Ontology-based Knowledge Representation and Semantic Search Information Retrieval: Case Study of the Underutilized Crops Domain

Abba Lawan, MSc.

Thesis submitted to The University of Nottingham
for the Degree of Doctor of Philosophy

March 2017

Abstract

The aim of using semantic technologies in domain knowledge modeling is to introduce the semantic meaning of concepts in knowledge bases, such that they are both human-readable as well as machine-understandable. Due to their powerful knowledge representation formalism and associated inference mechanisms, ontology-based approaches have been increasingly adopted to formally represent domain knowledge. The primary objective of this thesis work has been to use semantic technologies in advancing knowledge-sharing of Underutilized crops as a domain and investigate the integration of underlying ontologies developed in OWL (Web Ontology Language) with augmented SWRL (Semantic Web Rule Language) rules for added expressiveness.

The work further investigated generating ontologies from existing data sources and proposed the reverse-engineering approach of generating domain specific conceptualization through competency questions posed from possible ontology users and domain experts. For utilization, a semantic search engine (the Onto-CropBase) has been developed to serve as a Web-based access point for the Underutilized crops ontology model. Relevant linked-data in Resource Description Framework Schema (RDFS) were added for comprehensiveness in generating federated queries.

While the OWL/SWRL combination offers a highly expressive ontology language for modeling knowledge domains, the combination is found to be lacking supplementary descriptive constructs to model complex real-life scenarios, a necessary requirement for a successful Semantic Web application. To this end, the common logic programming formalisms for extending Description Logic (DL)-based ontologies were explored and the state of the art in SWRL expressiveness extensions determined with a view to extending the SWRL formalism. Subsequently, a novel fuzzy temporal extension to the Semantic Web Rule Language (FT-SWRL), which combines SWRL with fuzzy logic theories based on the valid-time temporal model, has been proposed to allow modeling imprecise temporal expressions in domain ontologies.

List of Publications

1. **Lawan, A.**, Rakib, A., Alechina, N., and Karunaratne, A., 2016: The Onto-CropBase - A Semantic Web Application for Querying Crops Linked-Data. In Beyond Databases, Architectures and Structures. Advanced Technologies for Data Mining and Knowledge Discovery Volume 613 of the series Communications in Computer and Information Science, pp 384-399.

2. Rakib, A., **Lawan, A.**, and Walker, Sue., 2014: An ontological approach for knowledge modeling and reasoning over heterogeneous crop data sources. In: Pattern Analysis, Intelligent Security and the Internet of Things, Advances in Intelligent Systems and Computing, Volume 355, 2015, pp 35-47.

3. Panchanathan, S., **Lawan, A.**, and Rakib, A. 2015: MyGeo-Explorer: A semantic search tool for querying geospatial information. In ARPN Journal of Engineering and Applied Sciences, Vol.10(23), 2015.

4. **Lawan, A.**, Rakib, A., Alechina, N., and Karunaratne, A., 2014: Advancing Underutilized Crops Knowledge using SWRL-enabled Ontologies - A survey and early experiment. In: CEUR Workshop Proceedings of The Second International Workshop on Linked Data and Ontology in Practice, Vol-1312, Pages 69-84, 2014.

5. **Lawan, A.**, Rakib, A., Alechina, N., and Karunaratne, A., 2015: PGR Research Showcase - The Onto-CropBase Poster Presentation. By: UNMC Research and Knowledge Transfer, Jalan Broga Semenyih, Malaysia. April 2016.

6. **Lawan, A.**, Rakib, A., Alechina, N., and Karunaratne, A., 2014: PGR Re-

search Showcase - Future Web for Future Crops. By: UNMC Research and Knowledge Transfer, Jalan Broga Semenyih, Malaysia. April 2015.

Acknowledgments

I would like to express my highest appreciation to my supervisor Dr. Abdur Rakib for the continuous support and motivation throughout my Ph.D. study and related research. Special thanks to my co-supervisors: Dr. Natasha Alechina and Dr. Asha Karunaratne for their insightful comments and encouragement, which helped to widen my research perspective.

I thank Dr. Chris Roadknight and Dr. Chen ZhiYuan for their professional advice and enlightenment during the first and second-year evaluation of my research work. I also thank my fellow CS Research labmates for all the stimulating research talks and fun discussions in the past four years.

Last but not the least, I would like to thank my family for their continued support both physically and spiritually, special mention to my wife Aisha and our daughter Nur Khadija for their patience throughout this thesis writing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis deals with the problems of semantic heterogeneity in domain knowledge representation, integration, and sharing, the goal of which is to achieve consistent and reliable knowledge systems. As a foundation of any Knowledge Representation System (KRS), ontological analysis helps to clarify the structure of domain knowledge, disambiguates conflicting terminologies and ensure consistent vocabulary. We categorize ontology-based knowledge systems into two broad categories: reference and legacy ontology-based knowledge systems. The reference ontology-based systems utilize ontologies simply as standardization models for existing knowledge bases. Whereas, the legacy ontology-based systems uphold ontologies as 'part and parcel' of a knowledge system. Basic tasks of ontological knowledge modeling crossroads between the knowledge representation field of AI (Artificial Intelligence) and the Semantic Web project, which employs semantic technologies and techniques to extend the existing World Wide Web into the Web of linked-data.

In the succeeding sections of this chapter, an overview of the research area is introduced in Section 1.1 and Section 1.2, highlights the motivation and problem statement behind the tasks of Ontology-driven CropBase knowledge system. The research questions were further discussed in Section 1.3 highlighting the existing research gaps followed by formulated questions to be answered during the course of the project. The thesis Aims, objectives and Methodology is then presented in

Section 1.4. This is followed by a brief summary of the thesis contributions in Section 1.5 and a thesis ontology visualization is shown in Section 1.6 highlighting the relationships between the research contexts. Lastly, the overall outline of the thesis is presented in Section 1.7.

1.1 Background: Overview of the Research Area

Knowledge representation in the field of AI involves the use of principles and structures to preserve information in a way that facilitate inference. However, due to the divergent nature of application domains and the methodologies employed in knowledge engineering, similar information can appear to be entirely different. This is either due to inconsistencies in nomenclature or the diversity of knowledge structures involved, the latter often resulting in greater ambiguity. Moreover, these ambiguities and inconsistencies in domain knowledge remain the major obstacles to sharing data and knowledge among disparate researchers. Such problems are not uncommon in crop-related research areas such as the Crops for the Future Research Center (CFFRC), where researchers from different background must work together to produce reliable knowledge systems that will aid users in decision-making on Underutilized Crops (UCs) and their products. Moreover, little information exists on UCs and the available knowledge is usually available in informal sources on the web, such as Wikipedia. However, such information are not entirely authoritative and usually incomplete, thus not suitable for critical decision support. In essence, a standard vocabulary (ontology) needs to be shared and adhered to by researchers in order to interface the correlated research goals for the center.

Due to their powerful knowledge representation formalism, associated inference mechanisms, and semantic interoperability among divergent resources, ontologies have been increasingly adopted as knowledge modeling tools, by experts from different application domains. Moreover, considering the hierarchical nature of crop-related taxonomies and the promising role of semantic web applications in the future of decision support systems, an *Ontology-driven knowledge representation system* was proposed to standardize the vocabulary of underutilized crops knowledge (for the CFFRC and its partner organizations) and to help in developing such applications that will aid users in decision-making on underutilized crops.

Ontology is one of the knowledge representation tools for the Semantic Web [6], designed to provide explicit specification of concepts in a particular domain. As a web of data [7], ambiguities and inconsistencies are bound to exist in the Semantic Web as data moves across applications, enterprises, as well as community boundaries. As such, formalized domain knowledge representation standards are needed and more importantly, ontologies are supported by logic programming rules for added expressiveness, with inference mechanisms and reasoning tools to ensure consistency of stored knowledge.

This ability to reason about knowledge consistency and infer new knowledge from existing facts, make ontologies not only useful in the semantic web but also a decent alternative to static knowledge-bases for decision support systems. In this research work, we proposed an ontology-driven CropBase knowledge system for underutilised crop research and development. The tasks will contribute to Knowledgebase population and continued platform development. Research activities will include design and implementation of an ontology-driven integration tool that will enable access to information and assist in decision making for crop growers. We discuss more on the definition of ontologies and their role as knowledge repositories in Chapter 2.

1.2 Motivation

With the United Nation's decade long efforts on food security, there is an awakening on the need to revitalize the cropping of neglected (underutilized) crop species, many of which have the potentials of providing food security and nutritional sustainability [8, 9]. As the name implies, Underutilized Crops are those that are currently neglected though previously grown and consumed with considerable nutritional or market value [10]. However, lack of formal technical knowledge was identified as one of the constraints to research and development on these group of crops. The Crops for the Future (CFF)[1], is one of the research bodies dedicated

[1]http://www.cffresearch.org/

for research and development on Underutilized Crops and with various researchers working on related projects, there is the need for standardized knowledge representation framework for efficient knowledge-sharing and integration of its diverse sources.

Due to their wide acceptance as knowledge-sharing and integration tools, domain ontologies have been effectively utilized as knowledge management tools most especially in the field of Life-Sciences. In this thesis, we explore the ontology-based knowledge systems and propose the use of ontologies with relevant semantic web technologies to advance knowledge standardization and information-exchange on underutilized crops as a domain, with the ultimate goal of developing an efficient knowledge model for a farming decision support system.

1.2.1 Problem Statement

The Ontology-based domain knowledge model in Web Ontology Language (OWL), being both human as well as machine understandable, aims to serve as integration point for Underutilized Crops meta-data and provide efficient knowledge-sharing formalisms for data exchange among researchers, data sources and across knowledge systems.

However, while various crops ontologies do exist, as discussed in Chapter 2, they mostly conceptualize major crops with little to no information on the underutilized crops — the main focus of the Crops for the Future (CFF). These ontologies moreover, are mostly developed through tedious methodologies and using highly inexpressive ontology languages with little chances of utilization. Moreover, ontology contents typically focus on representing the crop anatomy, genes and their physiological structures, which only serves as collective nomenclature targeting few domain experts and therefore not suitable for wider sharing and reuse. These confirms the CFF assertion that ”Underutilized crops are also those under-represented” [2].

The lack of efficient knowledge models on underutilized crops is a major research problem [10] and typically represents a common challenge to knowledge-

[2] Crops for the Future (CFF) 2015: http://cffresearch.org/FutureCrop-@-LandingArticle.aspx#Underutilised_Crops

sharing and terminological standardization. Furthermore, the inexpressiveness of these ontology languages leads to various inconsistencies in domain knowledge gathering and utilization thereby resulting in insufficient knowledge systems for decision making, among others.

With the proposed UC-ONTO development methodology using the Web Ontology Language (OWL) and the Semantic Web Rule Language (SWRL) for added expressiveness, an Underutilized Crops Ontological knowledge system will be developed to provide a consistent model for standardization of underutilized crops concepts and related terminologies as well as provide an efficient knowledge-sharing formalism for the underutilized crops meta data. In a nutshell:

> ”The study addresses the problems of semantic heterogeneity in domain knowledge representation, integration and sharing — the goal of which is to achieve consistent and reliable knowledge systems.”

1.3 The Research Questions

Following the review of essential literature, we first identified key areas of existing research gaps on the use of semantic technologies (ontology-based systems) to advance the knowledge management of crops, as follows:

- Insufficient knowledge models on underutilized crops — As mentioned earlier, we quickly realized that most of the underutilized crops (UCs) were also under-represented. This is not only true for the semantic knowledge models but also in the main literature. Furthermore, with the major focus of our data sources (the CFFRC), being to create, share and utilize the undocumented knowledge regarding the farming practices of UCs for their entire value-chain (farm to market), the need for ontological models to formally represent the domain facts cannot be overemphasized.

- Research gaps in Applications and Ontology Utilization — While this is not peculiar to the crops knowledge domain, it is a widely accepted fact in the Semantic Web community that there are more ontologies being developed than they are being utilized. This can be seen from the thousands of ontologies deposited in various repositories [3] but with only a handful of applications that utilize such ontologies.

- Research gaps in ontology modeling languages — With focus on the widely accepted semantic web ontology languages, our initial experiments of developing the Underutilized Crops Ontology models quickly exposes the expressive limitations of both OWL and its rule extension, the semantic web rule language (SWRL). For one, OWL can neither comprehensively express incomplete knowledge, nor queries temporal information inherent in a domain knowledge. Whereas, the SWRL formalism is lacking expressive constructs for managing complex temporal knowledge such as the imprecise temporal expressions (ITEs), which commonly exists in expert narratives of domain facts and processes.

To address these research gaps, we discuss several research questions regarding 'ontology-based knowledge systems' for underutilized crops. However, the two main research questions can be summarized in the following:

> *The use of ontologies to standardize knowledge representation in the field of Crops, and how Semantic Web agents can utilize those ontologies with relevant linked-data to aid users in decision making on Underutilized Crops and their products.*

More specifically, the following questions were asked to come up with the above research statements:

Q1. Can the Semantic Web technologies (esp. ontologies) be realized and utilized in the Agricultural domain?

Q2. How can the available crops data be effectively modeled to provide a standard representation model of relevant concepts and terminologies?

The resulting model should allow defining new terminologies, aligning and merging existing models, checking for consistencies and overall classification of concepts.

[3] https://www.w3.org/wiki/Ontology_repositories

Q3. Can the problem of ambiguity in dealing with concepts/terms be resolved using ontologies? Furthermore, can the ontologies sufficiently provide a controlled vocabulary of terms and concepts that can describe the underutilized-crop domain?

The resulting ontology is expected to standardize underutilized crop knowledge representation across contributing databases.

Q4. Can the resulting ontologies and other data sources be annotated with relevant semantic techniques to provide a semantically interlinked data for underutilized crops?

This linked data can be used in conjunction with our domain ontologies to provide a pool of knowledge for the Semantic Web (SWeb) agents to make informed decisions when answering user queries.

Q5. Can Semantic Web Applications be developed as interactive decision support systems that utilize the available knowledge in the ontologies to answer user queries relevant to the underutilized crops domain?

Q6. How can the ontology development languages be enhanced to effectively represent and query imprecise temporal information inherent in domain knowledge ?

1.4 Aims, Objectives and Methodologies

The aim of the thesis is thus to formally model the factual knowledge on underutilized crops and related concepts using OWL-based ontologies and utilizing those ontologies with relevant linked-data, to serve as knowledge bases for farming decision support system. The choice to develop the underutilized crops ontology in OWL (Web Ontology Language) was made in order to utilize its expressive powers (see Sections 3.2 and 4.2.1) and with OWL being the standard ontology language for the Semantic Web, approved by the World Wide Web Consortium (W3C) [4], efforts to provide the crop ontologies in OWL format will undoubtedly

[4]http://www.w3.org/Consortium/

enhance knowledge-sharing and integration in the field of underutilized crops. Moreover, semantic web applications can be developed to easily utilize such ontologies. To this end, a semantic search tool called the Onto-CropBase was designed and developed (see Sections 3.4 and 4.3). The limitations of the expressive powers of OWL motivates the semantic web community to consider the use of SWRL rules for added expressiveness (see Chapter 2.8.3) and the thesis further explored the expressive limitations of the SWRL formalism leading to the evolution of a new language extension (The FT-SWRL model) for representing imprecise temporal expressions commonly found in domain knowledge was proposed (See sections 3.5 and 4.4).

To achieve the goals of the proposed research, we apply the following methodology:

1. Studying existing information: analyzing existing vocabularies in the domain of Crops (focusing on underutilised crop, e.g. Bambara groundnut) and identify potential problems and semantic ambiguities. Identify and collect information sources required for the development of the knowledge tool.

2. Underutilized Crops ontology design and development: combine existing crop terminologies and expand them with new information concerning underutilised crops and other relevant knowledge. Express this information in Web Ontology Language (OWL) augmented with the declarative assertions of the Semantic Web Rule Language (SWRL) for comprehensive modeling. The ontology engineering approach incolves:

   i. Concepts generation through Data sources, Competency questions, ontology reuses and reflexive ontology development.

   ii. Populate the Ontologies with relevant data sets from available knowledge sources.

   iii. Develop domain-specific, user-defined rules (SWRL rules) to describe complex concepts of the knowledge domain.

3. CropBase tool: the tool will utilize the underutilized crops ontology as the

main knowledge source for answering user queries. It combines the ontology with relevant knowledge linked as RDF datasets for completeness.

Data Sources, Tools and Target Audience. Other relevant research design considerations include:

4. The Data Sources: major knowledge sources for our research includes experimental data (published and unpublished) from the Crops for the Future (CFF) researchers and partners. Others includes, collaborative ontology development environments focusing on crops ontologies and individually sourced domain knowledge from other collaborative CFF research projects.

5. Tools: for flexibility and cost effectiveness, we rely heavily on open-sourced development tools and platforms. For the ontology development, we select Protégé[5] ontology editor as the development tool being a rich open-source development environment with vibrant community of developers and user support. For the CropBase tool however, Java Enterprise Edition (Java EE) and the Apache Jena[6], a java system for RDF, were used in the Eclipse Integrated Development Environment for the development of the semantic search engine. Other various tools were also utilized but omitted here for brevity.

6. Stakeholders: targeted (expected) users of our ontologies and tools include all persons and/or organizations interested in underutilized crops and their products, such as: CFFRC and its partners, Farmers, Agricultural organizations, etc.

1.5 Contributions

This thesis explored the field of semantic technologies and ontology-based knowledge systems, their approaches, frameworks and practices all of which led to the

[5]http://protege.stanford.edu/
[6]https://jena.apache.org/

following novel contributions:

The thesis investigated the ontology development and standardization approaches and the integration of ontologies with logic programming rules for added expressiveness. Due to their powerful knowledge representation formalism and associated inference mechanisms, ontology-based approaches have been increasingly adopted to formally represent domain knowledge. The thesis then proposed the use of ontologies to advance knowledge-sharing of underutilized crops as a domain and investigated the integration of those ontologies in OWL (Web Ontology Language) with SWRL (Semantic Web Rule Language) rules for added expressiveness, see Sections 3.2 and 4.2.1. From the investigations, it can be concluded that ontology development in the agricultural domain can best be tackled through 'knowledge reuse' of existing upper domain ontologies.

