In order to integrate the KB into other systems it is embedded into a communication and change platform which allows the propagation of a limited set of meta-elements. These elements comply with Second Order Logic (SOL) requirements since they allow queries on FOL-predicates (Huth and Ryan 2004):

- $\text{class}(X)$ = Defines a class X .
- $\text{concept}(X, Y)$ = Defines a concept X of class Y
- $\text{subclass}(X, Y)$ = Defines X as subclass of Y .
- $\text{relation}(\text{concept}(X, Y), \text{Pre}, \text{concept}(X2, Y2))$ = Connects a domain $\text{concept}(X, Y)$ with an range $\text{concept}(X2, Y2)$ via Pre .
- $\text{restriction}(\text{ResType}, \text{Cls}, \text{Pre}, \text{ResValue})$ = Defines a restriction ResType and a restriction value ResValue for one relation Pre based on one class Cls . Restrictions for subclasses of a class may vary.

By this, every system can be integrated, that implicitly or explicitly is representable by these elements. As depicted in Figure 11, the systems can be data bases, Object Oriented Programming Languages (or CORBA), Ontologies, Logic Languages etc. Each adapted system is connected to the mediating system by an Event Listener, a Change Propagator and a Retrieval Component which can iterate over the meta elements mentioned above. Each communication between two integrated systems is transformed according to alignment-results and syntactical transformations conducted by XML-Transformers. For more see (Lehmann 2007a).

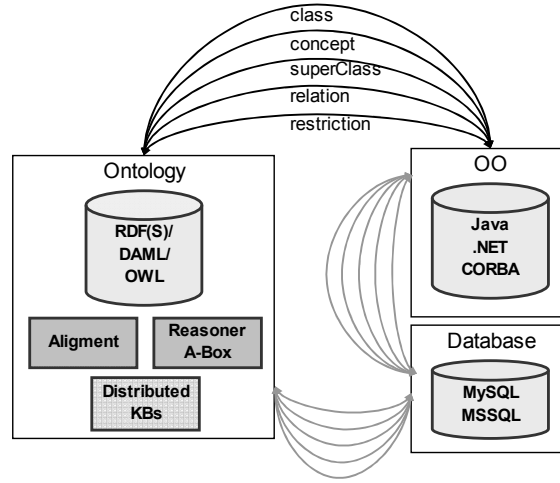


Figure 11. Ontology mediated communication

Prolog as the second (after OWL) integrated system has no strict class-type system such as most Knowledge Representation Languages or Object Oriented Programming languages. It distinguishes between strings, Boolean-values and numbers only. That is one reason why this transformation means a change in semantics at structure level. In order to generate a structure which covers the semantics of OWL's meta-elements (elements that define structure) within Prolog concepts are used which are not of Prolog's meta-elements (for meta-level descriptions see Djuric et al. (2005)). This means a modeling of OWL's meta-elements at Prolog's instance level.

The transformation from OWL to Prolog creates a Fact-Base (Friedmann-Hill 2003) and does not define predicates for inference on these facts. And it does not natively allow the usage of OWL features within Prolog. For this further preparations are required. Predicates to work on EBO-KBs usually do not depend on a certain Focus Area. In order to separate Knowledge about a Focus Area from (predicates) to work on a Focus Area's data a modularization is conducted. As depicted in Figure 12 the grey coloured "focusArea.pl"-Module contains transformation results and imports two other Modules. This is the so called functions.pl, the set of EBO-specific predicate definitions and the semantics.pl containing alignment information.