As regards the techniques for ontology generation, an extensive investigation is on the existing methodologies of ontology development is presented, counting their major benefits and drawbacks. The thesis summarized these methodologies as either from scratch or through ontology reuse. In either case, extra tasks of *'ontology versioning* and *assembly'* were recommended to manage the problem of tracking ontology versions and how they can best be assembled into a coherent domain specification respectively. Moreover an investigation on how to generate ontologies from existing data sources in presented with a reverse-engineering approach of generating domain specific conceptualization through 'Competency Questions' posed from possible ontology users to the domain experts. A validation of these techniques was undertaken through detailed case studies presented, where we pioneered the first SWRL-enabled underutilized crops ontology (UC-ONTO) in OWL. The ontology specifies the basic information of the underutilized crops focusing on the farming practices of Bambara groundnut as an exemplar crop.

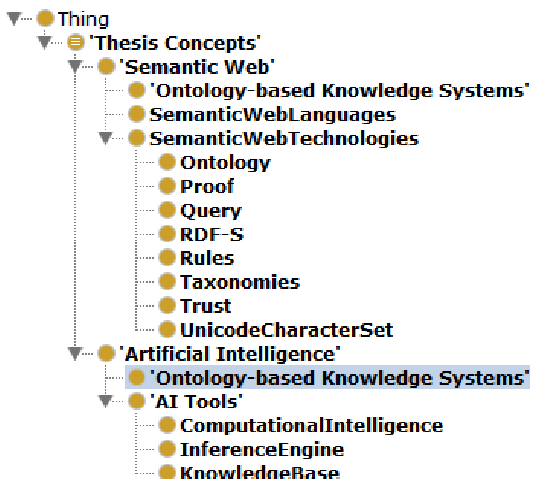
While the ontological knowledge modeling is imperative, of equal importance is the standardization of this knowledge. In this area, the thesis show how the Competency Questions, designed at the beginning of ontology development process, can be used to check the quality of the knowledge model by posing them as queries. The responses can always be measured in comparison with the elicited knowledge from the domain experts and inconsistencies can be remedied. This and other ontology standardization approaches mentioned in Section 4.2.6 helps in better understanding of ontologies, which can lead to further reuse.

Regarding ontology utilization, a model was proposed and implemented on the ontology-based semantic searching of linked data. A semantic search engine called the Onto-CropBase, was provided as a case study, which serve as web-based access point for the underutilized-crops ontology model. The knowledge base integrated other domain knowledge in the form of linked-data in Resource Description Framework (RDF) schema for comprehensiveness. Presented here in Sections 3.4 and 4.3, the model emphasized the 'hybrid integration' approach to linked-data management, which allows for generating federated SPARQL queries through the semantic search engine. Whereas successful ontology development efforts help to strengthen semantic knowledge modeling and representation, the ontology utilization though has remained a challenge in the semantic web community. Efforts such as the Onto-CropBase tool [11], will greatly help in advocating the dissemination of knowledge, from the numerous ontological knowledge bases, to non-technical stakeholders for informed decision-making. From the preliminary evaluation of the tool, it was concluded that the ontology-based semantic search system can efficiently serve as a first-hand information portal on underutilized crops knowledge domain.

Regarding the limitations of expressive powers of OWL and SWRL as ontology and rule languages for the semantic web respectively, an extended expressiveness formalism is proposed to the duo to allow for modeling even more complex scenarios. Specifically a SWRL model is proposed to handle Imprecise temporal expressions commonly found in domain knowledge. While the OWL/SWRL combination offers a highly expressive ontology language for modeling knowledge domains with higher degree of flexibility, the combination is found to be lacking supplementary descriptive constructs to model complex real-life scenarios – a necessary requirement for a successful Semantic Web. To this end, the thesis explored the common logic programming formalisms for extending Description Logic (DL) ontologies and investigated the state-of-the-art in SWRL expressiveness extensions, proposed over time. A novel fuzzy temporal extension to the Semantic Web Rule Language (FT-SWRL) is then proposed, which combines fuzzy theories based on valid-time temporal model. The FT-SWRL formalism, presented in Sections 3.5 and 4.4, offers a standardized approach for modeling imprecise temporal information in OWL ontologies. The SWRL fuzzy temporal model is introduced in two parts: a SWRL fuzzy temporal ontology and a set of imprecise temporal built-ins. While the ontology model defines the necessary linguistic terminologies and variables, the SWRL fuzzy temporal built-ins defines the semantics for operation on the data modeled using the fuzzy temporal ontology model.

1.6 Thesis Context Ontology

As earlier mentioned, the thesis context intersects two main areas of research, namely, the knowledge representation of AI and the Semantic Web. In Fig. 1.1, we present our thesis context in a form of ontology for highlighting the contextual relationship between the components of the thesis and the underlying research areas.



```
Thing
  'Thesis Concepts'
    'Semantic Web'
      'Ontology-based Knowledge Systems'
      SemanticWebLanguages
      SemanticWebTechnologies
        Ontology
        Proof
        Query
        RDF-S
        Rules
        Taxonomies
        Trust
        UnicodeCharacterSet
    'Artificial Intelligence'
      'Ontology-based Knowledge Systems'
      'AI Tools'
        ComputationalIntelligence
        InferenceEngine
        KnowledgeBase
```

Figure 1.1: Thesis Ontology fragment showing Concepts from the research context

While the field of AI has been around for decades, the Semantic web vision was outlined in [6] in the early 2000's. In terms of data semantics, the Semantic Web leverages existing techniques of AI, though implementations are achieved through new set of semantic web technologies as clearly shown in Fig. 1.2. Built on the basic RDF technology (see Fig. 2.1), knowledge representation on the semantic web focus on graph data model, which is a simple interconnection of pieces of data. However, more AI comes to play a role with the layering of ontologies, rules and logic layers aimed to allow more expressiveness in knowledge representation, reasoning and inference. We discuss more on ontologies and the Semantic Web in Chapter 2.

1.7 Thesis Outline

The studies leading to this thesis are outlined here. The thesis comprises six chapters, each organized to highlight the evolved nature of our contributions. The chapters are introduced as follows:

- Chapter 1: **Introduction** set the stage for ontology-based knowledge systems, highlighting the background of the study, motivation, aims and the research objectives. Followed by the research questions, methodology of the research, a summary of our contributions and a thesis context ontology to graphically highlight the research context.

- Chapter 2: **Literature and Methods** focuses on six key dimensions, which consolidate the reviewed literature; following the chapter introduction, Section 2.2 discusses the ontology and the semantic web. Ontologies as knowledge management tools are then discussed highlighting the classifcation of ontologies followed by exploring the relevant works of ontologies in the Crops domain. The ontology development approaches, ontology development environments, and ontology evaluation are then discussed followed by the literature on extending ontologies with logic programming rules. The expressiveness extensions of the SWRL were then explored with a critical review of their decidability requirements. Ontology-based search engines

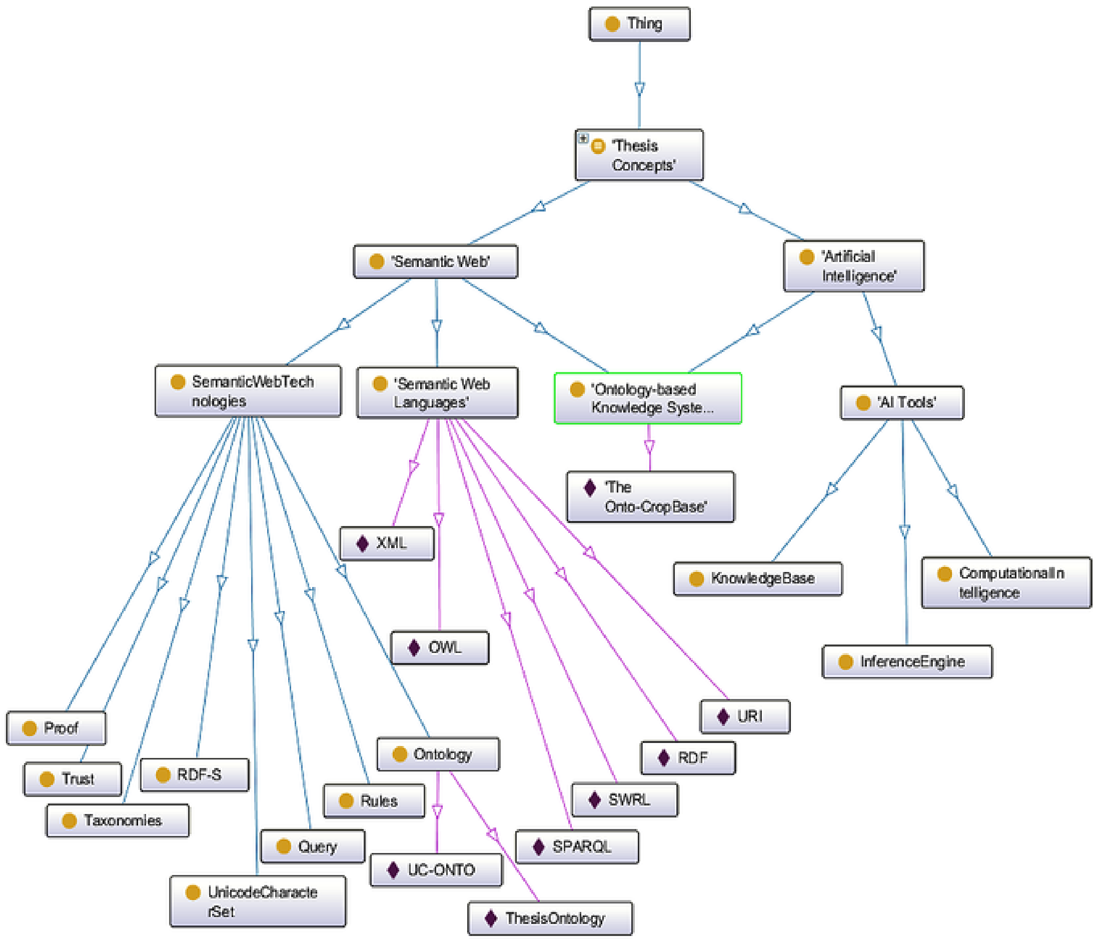


Figure 1.2: Thesis Ontology graph showing detailed thesis concepts.

Note: we omit the details of AI components and the philosophical background of ontologies for brevity and focus.

are then discussed to highlight the relevant works. Lastly, a review on ontology language extensions is presented with a focus on research efforts in managing temporal uncertainties in OWL ontologies, followed by the chapter summary

- Chapter 3: **Research Methodology** explores the research methods and approaches for ontology engineering, ontology utilization and the ontology language extension. Following the chapter introduction, the next section highlights the ontology engineering methodology for the Underutilized crops Ontology (UC-ONTO). The Ontology development with Protege section highlights development process of ontologies using the Protege ontology development environment. This is followed by the Onto-CropBase development methodology highlighting the knowledge base integration approach, the semantic search interface design and the selection of the mediator component. Lastly, the model design of the Fuzzy temporal semantic web rule language (FT-SWRL) extension is then presented discussing the design of the pioneer FT-SWRL model followed by the Chapter summary.

- Chapter 4: **Results: Case Studies** describes the results as implementation cases of the pioneer underutilized crops ontology(UC-ONTO), the Onto-CropBase semantic search engine and the proposed FT-SWRL model. Following the chapter introduction, case study one describes the results of our Ontology engineering, standardization and extension with SWRL rules. This is followed by the Ontology utilization case study describing the Onto-CropBase semantic searching and functionalities. The Ontology Language extension case study describes the model implementation of the Fuzzy Temporal SWRL model and lastly, the Chapter summary.

- Chapter 5: describes **The Analysis of Results and Discussions** highlighting the validation of the case studies development process. Following the chapter introduction, three main sections were presented in the chapter: The Underutilized crops ontology evaluation is first discussed following the key metrics of ontology evaluation and the validation of the SWRL rules extension of the UC-ONTO. This is followed by the Onto-CropBase evaluation highlighting the domain experts evaluation, comparison with relevant tools and an analysis of the scalability of the Onto-CropBase tool. Lastly, the FT-SWRL model evaluation is discussed highlighting the metrics of the FT-SWRL ontology followed by the format and usability of the FT-SWRL rules and built-ins respectively. A discussion and summary is then presented with a general overview of the main sections.

- Chapter 6: **Conclusion** of the thesis begins with a summary of the major contributions followed by a discussion of the technical limitations and suggestions for improvement as future works. Lastly, a summary of the research conclusion is presented as the concluding remarks.

Chapter 2

Literature Review

2.1 Introduction

This chapter gives an overview of the research context and the related works in ontology-based knowledge systems. The chapter is structured as follows: It first introduce ontologies and the semantic web in Section 2.2 including an overview on the evolution of web ontology language (OWL) from the Description Logics (DL), the expressiveness of OWL-DL, and the OWL-2 profile. An overview of ontologies in the Crops domain is presented in Section 2.4 to highlight the relevance and state-of-the-art of the knowledge domain. This is followed by discussion of the literature behind ontologies as knowledge management tools in Section 2.3 and the relevant approaches to ontology engineering methodologies in Section 2.5. The discussion categorizes ontology modeling tasks into three: ontology engineering from domain facts, ontology reuses and ontology standardization for improvement. Section 2.6 gives an overview of some of the popular ontology development environments highlighting their comparable advantages and limitations. The next section explores the ontology evaluation strategies in 2.7 with emphasis on the ontology evaluation through domain experts and competency questions. Common logic programming formalisms for extending DL-based ontologies are presented in Section 2.8 highlighting the motivations, concerns and the decidability requirements for each formalism. Suitability of the Semantic Web Rule Language (SWRL) as an example formalism for extending OWL ontologies was established by exploring its compatibility, limitations, syntax and semantics among others. Various SWRL expressiveness extensions were then discussed in Section 2.9 followed by a study of their decidability and completeness. Ontology-based search engines were reviewed in Section 2.10 highlighting the comparable literatures behind keyword-based versus semantic search systems. Section 2.11 explores the relevant works on ontology language extensions focusing on managing temporal imprecisions in domain knowledge representation and lastly, Section 2.12 gives a brief summary of the reviewed literature.

2.2 Ontology and The Semantic Web

The Semantic Web is envisioned as an extension to the current web of documents. It is also described as Web of linked data [6]. The semantic web project, which uses ontology as its knowledge model, aims to connect conceptual data from different sources and enrich them with semantic annotations for the comprehension of both humans and machines. One major goal of the Semantic Web is to provide markup services that enables access to information from various sources in the Web. Hence, the Semantic Web is more concerned on the meaning of data as compared to the current World Wide Web (WWW), which is more concerned with human readable structure with little emphasis on machine inference. Figure 2.1 below shows the Semantic Web Stack.

Ontology on the other hand, is a term borrowed from philosophy and is widely quoted as ”an explicit specification of a conceptualization” [12]. ’Conceptualization’ is any simplified version of knowledge that we wish to represent for some purpose. Ontology is also defined as any common vocabulary employed and adopted to share information in a given domain [13]. By adding semantics to domain concepts, ontologies help to model a knowledge base that is both human understandable as well as machine readable. As such, developing ontologies requires a reasoner to be invoked to detect inconsistencies and for inference. Commonly used reasoners include Pellet, HermiT, Fact++, KAON2, Cerebra Engine and RACER among others [14].

Designed to be the knowledge modeling language for an open-web of linked data [15, 16], ontologies are often developed as foundational specifications, do-

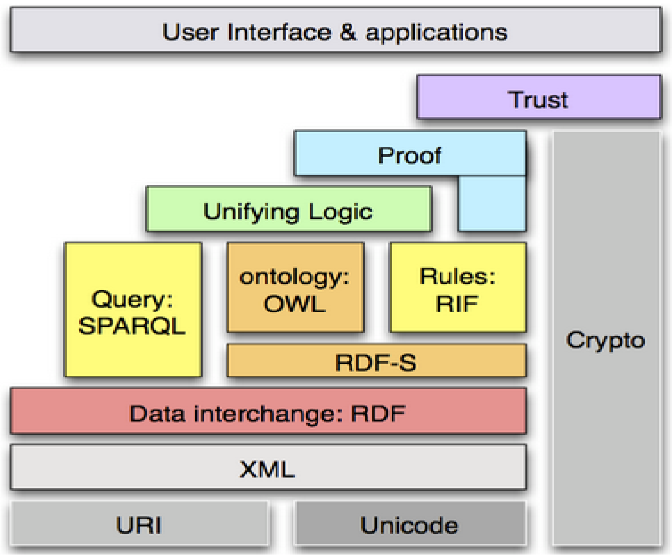


Figure 2.1: The Semantic web language stack[1]

main conceptualizations, as well as task-specific domain ontologies. The foundational ontologies explain generic concepts, domain ontologies explain the vocabulary of a scoped knowledge area, and application or task ontologies describes activities or specific application vocabulary in a domain. We discuss ontology classification citing examples in Section 2.3.2.

2.2.1 Preliminaries

- Clausal Logic (CL) [17] is a particular kind of First-Order Logic (FOL) — a powerful formalism used to express relationships between objects in a given knowledge domain through quantified variables and predicates. CL provides the basis for Logic Programming (LP) through its fragment called Horn Clause Logic (HCL).

- A Horn-Clause is logic program rule containing disjunctions of literals (rule atoms) with at most one positive literal [18]. A finite collection of Horn clauses together with ground facts are referred to as logic programs (LP). Simply put, LPs are a finite set of rules and facts.

- Datalogs on the other hand, are function-free horn-clauses, having only variables and constants as terms.

- Description Logic (DL) [19] is basically a decidable fragment of FOL that allows modeling of knowledge domains using concepts (classes), the binary relations between them (called roles or properties), and individual instances (facts). Being decidable, the DL forms the basis of most domain modeling (ontology) languages including OWL.

- Decidability: In the context of this work, decidability refers to the ability of a Reasoner to achieve inference within a finite amount of time [20] e.g. checking the consistency of logical consequences of ontology axioms and return true or otherwise. That is, given any ontology, a set of rules, and a sentence, the Reasoner can check that the sentence is entailed by the ontology and rules. More importantly, there exists a terminating procedure.

2.2.2 Semantic Web Technologies

The Resource Description Framework (RDF), its schema RDFS and the Web Ontology Language (OWL) are the common ontology languages for the semantic web. Fig. 2.2, shows these languages arranged in layers of syntax and semantics.

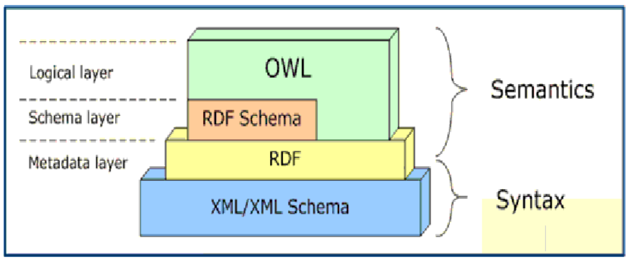


Figure 2.2: Layered architecture of ontology languages [21].

In principle, RDF is an XML-based language designed to identify and describe information in a web page or any object on the web. It is lightweight and flexible. The basic elements of RDF are resource, property and statement. As explained in [22], the expressive limitations of Resource Description Framework (RDF) and its Schema in describing some application domains brought about the need for

OWL ontologies on the web and thence necessitates the development of expressive ontology development languages. A summary of the web ontology language evolution is presented in Fig. 2.3 showing the OWL family tree.

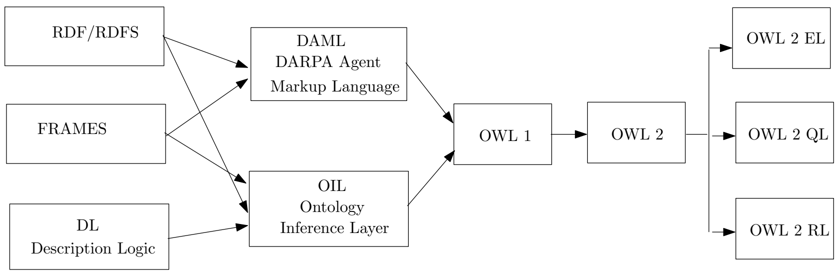


Figure 2.3: OWL family of ontology languages

The earliest is the extension of RDF by the Defence Advanced Research Agency (DARPA) and W3C to develop ”DAML” (DARPA Agent Markup Language), an ontology language that have more expressive capacity than RDF to allow agents' interaction on the web [23]. As shown in Fig. 2.3 the DAML ontology language is designed to further incorporate the flexibilities of RDF with the expressive powers of FRAMES knowledge representation language [24]. Frames-based knowledge model represents domain knowledge using Classes to represents objects, Slots to represent named binary relationships between the Classes and Facets, which describes a named ternary relationship between classes and primitive data types. The quest for higher expressiveness of domain ontology languages influenced by the available implementation tools, such as RDF and FRAMES, leads to the evolution of the 'Ontology Inference Layer' (OIL). The OIL [25] is a DL-based ontology interchange language that combines the conceptual definition of Description logics with the explicit knowledge representation formalism of frames while remaining compatible with the RDF schema. Hence the name, ontology interchange language.

With the introduction of the Semantic Web however, it became imperative to have a web-based ontology language, an effort that leads to the adoption of Web Ontology Language (OWL 1). OWL 1 is also termed as OWL-Full, due to its full expressive powers combined from the DAML and OIL ontology languages. In order to achieve reasoning over OWL ontologies, a subset of OWL-Full that completely conforms to the DL framework is extracted as the 'OWL-DL'. Hence, at the cost of decidability, the OWL-Full subsumes the OWL-DL in terms of expressiveness. Furthermore an OWL-Lite profile is defined as a subset of OWL-DL and becomes the least expressive though highly tractable fragment of the web ontology language OWL 1. In essence, the web ontology language (OWL-DL) gives a very expressive and decidable ontology language and since OWL-DL utilizes the extensive research on DL, it serves as the foundation for the recent language profiles of OWL version 2 — discussed below.

2.2.3 The OWL-2 Profile

The Web Ontology Language version 2 (OWL 2) [26] is a formally recommended ontology language for the Semantic Web and much like its predecessor, OWL 1, it allows logical domain modeling by defining and describing classes, individuals, their properties and relationships with each other or data values, with the addition that OWL 2 ontologies are exclusively stored as Semantic Web documents. Specifically, the current version OWL 2 is able to provide a wider range of constructs for expressing concepts such as $transitive$ and $inverse$ properties, $cardinality\ restrictions$, as well as $inheritance$, among others. By targeting specific modeling needs of the web, it is thence divided into three sub profiles, viz. the 'OWL 2 Expressive Language' (OWL2EL), 'OWL 2 Query Language' (OWL2QL), and the 'OWL 2 Rules Language' (OWL2RL) [27, 28]. These sub-languages offer different expressiveness and computational desirability.

The 'OWL 2 RL', which lays the foundation for the Semantic Web Rule Language (SWRL) (see Fig. 2.4: OWL Evolution and Contextual Relationship with SWRL), is suitable for rule-based applications. It enables additional rules (such as horn clause rules written in the SWRL language) to be added to ontologies for more expressive descriptions of application domain. SWRL is an OWL-based rule language, which utilizes the abstract syntax of OWL extended with horn clauses (having antecedent and consequence) for rule assertions. The SWRL formalism is discussed in Section 2.8.2 and the Fig. 2.4 below, summarizes the contextual relationship between OWL profiles and SWRL.

Reasoning tasks over OWL 2 RL ontologies are achieved in Polynomial times and due to its ability to manipulate RDF triples directly, it is used for applications

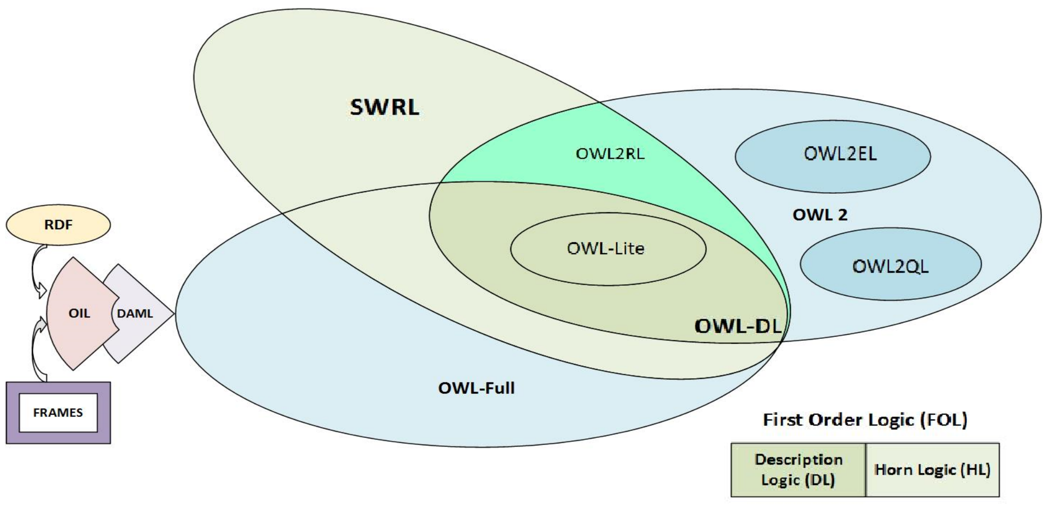


Figure 2.4: OWL Evolution and Contextual Relationship with SWRL

that need to access data directly or where data manipulation is important. The profile allows extending ontology with rules and OWL 2 RL semantics can be implemented using traditional rule-based engines [29] — the forward or backward chaining rule engines, including CLIPS and Jess. As such, the profile is basically more expressive than OWL 2 EL and OWL 2 QL, and allows developing applications with scalable reasoning while retaining the expressive power of OWL 2.

2.2.3.1 Reasoning in OWL Ontologies

One of the key benefits of using DL-based ontologies and of using ontology (in general) over other knowledge representation techniques as a whole is the ability to invoke a reasoner to process those ontologies [30]. Processing ontologies by a reasoner involves testing the hierarchy definition of classes in the ontology and then automatically compute the class hierarchy and also infer additional classification. Reasoning in OWL ontologies also helps to provide consistency checking, where a reasoner checks based on the given class definition whether or not a class can have any individual instances. This helps to avoid impractical classification of concepts in OWL ontologies.

Common DL-based reasoners available in open ontology development environments include the Pellet reasoner, HermiT, and Fact++, among others. During ontology development, a reasoner needs to be invoked to maintain consistency and reasoning can be performed continuously at development time and during ontology run time. For very large ontologies such as SNOMED-CT ontology containing over $400,000$ clinical concepts [31], reasoning is crucial at the design and development time to ensure correct and consistent classification. Also during query evaluations at run time, reasoning is needed to ensure correct inference of relationships between concepts and also to support rules execution.

2.2.3.2 The Language Expressiveness of OWL-DL

In DL-based languages, factual knowledge during individual assertions and terminology definitions are stored as formulas in First Order Logic (FOL). However, restrictions are usually attached to these formulas to ensure decidability and efficient reasoning over the ontology they represent [32]. These restrictions also specify the degree of expressiveness of the DL as compared to FOL, which can express almost everything, though is undecided in terms of computational complexities such as space and time.

To achieve decidability, OWL ontologies are generally restricted to their corresponding DL expressiveness algorithms and the OWL-DL has the expressive equivalence of a $\mathcal{SHOIN}(\mathcal{D})$ algorithm. The basic OWL-DL restrictions, represented by these letter-symbol keys are described below:

- $\mathcal{S}$ — An abbreviation of an Attributive (Concept) Language with Complement $(\mathcal{ALC})$ extended with transitive roles. The Attributive Language $(\mathcal{AL})$ is the basic DL language that allows the use of Concept intersection $(\cap)$, universal restrictions $(\forall)$, limited existential quantification $(\exists)$ and atomic negation of concepts $(\neg)$, which do not appear on the left-hand-side of axioms.

  The $\mathcal{ALC}$ or short form $\mathcal{S}$, is obtained when $\mathcal{AL}$ is extended with the full concept negation, i.e. complement of any concept, not only atomic concepts, can be expressed in the $\mathcal{ALC}$ such as for example, the Top concept $(\top \equiv C \cup \neg C)$ and Floor concept $(\bot \equiv C \cap \neg C)$, where C is any concept.

In owl, the Top concept is called 'Thing' (owl:Thing) and all classes are subclasses of 'Thing', while the floor concept is considered as 'Nothing'.

- $\mathcal{H}$ — An abbreviation of $\mathcal{ALC}$ extended with the role hierarchy (*owl:subPropertyOf* relationship).
- $\mathcal{O}$ — An abbreviation of $\mathcal{ALC}$ extended with Nominals (enumerated classes e.g. *owl:oneOf* or object value restrictions such as *owl:hasValue* relationship).
- $\mathcal{I}$ — An abbreviation of $\mathcal{ALC}$ extended with Inverse roles or properties, which allow expressing relationships in opposite directions (e.g. *owl:hasPart* and *owl:isPartOf*).
- $\mathcal{N}$ — An abbreviation of $\mathcal{ALC}$ extended with a Number or cardinality restriction. Semantics: $\geq$ n R.C or $\leq$ n R.C, where C is domain concept, n is the cardinality.
- $\mathcal{D}$ — The data values expressiveness ($\mathcal{D}$), which is sometimes attached to the algorithm as subscript ($\mathcal{SHOIN}_{(\mathcal{D})}$), denotes the abilities of DL and its family of languages, to use data values, datatype, and datatype properties to further express domain facts.

2.3 Ontologies as Knowledge Management Tools

Information Science and knowledge management practices, involves the development of tools and techniques for acquisition, representation, usage, preservation, as well as evolution of human knowledge. In order to use existing information to create knowledge, knowledge engineers need to understand and generate semantic relationships that are bound to exist between various terms, keywords and facets of domain knowledge. This can be made easier through the use of ontologies, which provides an explicit specification of terms, keywords or concepts in a given domain. These semantics, added to keywords using an ontology language, are human readable as well as machine process-able and thus gives an edge to using ontologies as tools for knowledge management. Moreover, in the article presented in [33], which explore the roles of biomedical ontologies in knowledge management and data integration, the author further highlights the roles of ontologies in knowledge management systems (KMS) to include: *annotation or indexing of resources, retrieval of data and information, data exchange and integration, providing semantic interoperability among domain concepts*, as well as *knowledge discovery*.

While biomedical ontologies are specifically mentioned in his article for domain referencing, these roles are without doubt applicable to other domain ontologies, including our underutilized crops domain. For example, we developed the UC-ONTO as domain ontology for underutilized crops that provide semantic relationships between crops-related terms and further integrate them with other crop data sources in RDFS to enhance information retrieval through a semantic search engine called the Onto-CropBase. Detailed description of the Onto-CropBase tool, including the comparable study semantic vs keyword-based searching, is presented in [11] and discussed here in Section 4.3.

2.3.1 Description Logics (DL) Based Knowledge Representation

The logical formalism behind ontological knowledge representation is known as description logic (DL). Being a decidable fragment of first-order predicate logic (FOL), DL is a collection of logic based knowledge representation formalisms designed for precise description and reasoning about the concepts in an application domain and the relations between them. DL uses logical symbols (operators, quantifiers, equalities, etc.) and variables combined with signatures of non-logical symbols such as unary predicates — which defines the domain concepts as Classes and the binary predicates — which define the number of individuals in a given class or the Roles and their multiplicity — which describes the relations between concepts. Typically, a DL axiom consists of these atomic concepts and individuals to represent complex knowledge of an application domain. As an example, consider a simple DL ontology described using axioms (2.1)-(2.3) that intuitively expresses the domain facts: ”Bambaragroundnut is a Legume Crop which has Leaf, Stem and Root as part of its Features and Leaf-spot is a disease of Bambara-

Groundnut".

$$BambaraGroundnut \sqsubseteq Crop \sqcap \exists isPartOf \cdot (Legumes \sqcap (\exists hasFeatures \cdot Features \sqcap \forall features \cdot (Leaf \sqcup Stem \sqcup Root))) \tag{2.1}$$

$$LeafSpot \equiv Disease \sqcap \exists affects \cdot Leaf \tag{2.2}$$

$$BambaraGroundnut(BambaraGroundnutInd) \tag{2.3}$$

The ability to efficiently model a knowledge domain and the decidable computational characteristics, various ontology languages have been influenced by the DL syntax and semantics leading to the evolution of traditional ontology languages, such as the Ontolingua, OKBC and F-Logic, among others [34]. With the introduction of the Semantic Web however, it became imperative to have a web ontology language with higher expressiveness. As mentioned in Section 2.2.2, this effort that leads to the serial development of various ontology markup languages such as SHOE, RDF, RDFS, DAML and OIL, OWL 1 and more recently the OWL 2 language profiles.

DL-based knowledge representation systems usually involves two important components called the T-Box and A-Box. The 'T-Box' or terminology-box contains the ontology concepts (owl:Classes) and Roles (owl:Properties) also called the terminologies. While the 'A-Box' or assertion-box contains assertions of individual instances from the ontology terms. Example axioms in the T-Box could be the DL axioms (2.1) and (2.2) defined above in the simple Crop ontology, while a member of A-Box could be the third axiom (2.3) which asserts the individual "BambaraGroundnutInd" into the ontology as a member of the "BambaraGroundnut" class.

2.3.2 Classification of Ontologies

Ranging from generic taxonomies to specific application-level knowledge models, Ontologies can basically be categorized into three [35, 36] namely: (i) The foundational ontologies, (ii) Domain ontologies and (iii) Application or Task-level ontologies.

Foundational Ontologies also called 'Upper-level' or 'Reference' ontologies, explain generic concepts and provide general taxonomies with multi-domain knowledge. The Unified Foundational Ontology (UFO) [37], Basic Formal Ontology (BFO) [38], General Formal Ontology (GFO) [39], and the GFO-Bio [40] among others, are common examples of foundational ontologies. Foundational ontology being a repository of general knowledge provides a means for semantic evaluation of lower ontologies such as the domain ontologies.

Domain ontologies on their part provide conceptual and more descriptive definition of terms within scoped domain boundaries, usually for an organization or knowledge community. They usually comprise of domain concepts, their relationships and individual instances. They offer a common vocabulary for sharing, reuse and standardizing knowledge of a specific community or domain of discourse. Larger domain ontologies are sometimes referred as upper-domain, such as BIOTOP [41], which is an upper-domain ontology for molecular biology linking smaller domain ontologies with the BFO, FAOs AGROVOC [42, 43, 44], which has in the past thirty years grown from simple multilingual agricultural index to a Linked-Open-Data (LOD) set. Other example domain ontologies includes the Crop Ontology [4], Plant ontology [3], Gene Ontology [2]. For the purpose of our onto-cropbase tool [11], a domain-level ontology UC-ONTO [45] describing the underutilized crops and their farming practices, was developed to serve as knowledge base.

Application ontologies describe activities or specific application's vocabulary in a domain. Developed to be used for specific applications, application ontologies usually utilize the domain ontologies by restricting its conceptualizations to model a specified application or task. For example, the Food Ontologies for nutritional applications in [46, 47, 48] and 'sensor ontologies' for manufacturing application reviewed in [49]. The availability and popularity of standardized domain-level ontologies greatly affect the development of application-level ontologies, reducing the process in most cases, to a simple task of narrowing down existing domain ontologies into task-specific ontologies. However, new domain/user-specific concepts can be easily generated from competency questions and user queries, when involved in the knowledge generation process. These newly-generated concepts may need to be harmonized with the existing ones, leading to the Ontology standardization process — see Section 2.5.4.