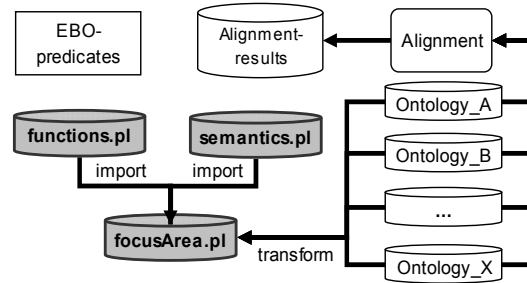
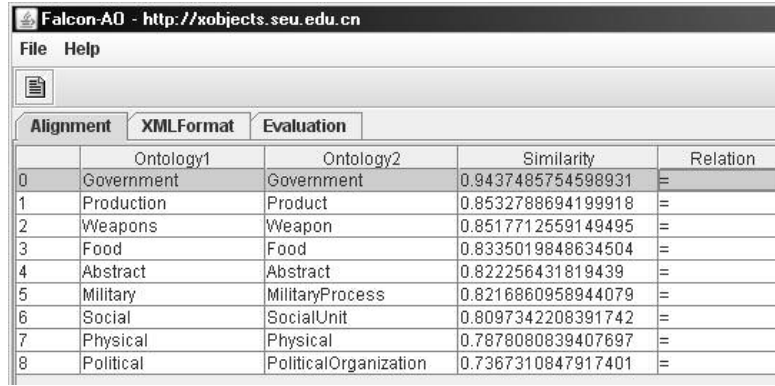


Figure 12. Concept for the integration of Alignment-results

By these imports the Alignment results can be used for all transformed KBs and allow the bridging of slight structural differences as well as of most natural language label conflicts. That is different names for similar or identical concepts.

3.3 Alignment oriented Transformation

One positive for the usage of Alignment is its ability to detect similarities based on names, structures, class/instance quotients and other characteristics (Bouquet, P. et al 2004). Figure 13 shows a subset of example results ordered by a similarity measure. For larger KBs this seems the only promising way for detecting most redundancy within a reasonable period of time.



The screenshot shows a web application window titled 'Falcon-AO - http://xobjects.seu.edu.cn'. It has a menu bar with 'File' and 'Help'. Below the menu is a toolbar with a document icon. The main content area has three tabs: 'Alignment', 'XMLFormat', and 'Evaluation'. The 'Alignment' tab is active, displaying a table with the following data:

	Ontology1	Ontology2	Similarity	Relation
0	Government	Government	0.9437485754598931	=
1	Production	Product	0.8532788694199918	=
2	Weapons	Weapon	0.8517712559149495	=
3	Food	Food	0.8335019848634504	=
4	Abstract	Abstract	0.822256431819439	=
5	Military	MilitaryProcess	0.8216860958944079	=
6	Social	SocialUnit	0.8097342208391742	=
7	Physical	Physical	0.7878080839407697	=
8	Political	PoliticalOrganization	0.7367310847917401	=

Figure 13. Alignment results

This section presents an option to integrate these Alignment results into other Inferences. The basic idea behind it is to substitute names of concepts by a flexible set of names which are connected to their according ontology.

While EBO-specific predicates allow queries such as *contradictious objectives* or indirect *logical dependencies* between concepts the semantics.pl describes extended structural definitions by integration results of alignment processes. Alignment results contain information about which property or class is sub-property or subclass of another or if two classes are identical (Bouquet et al. 2004). Figure 14 illustrates the usage of *same-as* definitions (line 12 to 14) for class memberships including a measure for similarity (relations follow the same principle). It defines which Classes are considered equal, followed by four example instances (line 17 to 20) which are member of class *Weapon* or *Weapons*.

```

11 % alignment results including similarity measure
12 sameAs('Government', 'Government', 0.93).
13 sameAs('Production', 'Product', 0.83).
14 sameAs('Weapons', 'Weapon', 0.83).
15
16 % assign four instances to classes 'A' and 'B'
17 concept('Weapons', 'A_instance_one').
18 concept('Weapons', 'A_instance_two').
19 concept('Weapon', 'B_instance_one').
20 concept('Weapon', 'B_instance_two').
21
22 % semantically equal members of different classes
23 concept_inferred(Class1, Sub) :-
24   concept(Class1, Sub);
25   (
26     sameAs(Class1, Class2, SimMeasure),
27     concept(Class2, Sub),
28     SimMeasure > 0.75
29   ).