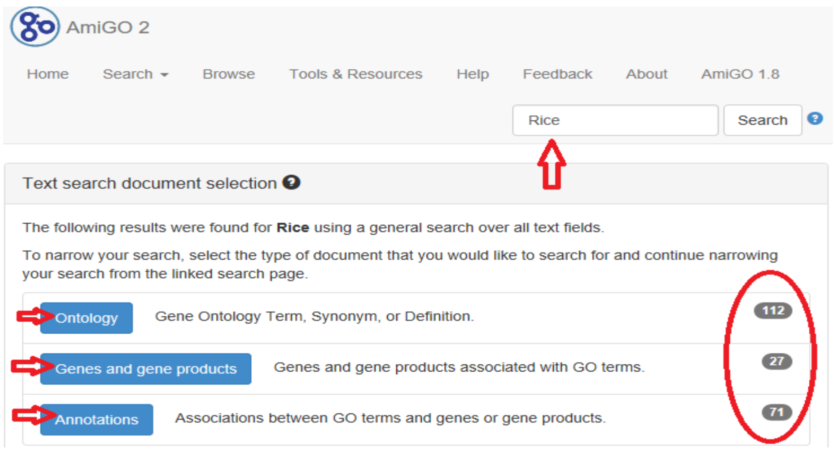
In the subsequent subsections, we briefly discuss some of the common approaches of generating domain ontologies. These approaches, depending on the nature of the ontology development and knowledge engineers involved may be combined together or individually employed at different stages of the ontology development.

2.4 Ontologies in the Crops Domain

As stated earlier, information on underutilized crops is usually dispersed among different resources: research papers, implicit knowledge, from the domain experts at CFF, etc. However, as common terminologies for the crop domain already exist in the literature, we model our ontology based on those standard terms obtained from popular agricultural ontologies such as the AGROVOC, Plant and Crop Ontologies among others. This section discusses the life-sciences domain ontologies with emphasis on the crops domain. We present the popular crop-related ontologies highlighting the expressiveness provided by their development language and showing their inadequacy in representing underutilized crops knowledge. Subsequently, we highlight the benefits of ontologies in crops knowledge modeling.

2.4.1 The Gene Ontology

The Gene Ontology [2] is a popular biological upper-domain ontology developed by the Gene Ontology Consortium to establish standards in the representation of gene-related knowledge for various species of organisms. It is designed as a collaborative community-based ontology development effort providing gene ontologies with three components: molecular functions, biological processes and cellular components, their annotations as well as tools to access and process the ontologies [50]. Like many existing biological ontologies, the Gene Ontology is developed using the java-based open-source OBO-Edit environment and therefore available mostly in the OBO format . OWL versions of these ontologies are provided though. However, as explained earlier, OBO ontologies even when converted to OWL formats, lacks the expressiveness provided by OWL.



Text search document selection

The following results were found for **Rice** using a general search over all text fields.

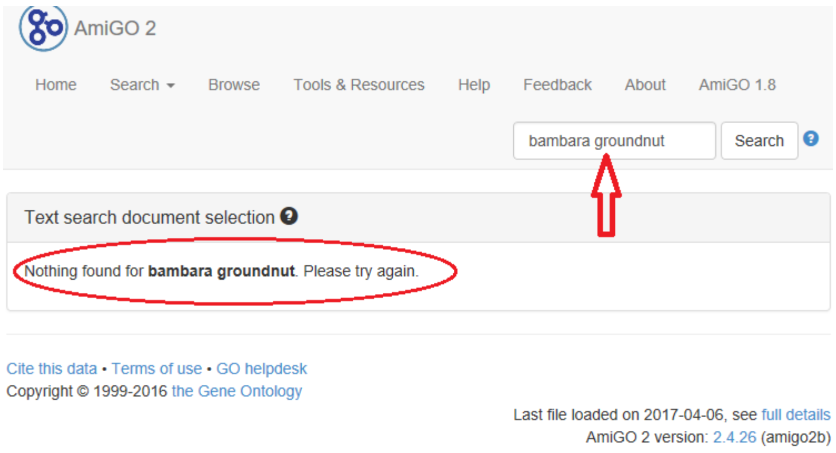
To narrow your search, select the type of document that you would like to search for and continue narrowing your search from the linked search page.

Ontology — Gene Ontology Term, Synonym, or Definition. — 112

Genes and gene products — Genes and gene products associated with GO terms. — 27

Annotations — Associations between GO terms and genes or gene products. — 71

Figure 2.5: Gene Ontology Search Tool (The AmiGO2) showing information on major crop (Rice). [2]



Text search document selection

Nothing found for **bambara groundnut**. Please try again.

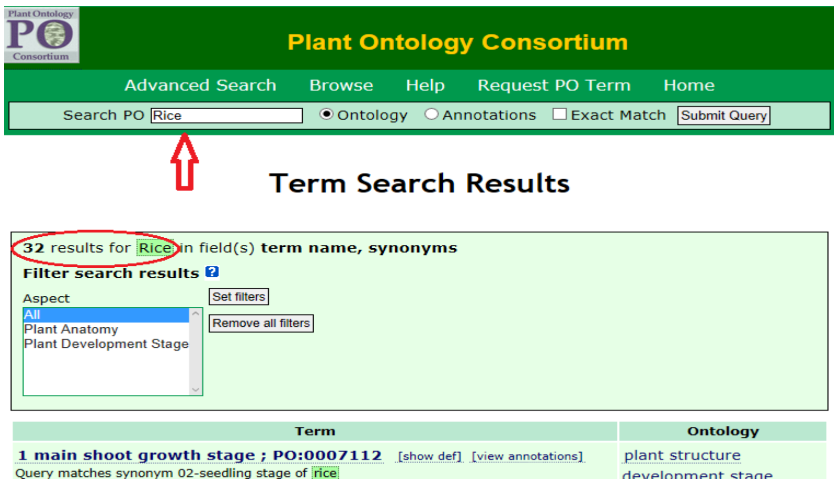
Copyright © 1999-2016 the Gene Ontology

Last file loaded on 2017-04-06, see full details
AmiGO 2 version: 2.4.26 (amigo2b)

Figure 2.6: Gene Ontology Search Tool (The AmiGO2) showing lack of information on underutilized crop (Bambara groundnut). [2]

2.4.2 The Plant Ontology

Considering it as a comparative tool for plant anatomy and genomic analysis [3], the Plant Ontology is developed to provide formal specification of terms that describe plant anatomy, morphology and growth stages with the first and later developed as components of the whole ontology. Plant Ontology utilizes the data model available in the Gene Ontology (GO) [2], for annotating the plant anatomy and growth stage ontologies with gene expressions and phenotype data from the GO. Similar to the Gene Ontology, the Plant ontology is also guided by the OBO Foundry ontology for seamless collaboration with other biological ontologies [3] and most of the ontology is available in the OBO format . However, some parts of the ontology are available in the OWL format. For efficient comparison of disparate data with similar terms, such as that of genomics, the use of ontologies is necessary for data curation and analysis as it helps to provide common structured vocabulary that permits automated reasoning.



Search PO Rice ● Ontology ○ Annotations ☐ Exact Match

Term Search Results

32 results for Rice in field(s) term name, synonyms

Filter search results

Aspect

All
Plant Anatomy
Plant Development Stage

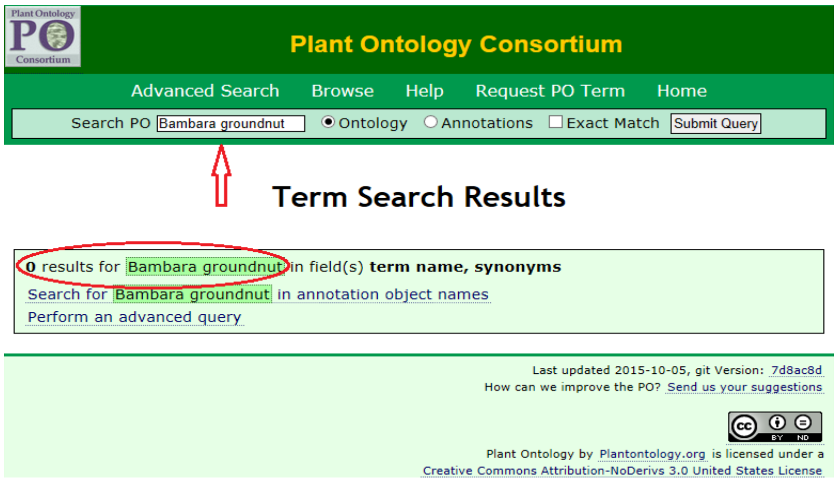
Term	Ontology
1 main shoot growth stage ; PO:0007112 [show def] [view annotations] Query matches synonym 02-seedling stage of rice	plant structure development stage

Figure 2.7: The Plant Ontology Search Tool showing information on major crop (Rice). [3]

2.4.3 The Crop Ontology Curation Tool

Citing data management, accessibility and retrieval challenges as the main motivation, Generation Challenge Program (GCP) [1] developed the Crop Ontology to facilitate community sharing of crop-related information by semantically characterizing and annotating historic generic crop data sets (traits, phenotype, germplasm, breeding, etc.) [51, 4]. With a simple web-based interface and the help of semantic experts as moderators of the ontologies, the Crop Ontology platform allows

[1]http://www.pantheon.generationcp.org


Plant Ontology Consortium

Advanced Search Browse Help Request PO Term Home

Search PO Bambara groundnut [✓ Ontology] Annotations Exact Match Submit Query

Term Search Results

0 results for Bambara groundnut in field(s) **term name, synonyms**
Search for Bambara groundnut in annotation object names
Perform an advanced query

Last updated 2015-10-05, git Version: 7d8ac8d
How can we improve the PO? Send us your suggestions

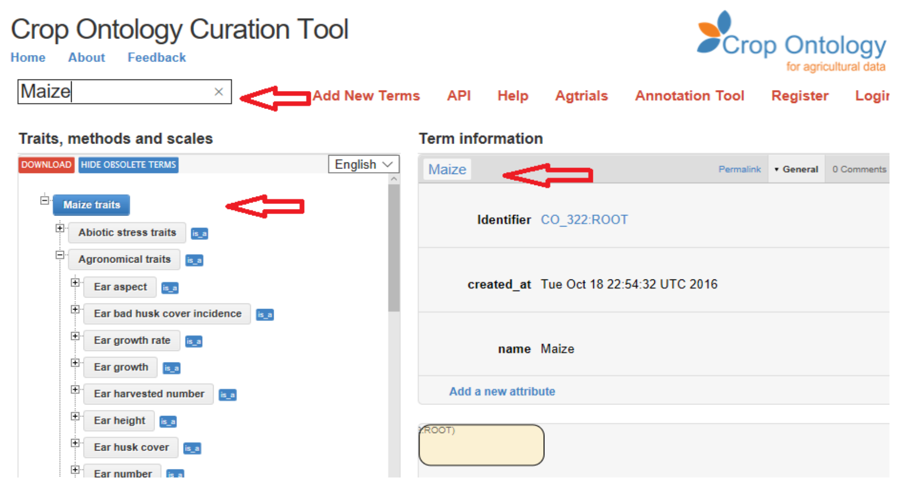
Plant Ontology by Plantontology.org is licensed under a Creative Commons Attribution-NoDerivs 3.0 United States License

Figure 2.8: The Plant Ontology Search Tool showing lack of information on underutilized crop (Bambara groundnut). [3]

community-based collaborative ontology development, where users can create and add their own ontologies to the pool. Originally in Open Biomedical Ontology (OBO) formats, the Crop ontology has evolved to utilize more terminological standards such as RDF and OWL [52].

With OWL being a widely used standard for developing ontologies, the effort to provide Crop Ontologies in RDF and OWL format, will help in knowledge-sharing between among researchers. This is basically due to the high expressive power provided by OWL language constructs, the efficient reasoning support, and the added advantage of using rules to integrate OWL ontologies with declarative languages such as SWRL. Moreover, Semantic Web applications can be developed to utilize OWL ontologies.

From the foregoing exploration, it can be seen that the ontologies are able to provide an efficient and comprehensive hierarchical representation of their domains with common roles between concepts being of the form ’is-a’ and ’part-of’ relationships, which simply put, denotes that a concept is either a subtype of the connecting concept or that of the root/ancestral concept. However, they seem to lack complex representation of roles or relationships between concepts, which is one of the major differences between ontologies and hierarchical taxonomies such as thesauri. In a similar gesture, authors of ’Crop Ontology: vocabulary for crop-related concepts’ in [52], have suggested the use of OWL-DL in their future



Identifier CO_322:ROOT

created_at Tue Oct 18 22:54:32 UTC 2016

name Maize

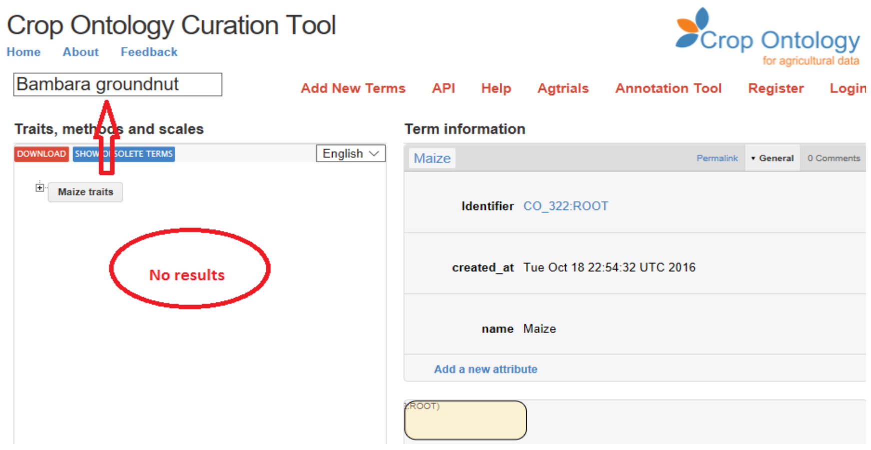
Figure 2.9: The Crop Ontology Curation Tool showing information on major crop (Maize). [4]

works of for added expressiveness and complex domain modeling.

2.4.4 Contributions of Ontologies in Crops Modeling

Based on the analysis of ontology development literature above, the contribution of ontologies to domain knowledge modeling includes the Standardization of domain knowledge, Organization and sharing of domain information, as Integration tools and for domain knowledge comparisons. Other non-direct advantages include the Separation of domain knowledge from operational knowledge. Moreover, domain ontologies are useful for developing semantic applications. These contributions are not exclusive to the crop domain as they are mostly generic advantages brought about by ontologies. A text book detail on benefits of ontologies in domain knowledge modeling, with a focus on Bioinformatics domain, is presented in [53].

More specifically, contribution of ontology to the crops domain can be summarized in the following points: (i) For organization and sharing of crops information and meta-data — leading to standardization (e.g. FAO's AGROVOC, CFF's UC-ONTO, etc.) (ii) For organization and sharing crop information (iii) As integrative comparative tools (iv) Separation of Crops knowledge from Operational knowledge in enterprise applications, and also (v) Useful for developing semantic web


Crop Ontology Curation Tool
Home About Feedback
Crop Ontology
for agricultural data
Add New Terms API Help Agtrials Annotation Tool Register Login
Traits, methods and scales
DOWNLOAD SHOW OBSOLETE TERMS
English
Maize traits
No results
Term information
Maize
Permalink General 0 Comments
Add a new attribute
(ROOT)

Bambara groundnut

Identifier CO_322:ROOT

created_at Tue Oct 18 22:54:32 UTC 2016

name Maize

Figure 2.10: The Crop Ontology Curation Tool showing lack of information on underutilized crops. [4]

applications.

These contributions are however not exclusive to the crop domain as they are mostly generic advantages brought about by ontologies and the semantic web project. Though, as explained earlier, ease of knowledge comparability by ontologies is more pronounced in the field of life sciences. A review on the recent trends and application of ontologies citing examples from other domain ontologies is presented in [54] and a text book detail on the 'uses of ontologies in bio-informatics' is given in [53].

2.5 Ontology Engineering Methodologies

2.5.1 Generating Ontologies from Scratch

Developing ontologies from scratch is usually associated with upper-domain ontologies. This is because, upper-level ontologies being generic taxonomies that are proposed as foundational standards, usually have no ancestral ontologies that can be reused. However, exceptional cases in mid-level (domain and application) ontology development may require similar approach, especially when creating pioneer domain ontologies such as the Underutilized Crop Ontology (UC-ONTO) [45]. However, analogous to software development tasks, there is yet an agreed-upon fool-proof methodology for ontology development. What is certain, though, is that ontology development is *iterative*. The iterative nature of domain ontology development is highlighted by many researchers such as by [55], and [56] — who highlighted that in addition to the repetitive nature, ontology development is also fast becoming a collaborative effort. The collaborative Web Protégé is thence introduced in [56] as a tool to support the collaborative nature of ontology development. Key participants in ontology development efforts may include the knowledge engineers, domain experts and social engineers among others.

In a more holistic approach proposed in [23], ontology development is recommended to begin with the following tasks: (i) Define the scope of the knowledge domain, (ii) Define the purpose of developing the ontology, and (iii) Identify the potential users of the ontology. In essence, determining the expected *size of the ontology* and precisely knowing the *application domain* it will cover, while having its *users* in mind during the entire development process, greatly helps in smoothing the domain ontology development process.

In our opinion however, an ontology implementation must follow the dedicated object modeling approach, where: (i) Knowledge engineers or Ontologist should first identify the classes for the ontology — this can be done through competency questions or simply through identify key domain concepts from the knowledge source, (ii) secondly, arrange the classes in a taxonomic order (i.e. super-class-subclass relationships), (iii) defining the object and data-type properties (relationships), and finally (iv) asserting individual instances as values.

As such, these steps were performed repeatedly for developing each component version of the UC-ONTO, leading to the final complete version. Two final stages were termed *versioning* and *assembly*. In (v) Versioning, we assign a label to represent each ontology fragment, specifying where it fits to the larger ontology. While in the (vi) Assembly stage, all the smaller ontologies are put together and the reasoner is invoked to assert the overall classification and check for consistency. The complete UC-ONTO development methodology is summarized in Section 3.2.

2.5.2 Ontology Generation from Heterogeneous Data Sources

Domain knowledge typically exists in different sources and formats. These sources, which usually contain the domain knowledge as well as the meta-knowledge, can sometimes be used as knowledge sources feeding into the ontology development process. In this subsection, we discuss the semi-automated approach to generating ontologies from external data-sources such as the structured sources e.g. relational databases (RDBs), semi-structured sources e.g. Extensible Mark-up Language (XML), and unstructured knowledge existing as texts. Automatic generation of ontologies from data sources is still an ongoing research trend and in order to enhance knowledge reuse and smooth the migration from the current web of documents to the Semantic Web of data, ontology development has to address the challenges of heterogeneous data sources. Various researchers have proposed tools and techniques for generating ontologies from heterogeneous data sources such as relational database to OWL (RDB2OWL) and XML to OWL (XML2OWL) add-ins in Protégé, among others [57, 58, 59, 60].

For the development of the Underutilized crops ontology (UC-ONTO), conceptualization approaches were proposed or tested and eventually utilized to generate the pioneer OWL2-based Crops ontology. These conceptualization approaches include: Concept generation from structured and unstructured data such as XML, RDF, PDF, Word files, Excel sheets and research notebooks.

2.5.2.1 Generating OWL Concepts from XML and RDB Files

The 'XML to OWL' tab in Protégé ontology editor allows generating OWL concepts directly from an XML document by converting every data root in the XML tree into a class and resulting list in the root as their individual instances. Where as for generating ontologies from relational tables an 'RDB2OWL' [58] Protégé plug-in may be employed. The relational data to OWL mapping is achieved through an XML2OWL as intermediary. In essense, the plug-in generates OWL ontologies by converting tables into XML trees and then using the XML to OWL conversion to complete the process. Relationships between concepts are defined by the table meta-data.

Generating OWL ontology from an XML document in Protégé involves loading the XML instance document into the XML2OWL converter and then generating the OWL concepts into existing ontology using the 'import' tab — see Fig. 4.2. However, in order to avoid inconsistencies and cluttering of existing ontologies, an empty OWL-DL ontology file needs to be created before loading the XML file and the XML-Tab should only be created in the active ontology that one intends to accommodate the XML instances. This is because once the 'import' button is clicked, Protg directly append the instances generated from the XML documents to the active ontology. Moreover, as there is an option to inspect the XML tree before the importation, careful inspection needs to be done to ensure correct XML documents are used — the process is irreversible.

However, while these approaches help to easily generate OWL classes, properties and in some cases instances, the resulting ontologies are usually inconsistent with few to no-relationships generated among concepts. Moreover, while these tools deals with the heterogeneity at initial stage of the ontology development, inconsistent ontologies can be managed through ontology matching, alignment and merging — discussed in Section 2.5.4 ontology standardization approaches. We present a detailed discussion on dealing with heterogeneous ontologies and data sources in [61].

2.5.3 Ontology Reuse: Generating an Ontology from Ontologies

As suggested in the 5-star scheme of the Semantic Web [62], it is always advisable during ontology development to 'reuse' existing ontologies. This can be achieved by 'importing' relevant ontologies to use them as-is or 'adopt' some of their specific domain features. Adoption is usually applied when importing upper-domain ontologies that needs to be extended with more specific concepts and terminologies. We discuss the process of ontology import using the Protégé ontology editor in Section 3.3.7 of the Methodology Chapter. Importing ontologies does not only simplifies and speedup the ontology development process but also helps to ensure the use of standard vocabularies for a given knowledge domain or community of discourse. For example, in the field of agriculture, where various upper-level ontologies exist (cf. Section 2.4), the ontology reuse approach is commonly applicable. The ontology-reuse approach no doubt saves a great amount of development time on the part of knowledge engineers and helps in generating richer as well as standardized vocabularies.

2.5.4 Ontology Standardization Approaches

A notable stage in the ontology development process is the standardization of the concepts and roles defined in the ontology. Domain ontology standardization typically involves merging, matching and alignment of (imported) ontologies and the use of foundational ontologies to streamline concept definitions. Other forms of ontology standardization approaches involve domain expert's validation of the model as well as providing natural language annotations — for ambiguous concepts or terminologies. An effective ontology standardization results in general understanding and acceptance of the resulting ontologies — leading to further reuse. It also helps to minimize inconsistencies in nomenclature across the knowledge domain. In the following subsections, we briefly highlight the common ontology standardization approaches.

2.5.4.1 Matching, Alignment and Merging of Ontology Fragments

Ontology development typically involves reusing other ontologies and to this reuse and the iterative nature of ontology development, there is a need for continuous alignment, matching and ultimately merging two or more ontologies. Ontology alignment is commonly associated and sometimes even confused, with ontology matching. In [63], the authors clearly differentiate the terms as follows:

Ontology Matching involves examining two or more ontologies with the aim of finding relationships, similarities or correspondence between them. The result of ontology matching is the *Alignment* of the respective Ontologies, where semantic relationships between concepts in the two ontologies are identified. While Matching is simply to find correspondences, *Ontology Merging* on the other hand is action oriented as it involves creating a new ontology from two or more overlapping ontologies without changing the sources. Another concept worthy of note is *Ontology Mapping*, which is a directed matching of two or more ontologies, where entities of one ontology are mapped to at-most one entity of another ontology.

Tools for ontology merging (by extension matching and alignment) exist, such as Protégé's PROMPT [64] which basically use bridge axioms (also called articulation axioms) to integrate entities of one ontology into the entities of another. Similar to schema integration of databases, ontology merging uses the bridge axioms to find similar concepts basis of ontology merging. Due to its clearly defined user interface, its step-by-step informative feedback during the process, the merging tool available in recent version of Protégé is often used for resolving terminological conflicts and ontological differences — including in the work presented in this thesis.

Concepts Alignment with Foundational Ontologies Upper-level ontologies, being repositories of a more general basic knowledge, provides a means for semantic evaluation of lower ontologies such as the domain and application-level ontologies. Moreover, being standards themselves, upper-level or foundational ontologies can be employed to enhance standard concept definitions for lower domain ontologies. Upper-level ontologies commonly consist of general taxonomies describing multi-domain knowledge. Examples of foundational ontologies, include: the Unified Foundational Ontology (UFO) [37], Basic Formal Ontology (BFO) [38], General Formal Ontology (GFO) [39], and the GFO-Bio [40], among others. Larger domain ontologies such as BIOTOP [41], which is upper-domain ontology for molecular biology linking smaller domain ontologies with upper-level ontologies such as the BFO, can also be employed to standardize the conceptual definitions of smaller domain-level ontologies. The standardization can be achieved through imports of relevant sections of the foundational ontology or by simply aligning the concept definitions in the domain ontology with that of the upper-level ontologies, so that similar terminologies and concept definitions appears in both the domain and foundational ontology. Where the fragment of the foundational ontology is available, it can be *merged* with existing domain ontologies or used as a foundation for the development of new domain ontology.

2.6 Ontology Development Environments (ODEs) — An Overview

2.6.1 The Protégé Ontology Development Environment

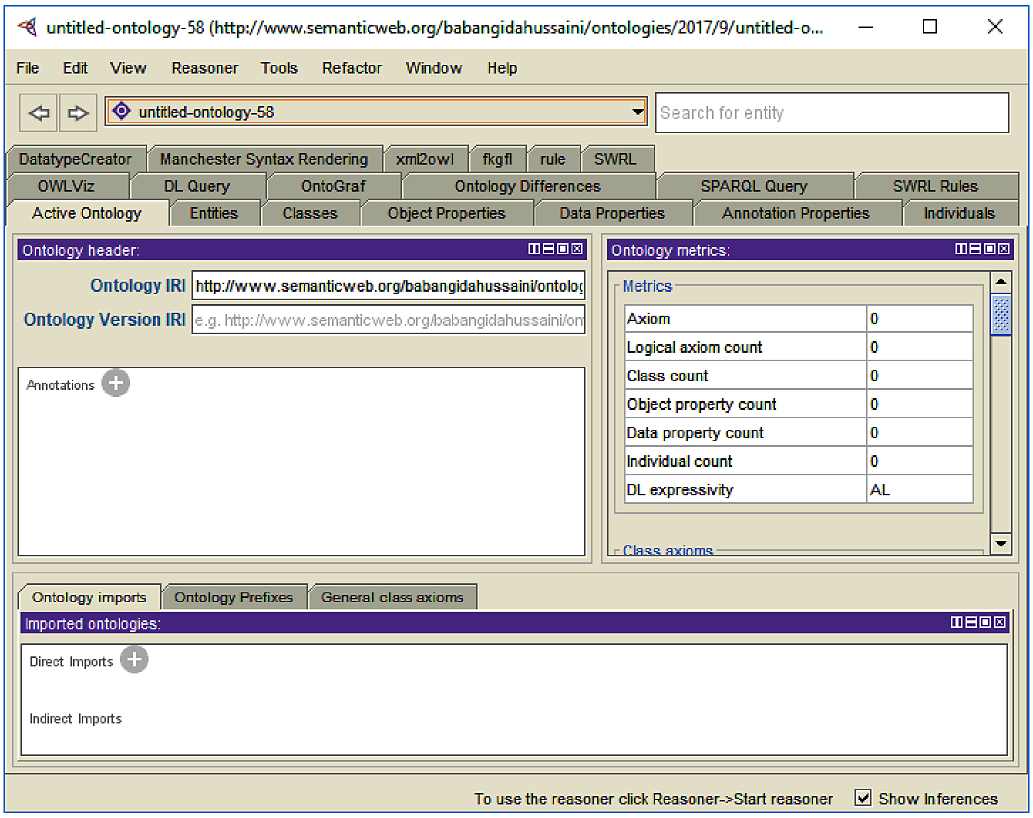
Protégé ontology-based knowledge development environment provide full support for the OWL 2 profile and the Semantic Web Rule Language (SWRL). It has a flexible and user-defined interface arranged in tabs and movable widgets, as shown in the Fig. 2.11 below. More importantly it provides support for user-defined rules using the ’Rule’ plug-in and rules tab, where users can write OWL 2 RL and SWRL rules for manipulating ontologies with Rule-based Reasoners (such as Pellet and Hermit) that can reason over the rules without having to use a rule-based engine. Figure 2.11 shows the ’Active Ontology’ interface showing the Ontology IRI, version IRI, language, the Ontology Imports tab, the Ontology Metrics tab, etc.

Other tabs, can be created using the Windows views menu. For ease of ontology development, various tabs have been added to the Protégé ODE such as the Individuals tab, the OWL Viz and OntoGraf for ontology visualization and the Ontology Differences tab, for automatic identification of differences between two or more loaded ontologies. Others are the DL and SPARQL query tabs, the SWRL rules tab and the Datatype creator tab — as shown in the second layer of the tab menu. The first layer however, shows the default tabs (Entities, Classes, Object and Datatype Properties) that are inherent in the Protégé installation. figure 2.11

2.6.2 The TopBraid Composer Integrated Development Environment

The TopBraid Composer (TBC) is another W3C standards compliant ODE for developing, managing and testing ontological knowledge models. Developed by TopQuadrant[2], TBC is an enterprise-class development environment for developing Semantic Web ontologies and building semantic applications. It is integrated with the AllegroGraph plug-in to provide a scalable RDF backend. With a pub-

[2]https://www.topquadrant.com/


untitled-ontology-58 (http://www.semanticweb.org/babangidahussaini/ontologies/2017/9/untitled-o...

File Edit View Reasoner Tools Refactor Window Help

untitled-ontology-58

Search for entity

DatatypeCreator | Manchester Syntax Rendering | xml2owl | fkgfl | rule | SWRL
OWLViz | DL Query | OntoGraf | Ontology Differences | SPARQL Query | SWRL Rules
Active Ontology | Entities | Classes | Object Properties | Data Properties | Annotation Properties | Individuals

Ontology header:

Ontology IRI: http://www.semanticweb.org/babangidahussaini/ontolog
Ontology Version IRI: e.g. http://www.semanticweb.org/babangidahussaini/on

Annotations

Ontology metrics:

Metrics

Axiom	0
Logical axiom count	0
Class count	0
Object property count	0
Data property count	0
Individual count	0
DL expressivity	AL

Class axioms

Ontology imports | Ontology Prefixes | General class axioms

Imported ontologies:

Direct Imports

Indirect Imports

To use the reasoner click Reasoner->Start reasoner ☑ Show Inferences

Figure 2.11: Protégé Ontology Editor Interface showing Ontology summary page.

lishable API for developing web-based applications, the Top Braid Composer can integrate various knowledge models and data sources into a consistent semantic web application.

As shown in Fig. 2.12, the TBC main interface also provides the Class, Properties and Statement views with the Ontology Statistics view as an equivalent of the Ontology Metrics of the Protégé ODE. It also provides the Ontology imports view, the Instances view, Domain and Relevant views of selected concepts. These and the fact that the TBC also provides a drag-and-drop semantic web application development framework, makes it much simpler and flexible to work with and hence suitable even to non-technical experts in developing domain ontologies and semantic web application.

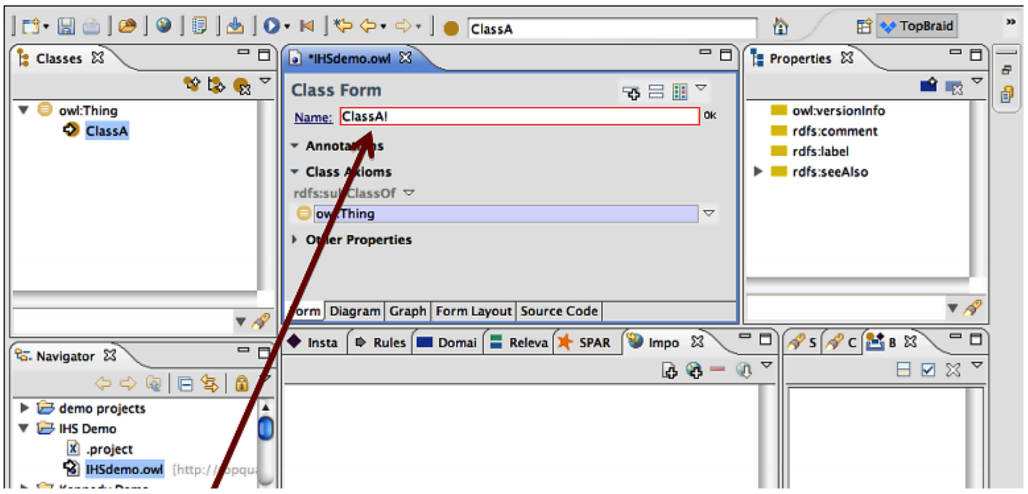


Figure 2.12: Top Braid Composer Ontology Editor showing Ontology Statistics page. [Image source: http://semanticommunity.info]

Cost: While the Protégé ODE is free and open-source tool with a rich community support for developers, the TBC ODE is licensed with a with a cost of licensing options ranging from 500 to 3500 USD. As such for a social enterprise application and not for profit projects such as the Underutilized Crops ontology development, the Protégé becomes an optimum choice for the ontology development tool.

Flexibility and Automation: However, the TBC[3] provides some value for money as it helps to automate some of the complex ontology development processes. Moreover, the TBC makes developing semantic web applications to utilize the developed ontological knowledge base easier, as it provides an integrated development environment with simulated web browser capabilities. Unlike the Protege ODE where separate API is needed to connect between the ontology data sources, linked-data, and web services, the TBC allows the complete process within a single IDE with rich set of libraries and GUI-based application development tools. Nevertheless, the open-source based Protege is still a better choice for developers due to the available API and large community of users for support.

[3]https://www.topquadrant.com/tools/ide-topbraid-composer-maestro-edition/

2.7 Ontology Evaluation

Ontology evaluation is an important step in the ontology development process which concerns the assessment of a given ontology based on identified criteria. This is particularly useful where ontology reuse — an important and highly encouraged approach, is advocated in order to identify whether a given ontology has fit to be reused for a particular purpose or application domain. However, despite its importance, there is yet an agreed-upon qualitative or quantitative approach for assessing the quality of ontologies. Nevertheless, there are several methodologies proposed by researchers over time. Common ontology evaluation techniques and their goals have been categorized in [65] to include the following: Vocabulary evaluation, Hierarchy or taxonomy evaluation, Contextual evaluation, Application-based evaluation, and Data-driven evaluation techniques. In all these cases, the ontology is evaluated based on the initial requirements of the ontology model and hence an efficient competency question document would help to guide the evaluation process.

In their quest for a qualitative approach to ontology evaluation, the authors of [66] proposed a Peer-review based ontology evaluation methodology. This approach allows non-authors of ontologies to provide some sort of qualitative ratings for ontology contents with a view to selecting the best fit for an application domain. A framework was then proposed to develop an ontology of metedata elements that can be used to measure the quality of ontology features. While this approach seems typical and simple, it is however difficult to provide peer-reviews on ontologies that are of particular knowledge domain. This does not only requires domain experts but requires Ontologists that also double as domain experts in order to efficiently evaluate the quality of contents as well as the conceptualization in domain ontologies. Moreover, as ontology reuse is a recommendation, not a requirement to ontology development, this will make the process of ontology engineering more difficult and tedious than it already is.