```

Figure 14. Integration Alignment results in Prolog

A query for all instances of class *Weapons*, using *concept_inferred(Class, Instance)* (line 23 to 29) would now return all four instances of class *Weapons* and *Weapon*. Equality is – in this particular case – inferred, if (and only if) the measure for similarity between two concepts is above a certain threshold, in this example 0.75. The benefit of this mechanism is its semi-automated nature. Similarities are detected at ontology-level and transformed to Prolog, where they can be integrated without any loss of semantics. After converting the OWL KB to Prolog the Logic Inferences can be run directly.

4 Conclusion and outlook

As described in this article, an OWL Model can be transformed to a First Order Logic Language (here: Prolog). Actions defined within OWL are transformed to Prolog relations and can – if Golog compliant – be referenced by Golog interpreters. All concepts remain consistent through all steps of transformation and therefore all inputs need only to be made at OWL level exclusively. Since primitive Golog-actions are referenced by Golog procedures – describing algorithms to solve problems – the system's users do not need to know about Prolog syntax but can access Golog functionality by modelling OWL concepts (which is supported by a larger number of more user-friendly systems).

Subject to research in the near future will be a generic model for higher level problem solving algorithms. Furthermore the potentially support of "Measurements Of Effectiveness" (MOE) (Smith 2003) should be evaluated in order to determine the achievement of objectives including e.g. fuzzy measurement or vague/ missing information.