Another qualitative approached highlighted in [67] is termed Full Ontology Evaluation Approach (FOEVal), which can be employed to evaluate both local as well as searched ontologies. The main approach of the FOEval method is to utilize certain *Metrics* that can be used to evaluate an ontology based on its purpose, users and domain coverage. Example metrics includes the completeness of ontologies, comprehensiveness, as well as the level of details, among others. An overall score is then computed for the ontology based on the weighted average for each metrics. Other notable evaluation approach to ontology evaluation is the User-centered evaluation approach [23, 66, 68], where intended users as well as domain experts of the ontology specialization are employed to validate the contents, structure and vocabulary of an ontology. This approach was tested in [68] where the authors obtained expert opinions on the validity of the domain ontologies. Further validation employs the users of the ontologies to measure the ontological commitment based on the user satisfaction.

From the evaluation approaches explored, a more comprehensive and flexible evaluation method to be proposed and adopted is the use of competency questions to evaluate ontologies.

Ontology Evaluation using Competency Questions. While competency questions are highly relevant in determining the goal and scope of ontologies, they are also equally useful during the evaluation process [23]. This is because, the questions identified during the ontology requirements stage can easily serve as a good starting point to plan the test cases. Furthermore, the answers elicited from domain experts would not only help to validate the vocabularies but also highlights the taxonomy as well as content accuracy. As such, these competency questions can be converted into competency queries to the completed ontology model. These can be easily asserted through DL-queries and SPARQL queries to check the precision and recall of the information provided by an ontological knowledge model. Where the query results do not match the answers to the competency questions, the ontology can thus be modified or extended to provide the desired validation metrics. However, one of the challenges to adopting CQs for ontology evaluation, as highlighted in [69], is the lack of supporting tools for competency questions management. The authors then proposed a ”CQ Checker’, a java-based model to support the requirement specification phase of ontology development using CQs.

The detailed methodology for Competency questions management are described in Section 3.2.1 and the use of Competency questions for evaluating the Underutilized Crops Ontology (UC-ONTO) is presented in Section 5.2.3

2.8 Logic Programming (LP) Rules in the Semantic Web

In the field of AI, extending knowledge bases with rules falls under the impress of hybrid knowledge bases (hybrid-KBs), where a domain conceptual model (ontology) and a rule-base are combined to form the knowledge-base of a given application system. Understanding such hybrid-KBs depends considerably on understanding the semantics of its component ontology and rule languages. Usually, a single interpretation is possible, where similar semantics are involved. However, where the combination involves different semantics, separate interpretations are required for the ontology and the rule base. Moreover, understanding these differences helps knowledge engineers to effectively decide when choosing a compatible rule formalism that can be used to extend a given ontology model.

Despite its success in achieving hierarchical definition and efficient classification of domain concepts when compared to its predecessor the RDF, OWL itself suffers expressive limitations, such as its lack of support for composite role definition between concepts. Hence, there is the need for a more expressive domain modeling language than OWL as established by various researchers citing both theoretical and practical examples [70, 71, 72, 73, 74, 75]. Rule formalisms were consequently adopted to provide the needed support for more expressive power to the OWL language both being fragments of the classical logic.

In the semantic web project, integration of OWL-DL and SWRL provides many advantages that cannot be achieved using either OWL DL or Horn rules alone. Moreover, extending ontologies with rules is favored due to the wide acceptance of rules in knowledge modeling and the success of Rule-based formalisms in commercial applications among others. The expressive limitations of OWL and the choice for Rules are not just mere coincidences. While OWL-DL ontologies provides simple, reusable and easy to understand knowledge models, they lack the expressiveness offered by rules. Furthermore, the rule formalisms apart from being in common practice, provides an efficient reasoning support to ontologies with the added expressiveness.

The LP rules are syntactically $if-then$ statements consisting of an antecedent and consequent — also called the rule body (B) and head (H) respectively. They are usually of the following form:

$$H \longleftarrow B_{1+}, ..., B_{n+}, notB_{1-}, ..., notB_{k-} \tag{2.4}$$

And the semantics ensures that the condition in the rule's head be evaluated to true whenever the conditions in the body are satisfied. However, there are variants to the basic format above such as rules with disjunctions in their heads or those extended to handle classical negations, etc. We discuss more on rule expressiveness extensions focusing on the SWRL formalism in the next chapter.

2.8.1 Need for LP Rules in Ontologies — Motivations for Augmenting Ontologies with LP Rules

As established from the reviewed literature [76, 77, 78, 79], comprehensive domain modeling using ontologies usually requires integration with user-defined rules. This may be due to advantages offered by the rule formalisms such as their wide acceptance, successful commercialization of rule-based applications, and the available reasoning tools to provide inferences. In the Semantic Web community, the use of rules to integrate ontologies is rooted in the blueprint of the semantic web stack. Comparable to ontologies, logic programming (LP) rules on the Semantic Web are designed to help with data integration. Rules are particularly important where an extra logical assertion may lead to a discovery of new relationships between concepts. As such, for a comprehensive domain modeling, there is, therefore, a need for augmenting ontologies with the declarative expressiveness of logic programming rules. An example case study highlighting our experience in ontology extension with the Semantic Web Rule Language (SWRL) is presented in Section 4.2.7.

In essence, the expressive limitation of OWL and the choice for Rules is not a mere coincidence. While OWL-DL ontologies provide simple, reusable and easy to understand knowledge models, they lack the expressiveness offered by rules. Furthermore, the rule formalisms apart from being in common practice, also provide an efficient reasoning support to ontologies with an added expressiveness [80]. The following advantages followed by suitable examples from our case study shows how SWRL rules can enhance the limited expressive powers of OWL ontologies:

- Rules allow declarative Assertions: Due to the declarative nature of rules axioms, they can be used to easily express complex domain concepts and assert solutions to hard problems.
- Logic and Data Separation: Using rules allow the separation of data or facts about a domain (ontology) from the logic (rules). This aid in change management as business logic not concepts usually changes.
- Tools Integration: Various tools that support rule-based development and reasoning exist. Example includes: CLISP, Jena, Protégé, etc.
- Built-ins and user-defined built-in libraries: Domain-specific rule assertions complements ontologies and increase the expressive powers of a reasoning system. The ability to incorporate standard as well as user-defined built-ins in SWRL provides added domain-specific expressiveness and the type of reasoning information that can be achieved. For example, if *Days After Sowing (DAS)* is greater than 50 then "flowering" is expected. Here, *greater than 50* is an integer constraint called datatype predicates — defined as a pseudo-built-in in SWRL.

$$\begin{gathered} Bambaranut(?b), DAS(?d), GrowthStage(?g), Flowering(?f), \\ swrlb : greaterThan(?d, 50) \longrightarrow GrowthStage(?g, "flowering") \end{gathered} \tag{2.5}$$

- Expressing Domain Properties: In addition to ontology relations, other non-explicit domain relations that do not appear directly from concepts classification can be specified using rules. Common examples where rules are needed is when expressing dependencies between relationships or properties such as expressing dependencies between two or more ontology properties. Example, the concept "if two crops share a similar cultivation region, then the two crops can be planted together" can be modeled easily with a SWRL rule as follows:

$$\begin{aligned} &Bambaranut(?b), Millet(?m), CultivationRegion(?cr),\\ &\quad hasCultivationRegion(?b, ?cr), hasCultivationRegion(?m, ?cr)\\ &\qquad \longrightarrow canPlantTogether(?b, ?m) \end{aligned} \tag{2.6}$$

- Expressing Queries: LP rules are also useful in modeling user queries. Note that queries are usually rules written without a value in the consequent. Since our work includes query-answering, such as: find all possible instances of a certain Crop type in a given geographical region. This can easily be written as a SWRL rule with no consequence and the results of such rules can be displayed as automated queries. An example:

  $$Crop(?crop),\ hasCultivationRegion(?crop, ?region) \tag{2.7}$$

  The above rule (2.7) will return all crops stored in the ontology and their cultivation region properties.

In what follows, Section 2.8.2 reviews the various Logic Programming (LP) formalisms for extending DL-based ontologies focusing on those formalisms that are compatible with OWL — the modeling language of our underutilized crops ontology (UC-ONTO) [45]. Furthermore, an evaluation is provided of their added expressiveness and their required conditions for decidability. For completeness, the motivations and common concerns for integrating ontologies and rules were discussed.

2.8.2 Common Rule Formalisms for Extending Ontologies

Due to the nature of syntactic or semantic integration between components, formalisms for integrating rules with ontologies can be categorized as either hybrid (loose integration) or homogeneous (tight integration) approaches [62]. In the homogeneous approach, common syntax and/or semantics are shared by both the rule and ontology component. For example, SWRL has a uniform model theoretic semantics with its component OWL axioms and is therefore homogeneous Other homogeneous formalisms include: DL+Log [81], KAON 2 [82], CLASSIC [83], CARIN [84], extended-CARIN and r-hybrid KBs [20].

In the hybrid approaches however, the semantics of the rule language is considerably different from that of the ontology language. In such cases, an interface is sometimes needed for exchanging knowledge between the ontology and rules, such as the use of Answer Set Programming (ASP) in ’dl-programs’ [85, 86]. Other hybrid approaches include the ’AL-Log’ formalism [87, 88], disjunctive Al-Log and disjunctive dl-programs. Though other formalisms exist in which the ontology and rules component of the KB uses a single vocabulary with no explicit separation such as in the ”logic of Minimal Knowledge and negation as Failure” or simply put, the Hybrid MKNF [87]. Some of the formalisms were briefly described below.

2.8.2.1 The SWRL Formalism

The semantic web rule language (SWRL) [89] is a W3C recommendation that extends the Web Ontology Language (OWL)[90] with horn-clause rules. This allows declarative representation of complex domain information that may not be possible in OWL alone. OWL being the recommended ontology language for the semantic web [6], has shown considerable expressive powers over other ontology languages, especially its predecessor, the Resource Description Framework (RDF)[4]. However, while OWL ontologies provide simple, reusable and easy to understand domain knowledge models, they lack the declarative expressiveness offered by rules. As evidently shown in the semantic web architecture (Fig.2.1), Rules are projected to support ontologies for efficient domain knowledge representation and the semantic web rule language (SWRL) is one of such rules languages syntactically closest to OWL. Expressive limitations of the RDFS and OWL formalisms can thus be augmented by the capabilities of rules designed for the complex assertion of facts that goes beyond simple declaration of domain concepts. Moreover, being in common practice with well-established logics, rules such as SWRL, offers an efficient reasoning support to ontologies with of course

[4]https://www.w3.org/RDF/

the benefit of added expressiveness.

SWRL Evoultion: SWRL is an expressive rule language designed to enable declarative assertions using OWL concepts. It is fashioned as the union of Horn logic (HL) and Description logic (DL) in order to achieve higher domain expressiveness and reasoning capacity than when using OWL alone. In other words, SWRL is a direct extension of OWL that utilizes its model-theoretic semantics and its syntax basically stemmed from the combination of DL-based OWL and HL-based Rule-ML. Classical SWRL rules include positive, function-free horn-clauses written as implications — consisting of an antecedent (body of the rule), as well as consequent(or head of the rule) — see example rule 2.8. A SWRL-enabled ontology thus contains an OWL Knowledge base and set of horn-clause rule axioms.

SWRL Syntax and Semantics In its human readable form, both SWRL's rule body (B) and rule head (H) typically consist of a conjunction of atoms, which can contain a combination of OWL constructs and axioms. Such constructs can be either in the form of OWL-DL class descriptions — of the form *C(x)*, individual-valued properties — *P(x,y)*, data-valued properties — *Q(x,y)*, OWL same individuals — *sameAs(x,y)*, OWL different individuals — *differentFrom(x,y)*, or the specific built-in functions — *builtIn(r, x,)*. Where 'x' and 'y' are either variables representing OWL classes, properties, and individual data values (in which case preceded by a '?') or OWL individuals themselves, and 'r' is any SWRL built-in function — such as *swrlb:greaterThan(), swrlb:Multiply()* etc.

$$atomB_1 \wedge atomB_2 ... \wedge atomB_n \longrightarrow atomH_1 \wedge atomH_2 ... \wedge atomH_n \quad (2.8)$$

Note that SWRL Rules are written with either $\wedge$ or comma(,) to denote a conjunction (we use them interchangeably in this thesis) and the semantics ensures that the condition in the rules head be evaluated to true whenever the conditions in the body are satisfied. However, there are variants to the basic format above such as rules with disjunctions in their heads or those extended to handle classical negations, etc. The aim of this review is to explore such various expressiveness extensions attributed to the SWRL formalism.

Decidability of SWRL Rules Even though SWRL offers an unrestricted opportunity of combining ontologies and declarative assertion of rules, it however, does so at the expense of decidability. As clearly pointed out in [91], ”there is undoubtedly no inference engine that can draw exactly the same conclusion as the SWRL semantics”. This is due to the highly expressive nature of the formalism and SWRL being in first-order horn clause is unlike its complement OWL-DL, undecidable. Therefore, for SWRL ontologies to be decidable, a restriction needs to be placed on the rule axioms. To achieve decidability, SWRL rules are made to conform to *DL-Safety*, which is a name for a restriction imposed on the rule axioms such that they contain only known concepts [80]. In essence, variables in DL-safe SWRL atoms must be bound only to those concepts or individuals that are known to exist in the ontology. DL-based reasoners (such as Pellet, HermiT and Fact++, etc) are needed to reason over SWRL rules. While the DL-safety may result in an incomplete deduction of knowledge in a given ontology, inferences from DL-safe rules are always formally sound.

However, as is the case with other logic programs, such efforts to keep the SWRL formalism within decidable language constructs and computationally feasible inferences, have resulted in considerable limitations to the ontology rule language. This is evidently discussed in Section 2.8.3, and thus leading to various expressiveness extensions of SWRL proposed over time. In what follows, we briefly compare the A-Box and T-Box decidability requirements followed by highlights in Section 2.8.3 on the strategic importance of SWRL in OWL ontologies — a reasonable justification for our study. For brevity, reasoning paradigm in SWRL is not discussed here and we refer the interested reader to [92].

T-Box versus A-BOX Decidability: Depending on the inference-subject involved, the DL-safety can be interpreted in terms of the T-Box (Concepts or Terminology Box) or the A-Box (Individuals or Facts Assertion Box) of a given ontology. In terms of the T-Box, DL-Safety imposes a limitation that ”only those concepts or terms previously defined in the ontology can be used in the SWRL rules” — meaning that no new or unknown concept may be added to the terminology box. Whereas in relation to the A-Box, the DL-Safety can be interpreted thus: ”variables used in the consequent of a rule must have also appeared in its antecedent” — which implies that no anonymous or loosed individuals can be introduced into the ontology’s A-Box.

Understanding these limitations can thus make it easier to design new SWRL extensions that stay within the decidable fragment of the DL, thereby allowing the inference engines or Reasoners to complete inferences within a finite time. Where such design cannot be easily achieved, a syntactic limitation or manual restriction were usually imposed on the use of the new extensions as discussed in details in Section 2.9.8.

SWRL [93] is an expressive rule language designed to enable rule assertions using OWL concepts in order to achieve higher reasoning capacity than when OWL alone is used. SWRL is a direct extension of OWL that utilizes its model theoretic semantics and its syntaxes adopted from the combination of OWL and Rule-ML [70]. In SWRL + OWL combination, the OWL axioms are extended to include horn-clause rules written as implications consisting of a body (antecedent) as well as head (consequent) as shown in Rule listing (2.5). SWRL ontology therefore, contains an OWL KB and a set of horn-clause rules. In SWRL, the rule’s body and head typically consists of conjunctions of atoms, which can either be empty or contain a combination of OWL constructs and axioms — Class and properties descriptions, individuals, built-in functions etc. For more detailed discussion on SWRL built-in functions, abstract syntax and semantics see [93] and for reasoning techniques employed in SWRL, interested reader is referred to [70, 94, 95].

Decidability Requirement — DL-safety: Even though SWRL offers an unrestricted opportunity of combining ontology with rules, SWRL rules being in first-order horn clauses are generally undecidable. The authors of [96] clearly points out that ’there is undoubtedly no inference engine that can draw exactly the same conclusion as the SWRL semantics’. To achieve decidability, SWRL rules are made to conform to DL-Safe Rules, which restricts rule axioms to contain only known concepts [82]. That is, variables in the DL-safe SWRL atoms must bind only to those concepts or individuals that are known to exist in the Class assertions

(T-Box) or individual assertions (A-Box) of the ontology. Special Reasoners such as Pellet and Hermit are needed to reason over SWRL rules. While the DL-safety may result in incomplete deduction of knowledge in a given ontology, inferences from DL-safe rules are always formally sound. More on the SWRL formalism including its advantages and limitations were discussed in 2.8.3. While, the SWRL expressiveness extensions were explored in Section 2.9.

2.8.2.2 The AL-Log Formalism

The AL-Log representation language is a successful combination of the description logic (ALC) and the DATALOG [97] — a deductive database query language based on the logic programming paradigm. The combination, which allows the specification of DL constraints in datalog clauses, exploits the structuring power of DL (the T-Box) in the deductive database systems [88]. An AL-Log system consists of structural and relational sub-systems. The former allows for expressing terminological knowledge (about classes, roles, individuals), while the latter allows expressing relational knowledge. In order to overcome some of the expressive limitations, a more expressive profile of the AL-Log called disjunctive AL-Log was proposed by [98], which utilizes the non-monotonic features of disjunctive datalogs (DATALOG$^{\neg\vee}$) such as negation as failure and disjunctions.

Decidability Requirement: Whereas SWRL uses DL-Safety to ensure decidability — with no syntactic restrictions imposed on the language, this is not always the case for other formalisms. Usually, restrictions have to be imposed on either the rules syntax or the level of semantic integration between the two components. However, since DL concepts in AL-Log are only used as constraints in the component datalog clauses, the rules can therefore be applicable only to named objects. Hence Al-Log rules are DL-safe equivalent and therefore decidable.

2.8.2.3 The CARIN Formalism

Similar to AL-Log, the CARIN family of rule languages also extends DL ontologies with function-free horn-clauses in the Herbrand model semantics. However, CARIN [87], differs from AL-Log in that both class names and those of properties are allowed to appear in rule bodies as predicates. That is not only having properties as typed constraints for concepts that already appear in rules, as is the case in AL-Log. CARIN also derives its semantics from the combination of its component languages — the description logic and horn-clauses.

Decidability Requirement: CARIN instantiations are decidable where non-recursive rules are used. However, for generality, some restrictions need to be imposed. The authors argue that decidability and sound reasoning is possible in CARIN where Weak-DL is employed but with a further syntactic restriction in the rules head. Such rules are also called *Role-Safe rules* [87, 84]. However, for a rule to be role-safe, at least one variable from each role-literal must occur in some non-DL atom, and that variable must not appear in the consequent of the rule.

2.8.2.4 DL + Log and the Hybrid MKNF Formalisms

Similar to the disjunctive AL-Log formalism, DL+Log knowledge representation formalism involves extending DL ontologies with a disjunctive logic program rules (specifically, the disjunctive datalogs DATALOG$^{\neg\vee}$) [81, 20, 99]. It should be noted however, that while the Al-Log formalism safely integrates the $\mathcal{ALC}$ fragment of DL and Datalog rules, DL+Log on its part, provides a weak-safe integration of any DL fragment with a disjunctive Datalog.

Decidability Requirement: For the DL+Log rules to be decidable, variables must appear only in positive atoms in the rule's body and those that appear in the head must also appear in the non-DL atoms — called 'weak safeness'. An extended version of this formalism is the 'Hybrid MKNF' [87], which is also a combination of DL ontologies with disjunctive logic programs. The basic difference is the interpretation of the rule components. In DL+Log, rules are interpreted either according to 'stable model semantics' where non-monotonic semantics are used or as 'material implications' where the rules are in first order semantics. However, in Hybrid MKNF as the acronym suggested, they are interpreted based on the Lifschitz's logic of 'Minimal Knowledge and Negation as Failure' [100].

2.8.2.5 The dl-programs

The *dl-programs* also extends DL ontologies with extended logic programs (non-monotonic using both classical and default negation). The main idea behind the dl programs KBs is that the logic program rules can be used to query the component DL ontology. Thus enabling new information embedded as queries to be added into the ontology. This in turn influences the resulting inference due to results or answers from the queries — interpreted according to the answer-set semantics [85, 101]. Hence, information flow is bi-directional in dl-programs knowledge bases, as facts can be exchanged between the ontology and rules components.

Decidability Requirement: As earlier stated, in order to achieve finite inference on most hybrid knowledge bases, restrictions are usually imposed on either the rules component syntax or on the level of semantic integration between the rule and ontology components [82]. Similarly, for $dl - programs$ to be decidable, the restriction goes that both universal and existential restrictions are not allowed in the rules body and head respectively. Due to such restrictions, the dl-programs formalism is found to be considerably less expressive than either of its components (OWL and hornclauses). On the other hand, homogeneous formalisms (such as SWRL) are found to be significantly more powerful than either of their components (the OWL-DL or horn-clause).

The Table 2.1 below summarizes the common formalisms discussed, highlighting their components, degree of integration, logic program formats, and their required condition(s) for decidability:

Formalism	**Components Integration**	**Integration Type**	**LP format**	**Condition for decidability**
AL - Log	Integrates ontologies (expressed in $\mathcal{ALC}$ fragment of DL) with positive Datalog rules.	Loose Integration (Safe-interaction)	Datalog (function-free) clauses (Uses only unary predicates i.e. classes in rules)	Decidable (Only LP predicates may appear in the rule consequent and DL fragments are used as constraints in the antecedent)
CARIN	Extends AL-Log to include DL roles. i.e., integrates ontologies expressed in any DL fragment and function-free Horn-clauses.	Tight Integration (generally unsafe interaction)	Datalog Clauses (Allow both classes & properties to appear in rules i.e. unary and binary predicates)	Role-Safe rules (Guarantees decidability for Weak-DL - ALCNR)
Disjunctive Datalog (DL + Log $^{\neg\vee}$)	Integrates ontologies expressed in any DL fragment and negative datalogs (DATALOG$^{\neg\vee}$).	Tight Integration (weakly-safe interaction)	Negative Datalog clauses	Weak Safeness (variables in rules must appear in positive non-DL atoms)
Hybrid MKNF	Integrates DL ontologies and disjunctive logic programs.	Tight Integration (Full integration)	Negative Datalogs (Interpreted according to the logic of MKNF)	The disjunctive Datalogs must be DL-safe

Formalism	Components Integration	Integration Type	LP format	Condition for decidability
dl-programs (DLP)	Integrates (intersection of) DL ontologies and extended logic programs.	Loose Integration	Non-monotonic Datalogs (with both classical and default negation)	Rule body No universal restrictions. Rule head No existential restrictions. (Generally decidable but less expressive)
SWRL	Integrates (union of) DL ontologies expressed in OWL-DL/SHOIN(D) and extended logic programs.	Tight Integration (Medium syntactic separation)	Datalog Horn Clauses	DL-Safeness DL atoms may also occur in the rule consequent (Generally Undecidable and highly expressive)

Table 2.1: Summary of Formalisms for Extending Ontologies and Rules

2.8.3 Why SWRL?

As shown in the case studies section, SWRL formalism was selected to extend the web ontology modeling by adding if-then rules. As discussed earlier, rules generally allow efficient declarative assertions in domain knowledge modeling. However, the semantic web rule language (SWRL) is particularly important in the semantic web knowledge modeling as it allows both the assertion of facts in OWL ontologies as well as their retrieval. Information retrieval is achieved through its slightly modified Semantic Web Query language(SQWRL), which is an SQL-extended version of SWRL for querying OWL ontologies [102]. The flexibility for users to define application-specific methods (user-defined built-ins) as extension to the SWRL formalism is an important feature that makes SWRL formalism indispensable where domain modeling using OWL is considered. Regarding the added expressiveness of SWRL to the OWL language, the authors of [103] summarize some of the basic advantages of SWRL as follows:

(a). The ability to use class names or their descriptions as predicates.

(b). The use of equalities and inequalities, and

(c). Allowing conjunctions of atoms in both SWRLs antecedent and consequent.

These syntactic advances, coupled with the numerous expressiveness advantages of logic programming rules enables SWRL to achieve complex representation of domain knowledge — making the formalism indispensable in the Semantic Web project. Further reasons for our focus on the SWRL formalism include the facts that:

(d). SWRL axioms can be used in transferring characteristics from one class or property to another without sub-classing — which goes beyond the expressive powers of OWL. A commonly cited example of this transferring property is the composite property assertion, popularly referred to as 'the uncle relationship'.

(e). SWRL rules allow inference of new individuals using existential operators (e.g. *swrlx:makeOWLThing*), some of which are defined as built-ins in the formalism (see Section 2.9.6). The ability of SWRL to achieve existential quantification as well as syntax extension through the use of user-defined built-ins, no doubt goes beyond the expressive powers of OWL and even those of the classical horn clause rules.

(f). Lastly, SWRL being a semantic extension of OWL have enjoyed considerable experts' commitment and engaging support from the Semantic Web research community.

Consequently, the compatibility of the SWRL formalism with OWL and also the RuleML, makes it easier and sometimes even compelling, for researchers interested in semantic web rules to equally extend SWRL's capabilities whenever the corresponding OWL language is augmented with additional syntax and semantics. A similar view was upheld in [5], where the authors affirms that the added expressiveness of OWL to cover negative property assertions inspired them to consider a corresponding SWRL extension. SWRL's negation extensions were first described and analyzed in [104].

Limitations of SWRL However, despite the expressive powers obtained from SWRL, the combination (of OWL and SWRL) is yet to guarantee an all-inclusive domain modeling language. The classical SWRL formalism cannot appropriately represent various real-world scenarios. For example, expert opinions, which forms a considerable part of every domain knowledge, are typically in the form of imprecise facts. Thus requiring formalisms that are capable of representing domain facts based on some *degree of certainty* or partial truth. Likewise, in legacy software models, much of formalized domain facts, such as business rules may only hold where other fragments of existing knowledge remain valid. Therefore, accurate representation of such scenarios requires *knowledge exclusion, prioritization of facts,* as well as *knowledge retraction,* among others. These forms of knowledge modeling scenarios are crucial to the effectiveness of expert systems and their related applications. With common examples found in information fusion, multimedia information processing, automated ontology merging and alignment, among others. In the remainder of this subsection, however, we highlight some of the commonly cited limitations of the classical SWRL formalism as follows:

- **Limitation in modeling imprecise domain knowledge.** The evolutionary and sometimes inconsistent nature of human knowledge necessitates representing vague or imprecise domain information. Hence, the inability of SWRL to represent inherently vague domain knowledge and business uncertainties is therefore, a huge setback in modeling real-world scenarios on the semantic web. This has led to the various fuzzy and probabilistic extensions of the semantic web rule language.

- **Lack of Non-monotonic constructs.** Rule-based modeling of a knowledge domain involves expressing domain facts or situations of domain objects, through the use of 'if-then' statements. This ordered representation of fact, basically reflects the ordered nature of the human knowledge — which typically includes ordering or setting a precedence of activities, leading to the addition of new facts to existing knowledge bases. The inability of SWRL to comprehensively model non-monotonic facts such as existentials, quantifiers, rule exclusion, and prioritization, etc., were also considered as weighty limitations that cannot be otherwise ignored. Moreover, it has been shown that deductive forms of inference, where new facts get inferred from the absence of other facts, cannot be represented by the SWRL formalism [105].

- **Removal of Facts.** Further examples of realistic scenarios encountered include the need for ’unlearning’ facts in knowledge bases, i.e. removal of facts. Constructs needed to model and achieve such scenarios were found to be lacking in the classical SWRL definition.

- **Lack of Support for Modeling Complex Scientific Knowledge.** Other limitations include the SWRL’s inability to efficiently model scientific and engineering knowledge domains, especially those involving complex mathematical formulas and constraints [106]. The lack of precise constructs to handle complex engineering formulas usually leads to the development of voluminous set of rules to explain few facts.

2.8.4 Common Concerns when Augmenting Ontologies with Rules

Key concerns when integrating ontologies and rules include, the expressiveness or representational adequacy of the combined KR formalism, the inference or reasoning capability of the combination, and maintaining their individual completeness or decidability [107]. Major concerns when integrating ontologies and rules may be summarized in the following questions: *’What will be the representational adequacy of the resulting KR formalism?, What is the inference mechanism of the combination?* and *How can the individual completeness and decidability be maintained?’* We briefly highlight the more generic semantic and reasoning problems that must be considered, at least theoretically, when combining ontologies with rules

- Monotonicity vs Non-monotonicity: DL-based languages (e.g. OWL) are naturally monotonic and based on the Open-World Assumption (OWA) while logic program rules usually works on the Close-World Assumption (CWA) and therefore non-monotonic. Recall that in OWA, existence of a $thing$ is assumed to be true unless it is stated that it is false. In other words, something that cannot be found in a KB shall not be assumed absent unless it is explicitly stated — a notion always assumed in ontologies but not in logic programming (rules). Common examples are the classical negation and negation as failure (NAF), which is supported in rules but not in the ontologies. As such, these inconsistencies must be taken into account when using rules on top of the ontologies. In our case, the problem is less pronounced as both OWL and SWRL, being descendants of DL, are non-monotonic in nature.

- Naming Conventions: Similarly in naming conventions, rule formalisms are usually based on the Unique Name Assumptions (UNA), where a concept name is assumed to be uniquely belonged to an individual instance. Whereas recent Ontology languages, operate under the Non-Unique Name Assumptions (non-UNA) such as for example, the ability to assign a single name to a class and individual in OWL 2. Unlike their predecessors however, OWL-DL and its OWL 2 progenies do not support the UNA, which means that a single individual can be referenced by two or more names. Thus, combining ontologies and rules always leads to the question of 'How to successfully integrate and reason over the semantics of non-UNA of OWL with the UNA of rule formalisms'.

- Decidability: another important aspect that must not be ignored when considering the integration of ontologies and rules includes the availability of reasoning tools to classify and assert the consistency of the combination. Recall, that the combination of ontologies and rules are not always decidable. As such, clear-cut conditions for achieving decidability of a combination must also be considered. For example, the OWL + SWRL combination is undecidable unless the rules are kept within the *DL-Safety* restriction.

These concerns need to be addressed and their effects evaluated during the selection of the rule formalism for integrating an ontology KB. However, a simple way out as found in our survey is to employ those components (the ontology and rules languages) that share similar syntax and semantics thereby resulting in a KB that can be easily classified within a single reasoning tool.

2.9 Semantic Web Rule Language Expressiveness Extensions

As mentioned in the previous section, despite the high expressive powers of the semantic web rule language, the OWL/SWRL combination does not guarantee a comprehensive ontology modeling language. Classical SWRL definition is found to be lacking descriptive constructs to model complex real-life scenarios. This has led to the proposals of various expressiveness extensions to the classical SWRL formalism, ranging from simple mathematical built-ins to language extensions that allow modeling of vague facts and predictive knowledge using SWRL rules.

This chapter explores the limitations of SWRL and reviews the available SWRL expressiveness extensions proposed over time[5]. The chapter is organized as follows: Section 2.8.2.1 gives a detailed overview of the SWRL formalism, highlighting its evolution, syntax and semantics, the need for SWRL in OWL, as well as its expressive limitations. This is followed by the essence of the chapter in Section 2.9, where we discuss, with running examples, the various expressiveness extensions added to the SWRL formalism. In order to ascertain their viability, the decidability requirements of the reviewed extensions were discussed next in Section 2.9.8. Section 2.9.7 provides a discussion on the added expressiveness with a table of summary categorizing the extensions alongside their added syntaxes and semantics. As a prelude to our novel SWRL extension, we discuss the relevant works in fuzzy temporal representation and reasoning in Section 2.11. Lastly, the chapter summary is presented in Section **??** leading to subsequent phases of the report.

[5]Manuscript titled: *Towards Comprehensive Domain Modeling on the Semantic Web - a Review of SWRL Expressiveness Extensions* is submitted to the Journal of Web Semantics, Elsevier, Manuscript Number: JWS-D-17-00081, March, 2017.

SWRL extensions are originally defined in [70] as *bindings* that provide a mapping between variables used in the SWRL rules to objects of a given domain. As previously mentioned much of SWRL's extensions were inspired by their corresponding OWL extensions. This is usually achieved by modifying the OWL's abstract syntax and semantics to adapt to the SWRL canonic mode. This leads to various proposals that are beneficial to the advancement of both the OWL and SWRL languages. The remainder of this section highlights some of the expressiveness extensions designed to address the commonly encountered limitations of the classical SWRL.

2.9.1 Fuzzy and Probabilistic Extensions — Dealing with Uncertainties and Incomplete Knowledge

Knowledge from domain experts is usually not without doubts and imperfections. This imperfect nature of real-world information processing necessitates the long-started efforts by the AI community to deal with vague knowledge representation. Notable early efforts in the field of classical domain modeling have attempted to improve the expressivity of Description Logic (DL) to cover uncertainties through Fuzzy Logic (FL) as discussed in [108] and probabilistic extensions [109], among others. Likewise, SWRL is found to be inadequate in expressing practical problems that involve vague or imprecise domain knowledge which is bound to be common in ontologies or the Semantic Web as a whole. Moreover, even with the success of semantic web in data provisioning, writing business and engineering rules using SWRL still remains a challenge. This is due to the fact that SWRL, much like other Semantic Web languages, stemmed from classical logics (see Figure 2.4), which are known to be incapable of modeling vague or imprecise information.

As highlighted in [110], other scenarios that require modeling of imprecise facts can be found in multimedia processing, ontology alignment, and information fusion, among numerous others. In response, SWRL has since received significant advancements to handle imprecise domain knowledge. One famous uncertainty reasoning techniques involve the combination of fuzzy logic with SWRL and another involves extending SWRL formalism with probability theories. This section reviews some of these advancements and briefly discusses their individual approaches.

2.9.1.1 The Fuzzy-SWRL Extension (F-SWRL)

Expressiveness: By imposing fuzzysets theory — precisely, the 'R-implication' of fuzzy logics [111], new semantics were defined by Horrocks et al. for a fuzzy extension of SWRL (f-SWRL) [110]. A fuzzy set is defined by its degree of individual membership (w) called weighted-degree or truth-value and usually computed from a *membership distribution function*. The fuzzy membership function does not only specify whether an element belongs to a given set or otherwise but also how much. Analogous to fuzzy sets, f-SWRL rules uses a truth value between '0 and 1' to express the degree of confidence for individual membership in a given class or property and also to express weights or importance of each atom in a SWRL rule.

Syntax and Semantics: Class and property definitions in f-SWRL have the following form $C(x) * w$ and $P(x, y) * w$, respectively. While a rule in f-SWRL is of the form:

Antecedent * $w_a \longrightarrow$ *Consequent* * w_c. Where $w \in [0, 1]$.

For example, the following f-SWRL rule (retained from [110]), declares that "being healthy is more important than being rich to determine if one is happy":

$$Rich(?p) * 0.6 \wedge Healthy(?p) * 0.8 \longrightarrow Happy(?p) * 0.9 \tag{2.9}$$

Where: Rich, Healthy, and Happy are fuzzy class URI refs, ?p is an individual valued variable and 0.6, 0.8 and 0.9 are the assigned weights of the corresponding fuzzy atoms.

In a nutshell, f-SWRL extends SWRL with fuzzy-based class and property definitions as well as fuzzy rule axioms. The result is that f-SWRL axioms are able to represent such information that says how much a concept is believed to be true and which facts are more important than others when making decisions. Users can, therefore, represent vague domain knowledge with f-SWRL rules by asserting the degree of confidence or otherwise of a given fact.