References

- Wache, H. *Semantische Mediation für heterogene Informationsquellen*. Berlin: Akademische Verlagsgesellschaft Aka GmbH 2003.
- McGuinness, D.L. and Harmelen, F.v. "OWL Web Ontology Language Overview". [cited 2006-11-24]; Available from: <http://www.w3.org/TR/owl-features/> 2004.
- McCarthy, J. and Hayes, P. "Some philosophical problems from the standpoint of artificial intelligence". *Machine Intelligence* 4, 1969. pp. 463-502.
- Levesque, H.J., Reiter, R., Lespérance, Y., Lin, F. and Scherl, R.B. "Golog: A Logic Programming Language for Dynamic Domains". *Journal of Logic Programming* 31, 1997. pp. 59-84.
- Thielscher, M. *From situation calculus to fluent calculus: State update axioms as a solution to the inferential frame problem.*, in *Artificial Intelligence*. 1999. pp. 277-299.
- Brachman, R.J. and Levesque, H.J. *Knowledge Representation and Reasoning*. San Francisco: Morgan Kaufmann Publishers 2004.
- Schiffel, S. and Thielscher, M. "Interpreting Golog Programs in Flux". [cited 2006-01-12]; Available from: <http://www.fluxagent.org/download.php?file=05-Commonsense-1.ps.gz> 2005.
- Smith, M.K., Welty, C. and McGuinness, D.L. "OWL Web Ontology Language Guide W3C Recommendation". [cited 2007-04-12]; Available from: <http://www.w3.org/TR/> 2004.
- Vassiliadis, V. "Thea: A Web Ontology Language - OWL Parser for [SWI] Prolog." [cited 2007-04-12]; Available from: <http://www.semanticweb.gr/TheaOWLParse/> 2005.
- Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. and Dean, M. "SWRL: A Semantic Web Rule Language Combining OWL and RuleML". [cited 15.10.2005]; Available from: <http://www.w3.org/Submission/SWRL/> 2004.
- Lehmann, T. "A Description Logic Example for the military context in Afghanistan". [cited 2007-04-23]; Available from: http://emma.informatik.unibw-muenchen.de/_portal/_content/persons/assistents/lehmann_files/diss/ontologies/DL_ex_Afg.owl 2007. Friedman-Hill, E. *Jess in action* Greenwich, CT: Manning Publications Co. 2003.
- Sattler, U. "Description Logic Reasoners". [cited 2007-04-12]; Available from: <http://www.cs.man.ac.uk/~sattler/reasoners.html> 2006.
- Dickinson, I. "Semantic Web research". [cited 2007-04-12]; Available from: http://www.hpl.hp.com/personal/Ian_Dickinson/semantic-web.html 2006.
- Bechhofer, S. "DIG Interface". [cited 11. Jan 2006]; Available from: <http://dl.kr.org/dig/interface.html> 2006.
- XythosSoftware, U.S. Joint Forces Command Implements Xythos in Multinational Experiment 4. 2006.
- Huang, Z. and Visser, C. "An Extended DIG Description Logic Interface for Prolog". [cited 04. Apr 2006]; Available from: <http://wasp.cs.vu.nl/sekt/dig/dig.pdf> 2003.
- Erlenbruch, T. and Horstkotte, A. *EBO Decision Support with the German Simulation System JOANA in Multinational Experiment 4*. 2005, Zentrum für Transformation der Bundeswehr. pp. 39.
- Tolk, A. *What Comes After the Semantic Web - PADS Implications for the Dynamic Web*. In *20th Workshop on Principles of Advanced and Distributed Simulation* 2006.
- Kalyanpur, A., Pastor, D., Battle, S. and Padget, J. *Automatic Mapping of OWL Ontologies into Java, June 20-24, 2004, Banff, Canada*. In *Sixteenth International Conference on Software Engineering and Knowledge Engineering (SEKE)*. Banff Canada 2004.
- Klyne, G. and Carroll, J.J. "Resource Description Framework (RDF): Concepts and Abstract Syntax". [cited 28. 11. 2005]; Available from: <http://www.w3.org/TR/rdf-concepts/> 2004.
- Tu, S. *Protege Shourt Course-Introduction*. 2005, Stanford Medical Informatics Stanford University.
- Klein, G. *Sources of Power - how people make decisions*. Cambridge, MA: MIT Press 1998.
- Huber, R.K., Eggenhofer, P., Römer, J. and Titze, K. *New Command and Control Capabilities*. 2005, NATO- NC3A: Munich. pp. 133.
- Russell, S. and Norvig, P. *Künstliche Intelligenz*. München: Pearson Education Deutschland 2004.
- Lehmann, T. *Architecture for Binding Ontology-Systems describable in Second-Order-Logic*. In *European Semantic Web Conference Workshop for Semantic Web Enabled Software Engineering*. Salzburg, Austria 2007 a
- Lämmel, U. and Cleve, J. *Lehr- und Übungsbuch Künstliche Intelligenz*. Leipzig: Carl Hanser Verlag 2001.
- Cawsey, A. *Künstliche Intelligenz im Klartext*. München: Pearson Studium 2003.
- Fallside, D.C. and Walmsley, P. "XML Schema Part 0: Primer Second Edition". [cited 2006-04-05]; Available from: <http://www.w3.org/TR/xmlschema-0/> 2004.
- DEU-Knowledge_Superiority_Cell_(KSC). *Knowledge Base Development for Effects-Based Operations Concept of Operations (CONOPS), Version 0.65*, Transformation, B.C.f., Editor. 2005, Bundesministerium der Verteidigung.
- Djuric, D., Gasevic, D. and Devedzic, V. *Adventures in Modeling Spaces: Close Encounters of the Semantic Web and MDA Kinds*. In *International Semantic Web Conference*. Galway: Workshop paper 2005.
- Bouquet, P., Ehrig, M., Euzenat, J., Franconi, E., Hitzler, P., Krotzsch, M., Serafini, L., Stamou, G., Sure, Y. and Tessaris, S. *Specification of a common framework for characterizing alignment, knowledgeweb deliverable d2.2.1v2*. 2004, Universität Karlsruhe: Karlsruhe.
- Huth, M. and Ryan, M. *Logic in Computer Science*. Cambridge: Cambridge University Press 2004.