Implementation and Efficiency: In assigning a predetermined weight on the consequent atom, the *f-SWRL* extension is shown to be suitable for writing fuzzy rules with atomic consequents and thereby making rule prioritization. This absolute form of prioritization, however, becomes a problem whenever new rules are introduced in the KB, as then all the existing weight values may have to be readjusted. Moreover, the proposal does not present a way of resolving such conflicts. As such, more support is needed for f-SWRL to deal with non-atomic fuzzy rules, i.e. rules with more than one atom in the consequent. Furthermore, complete fuzzy-rule declaration needs to be considered in f-SWRL since representing uncertainties in domain knowledge requires more than just fuzzy class and property definitions. Based on our evaluation, much of the f-SWRL proposal is still theoretical as neither implementation efforts nor practical scenarios were mentioned. Likewise, the fuzziness of the extension has been critically questioned by the authors of [112], asserting that syntax and semantics proposed in f-SWRL do not actually solve much of the fuzzification problem. For an abstract literature on fuzzysets theory and fuzzy logics, we refer the interested reader to [111].

2.9.1.2 The Vague-SWRL Extension

Highlighting their argument against f-SWRL's inability to provide a substantial fuzzy extension to SWRL, the Vague-SWRL extension was proposed in [113].

Expressiveness: In this proposal, the authors argues that the use of single membership degree to describe a fuzzy set — such as the single weight function in f-SWRL extension, is insufficient in representing vague information. Consequently, based on the theory of Vague sets [114], the authors propose the Vague-SWRL as another fuzzy extension of SWRL. Vague Sets are themselves an extension of fuzzy sets, where the degree of membership to a set is evaluated using two weighted intervals — as opposed to a single degree of membership employed in fuzzy sets. As such, Vague-SWRL rules uses an added weight value (w_2) called 'a second-degree weight' denoting further degree of membership to support and balance an initial weight (w_1). By introducing w_2, the proposal promises a more accurate representation of imprecise domain knowledge than a single membership fuzzy class and properties assertions of f-SWRL.

Syntax and Semantics: As presented in [114], the general form of vague-SWRL is written as:

$$\textit{(vc * fdw) (vcv * sdw) ...} \wedge \textit{(vp * fdw) (vpv * sdw) ...} \longrightarrow \textit{(vc * w) or (vp * w)}$$

Where, 'vc' = vague classes, 'fdw' = first-degree weights, 'vcv' = vague class values corresponding to 'vc' and 'sdw' = second-degree weights. Similarly, 'vp' = vague properties, 'vpv' = vague property values and 'w' = the atomic weights. The vague values, also called the membership intervals, are calculated as:

$$vcv/vpv = [t_v(x), \quad 1 - f_v(x)]$$

And the 'sdw', also referred to as the 'vagueness' or second degree of membership, is calculated as the difference:

$$w_2 = [(1 - f_v(x)) - t_v(x)]$$

Where, 't_v (x) is the true membership function of x, f_v(x) its false membership function, and $0 \leq t_v(x) + f_v(x) \leq 1$

Example Case. Consider the following additional information (membership degrees) added to rule (1): *"P is rich with a true value of 0.6 and a false value of 0.3. Also, P is healthy with a true value of 0.3 and a false value of 0.2"*.

With this additional info, we can represent the parameters:

$$vcv(Rich) = [0.6, 0.7] \; and \; vpv(isHealthy) = [0.3, 0.8]$$

Hence, the vague-SWRL form of rule (1) can then be written thus:

$$[Rich(?p) * 0.6][0.3] \wedge [Healthy(?P) * 0.8][0.1] \longrightarrow Happy(?p) * 0.9 \quad (2.10)$$

Here, 0.9 is the degree to which the consequent holds following the evaluation of the antecedent.

Implementation and Efficiency: As vague sets subsumes fuzzy sets, vague-SWRL ultimately subsume its corresponding f-SWRL rule in terms of expressiveness. By comparing rules 2.9 and 2.10, it can be seen that unlike f-SWRL, vague-SWRL is more than just a conjunction of weighted atoms. The added effort in the form of vcv and vpv are calculated to represent the uncertainties in the fuzzy membership classes Rich, Healthy, and Happy. Moreover, by specifying the upper and lower bounds of membership intervals (through the true and false values), vague-SWRL rules are more justifiably accurate in representing imprecise knowledge using fuzzy class and properties. Vague-SWRL is also claimed, by the authors, to be in ”an acceptable form of the Rule Interchange Format (RIF)”. A comparable extension, *Vague-RuleML*[115], is also proposed for the rule markup language (Rule ML).

However, similar to f-SWRL, the Vague-SWRL only represents fuzzy information for class and property memberships, which may be inadequate in representing practical uncertainties involved in knowledge domains and the semantic web. Also, a fair understanding of vague sets and vague knowledge representation is required to efficiently model fuzzy information using vague-SWRL rules. Hence, there is a need to improve the proposal with richer fuzzy modeling syntaxes to handle imprecise domain knowledge beyond the borders of class and property memberships — the SWRL-F extension below gives a good example.

2.9.1.3 The SWRL-Fuzzy Extension (SWRL-F)

The SWRL-F extension [108] was proposed as another Fuzzy Logic (FL) extension to the SWRL formalism. However, unlike f-SWRL and vague-SWRL extensions, which were based on the fuzzy sets principle, the SWRL-F extension expresses fuzzy reasoning in SWRL rules using a ’fuzzy control system’ approach.

Expressiveness: In this approach, the main ontology remains intact while a fuzzy ontology consisting of SWRL-F rule-base is added to model any ambiguous domain knowledge using logical variables. The fuzzy ontology is needed to define inherent fuzzy domain knowledge using fuzzy: sets, terms, variables and fuzzy values as entities. These entities were defined as classes with their respective object and data properties. Modelling a fuzzy fact requires the use of the fuzzy terms to calculate a fuzzymatch for the respective instances.

Syntax and Semantics: SWRL-F introduces a special object property called 'fuzzymatch' for each fuzzy set. The fuzzymatch is used when designing SWRL-F rules to match corresponding Fuzzy-Variables with designated Fuzzy-Values from each respective fuzzy set. For example, the following SWRL-F rule (adopted from [108]) can assert the vague facts; *"Persons with good health status are always very happy"* and can be written using SWRL-F rule as:

$$\begin{aligned} Person(?p) \wedge hasHealthStatus(?p, ?s) \wedge fuzzymatch(?s, goodHealthStatus) \\ \wedge \ isHappy(?p, ?h) \longrightarrow fuzzymatch(?h, VeryHappy) \end{aligned} \tag{2.11}$$

Here, the fuzzymatch attribute is used in the antecedent of the rule to calculate the degree of membership for the 'HealthStatus' variable (?s) — as employed in the fuzzy term 'goodHealthStatus' denoting the fuzzy set in this instance. While used in the Consequent, the fuzzymatch variable allows binding the new value 'VeryHappy' to the fuzzy value 'isHappy', giving SWRL a chance to express the wooly term, 'very happy'.

Implementation and Efficiency: In essence, the fuzzy ontology and SWRL-F rules are only added where necessary to handle uncertain domain knowledge representation. Furthermore, the authors claim that the new rule language, SWRL-F, is supported with an exclusive publicly-available ontology development environment having a test execution engine. However, several attempts at finding the link currently present an empty wiki page. An apparent limitation to the SWRL-F implementation is that some of the current OWL2 constructs cannot be utilized in the SWRL-F rules. Moreover, running the SWRL-F rules requires a modified version of the *SWRL-Jess* tab in Protégé ontology editor — an implementation requirement that may be difficult to domain experts. Moreover, using the language to model vague domain knowledge also requires an in-depth understanding of fuzzy logic and representation scheme.

On the other hand, due to its adoption of a well-established approach — the fuzzy control system approach, SWRL-F offers a more pragmatic and justifiable approach to fuzzy extension of SWRL as compared to its siblings, f-SWRL and vague-SWRL. The use of the 'fuzzymatch' attribute also makes SWRL-F suitable for developing semantic web applications with queries that require fuzzy inferencing from facts separately stored in the main ontology. In addition, since by design the SWRL-F fuzzy ontology is separated from the domain ontology, reasoner inferences are thus limited only to the SWRL-F rulebase. This is desirable nonetheless, as the modularity avoids introducing inconsistencies to the main ontology.

2.9.1.4 Fuzzy Non-monotonic Extension of SWRL (f-NSWRL)

The f-NSWRL [116], presents yet another fuzzy SWRL extension for dealing with non-monotonicity as well as uncertainties in domain information using SWRL rules.

Expressiveness: This extension is basically an advancement of the fuzzy-SWRL extension (Section 2.9.1.1) to incorporate a non-monotonic knowledge, specifically the knowledge negation (called classical negation) and Negation as Failure (NAF), which involves knowledge modeling based on the absence of positive facts in a KB. This is also referred to as Closed World Negation (CWN). Citing the importance of expressing classical negation and the NAF in handling rule exceptions, the authors asserts the motivations for f-NSWRL extension of SWRL — as a non-monotonic as well as a fuzzy extension of the SWRL formalism.

Syntax and Semantics: In its simplest form, the f-NSWRL uses the 'Not' and '$\neg$' symbols as operators to extend the original f-SWRL's fuzzy classes and properties definition. As an example, consider the addition of the following information to modify rule 2.9 in an attempt to determine if a person is happy; *"Person p is rich and p is definitely not hungry but the health status of p is not known"* Decision rules based on the above statement can be expressed using f-NSWRL as follows:

$$Healthy(?p) * 0.8 \land (\neg Hungry(?p)) * 0.5 \longrightarrow Happy(?p) * 0.9 \tag{2.12}$$

$$Healthy(?p) * 0.8 \land not(Hungry(?p)) * 0.5 \longrightarrow Happy(?p) * 0.9 \tag{2.13}$$

Note the difference in the use of the two operators as highlighted in the two equations. While the classical negation ($\neg$) is used to negate the assertion that Person (?p) is hungry in rule 2.12 and the NAF used in 2.13 to test the absence of an assertion that 'p is hungry'. A quick interpretation of the two rules is that rule 2.12 simply declares that 'A person must be *not hungry and healthy* to be happy'. Whereas rule 2.13 declares that 'a person is happy if she is healthy and is *not known* to be hungry'.

Implementation and Efficiency: Apart from handling negation and uncertainty, other notable aspects of the f-NSWRL proposal include a proposed markups in 'RuleML' for translating f-NSWRL to other rule languages and also procedures to handle rule prioritization — for setting rule precedence in cases of conflicting consequents. However, the proposal failed to mention the semantics of these extensions nor the inference mechanism for implementing a supporting Reasoner.

We discuss more on specific Non-monotonic extensions of SWRL in Section 2.9.2 below.

2.9.1.5 The Bayesian Extension of SWRL (Bayes-SWRL)

In order to allow modeling of predictive knowledge and the representation of inherently probabilistic domain knowledge (such as Statistical information) on the semantic web, various probabilistic extensions were proposed — to the semantic web languages. These include among others, the Probabilistic RDF (pRDF) and Probabilistic OWL (PR-OWL) [117], Bayes-OWL[118] and Bayes-SWRL [119].

Expressiveness: While Fuzzy Logic extensions of SWRL focus on representing the degree of certainty or otherwise of domain knowledge in SWRL-enabled ontologies, probabilistic extensions of SWRL such as Bayes-SWRL are more concerned with representing predictive knowledge based on an existing partial knowledge in the domain ontology or knowledgebase. Based on the Bayesian Networks of probability theory [120] and in line with the corresponding Bayesian OWL extension [118], the Bayes-SWRL extends the SWRL formalism with the ability to model probabilistic knowledge. This is achieved by combining SWRL with the expressive capabilities of the Bayesian Logic Programs thereby enabling probability assertions during inference.

Syntax and Semantics: The Bayes-SWRL extension naturally adopts the original SWRL's abstract syntax and semantics with the addition of few innovative symbols and patterns to represent the probability variable (the p-variable) and related terms — such as the Conditional Probability Table (CPT). Similar to the fuzzy weights (w) assigned to Vague-SWRL rule atoms, the Bayes-SWRL also use a Probability variable to assign probabilities to rules atoms in both the antecedent and/or consequent atoms. The probability value also ranges from [0, 1] and is optionally added to the rule atoms, with its values predefined in a conditional probability table supplied to the consequent atom in the form of an XML file. In its human readable syntax, Probability values of imprecise atoms are attached with an asterisk (*) to the rule atoms. While the CTP file path is attached using the '@' symbol (as shown in rule 2.14 below). In this form, a Bayes-SWRL that asserts that *"It might rains with a certain probability, if it is cloudy and humid"* can be written as:

$$\begin{aligned} &Cloudy(?cl) \, * \, p(cl) \, \wedge \, isHumid(?hm) \, * \, p(hm) \longrightarrow \\ &\qquad\qquad Rainfall(?r) \, * \, p(r)@"RainfallCPTs.xml" \end{aligned} \tag{2.14}$$

Where: the p-variables, * p(cl), * p(hm) and *p(r), represents the probability values for cloudy, humid and rainfall, respectively. Assuming probable rainfall chances were calculated, for optimal cloudy and humid conditions, in the supplied *RainfallCPTs.xml* file. A more detailed description of the abstract syntax and semantics of Bayes-SWRL can be found in [119].

Implementation and Efficiency: The proposal of Bayes-SWRL is equipped with a reasoning algorithm implemented by extending an existing OWL-DL reasoner based on the tableaux algorithm, the Pellet reasoner, to interpret the added syntax based on the defined probabilistic semantics. It is also equipped with a user interface for checking rules conflict and viewing the inference process, among others.

In our opinion, the extension is effective as it is able to provide a well-defined syntax and semantics for modeling uncertainties. However, the extension being a by-product of BLP and SWRL automatically inherits the constraints of both formalisms. Hence for Bayes-SWRL rules to be decidable, the DL-safety restriction must be adhered to by its reasoner. Similarly, the BLP preconditions apply to a Bayes-SWRL rule, such as: firstly, a consequent atom can only be influenced by a finite number of random variables — thereby adopting the closed world assumption. While this may not pose any serious threat to domain modeling, it however, limits the capabilities of Bayes-SWRL to represent NAF. Secondly, in the inference relation between atoms, there can be no cycle in the dependency graph. Hence expressing relationships is severely limited. Thirdly, a probability of any consequent atom can only be influenced by the probabilities of corresponding antecedent atoms in the same rule. Meaning that, probabilities defined in previous rules cannot be reused in new rules. Therefore, contrary to the non-monotonic nature of the real-world knowledge, especially those found in the semantic web, this precondition demonstrates a monotonic limitation of Bayes-SWRL knowledge bases. Moreover, while the conditional probability of a Bayes-SWRL rule is based on a well-founded semantics, as is the case with $pDatalogs$ [109], the declaration of an arbitrary set of probable states for the ground atoms beforehand, poses some close-world expressive limitation for modeling continuously evolving knowledge domains.

2.9.2 SWRL Non-monotonic Extensions

With the proposed extension of the OWL2 profile to handle negative property assertions, there is a corresponding effort to also extend SWRL with non-monotonic operations — notably presented in [104] and [5]. Moreover, since SWRL formalism basically involves OWL constructs coupled with rule axioms, any OWL extension inherently results in a new SWRL extension. While non-monotonic axioms involve constructs that produce knowledge bases conforming to the Closed World Assumption (CWA), monotonic constructs generally follow the Open World Assumption (OWA), as required in the Semantic Web environment. As such, the semantics of these non-monotonic extensions to SWRL have to be carefully modeled to keep the new rules decidable. For brevity of scope, we present here only the non-monotonic extensions added to the SWRL formalism. These include dealing with negation and removal of Facts, rules exclusion and prioritization among others. For a detailed description of Monotonicity and the applications of non-monotonic logic in rules, we refer interested readers to [5].

2.9.2.1 The 'not' operator — Weak Negation or Negation as Failure (NAF)

Owing to its monotonic background inherited from Description Logics, and the OWA of ontologies, the original SWRL formalism does not support negation as failure (NAF) — as that will logically violate the open-world assumption. As such, both its antecedent and consequent can contain only positive conjunctions of atoms or facts.

Expressiveness: Hence, the NOT operator (also called weak negation) was introduced to allow for expressing negation of facts. Negation of facts simply means, modeling the absence of positive, known or existing facts in a knowledge-base. It should be noted here that whenever a negation of membership is intended, a strong negation (also called complement) is easily achieved in SWRL using the *owl:complementOf* class description.

Syntax and Semantics: Abstract definition of the *not* operator is depicted in the following table:

Example Case. The following Non-monotonic SWRL rule declares that *"A person that does not have a 'spouse property' is automatically a member of the Singles class"*:

SWRL Elements	Pattern Matching (P)	Condition Test
$P(x;y)$ $Q(x;y)$	$(?xP?y)$ $(?xQ?y)$	$S(P) = True$
$not(P(x;y))$ $not(Q(x;y))$	$(?z\ rdf : type$ **owl:NegativeProperty**) $(?z\ rdf : subject\ ?x)$ $(?z\ rdf : predicate\ [P\ or\ Q])$ $(?z\ rdf : object\ ?y)$	$S(P) = False$

Table 2.2: Syntax of Weak Negation (NAF) [5]

$$Person(?p)\ \wedge\ Not(hasSpouse(?p, ?s)) \longrightarrow Single(?p) \tag{2.15}$$

For the complement operator or Strong Negation (as previously discussed in f-NSWRL — see Section 2.9.1.4), using the above example, we can write the rule by testing the existence of 'P' in the married class, as follows:

$$Person(?p)\ \wedge\ Married(?m)\ \wedge\ (notMarried(?p)) \longrightarrow Single(?p) \tag{2.16}$$

Where, the 'not' enclosed in the bracket together with the 'Married' class name, denotes the *complementOf* relationship for the married class.

Implementation and Efficiency: However, a special reasoner is required to run extensions having the weak negation (NAF) as shown above. On how to express the semantics of NAF, the 'non-monotonic inference process' has been highlighted in [5]. On the other hand, the classical or strong negation can be easily achieved in OWL and subsequently in SWRL, using the *owl:complementOf* class description, which can also be employed in the SWRL rules — as shown in rule 2.16.

2.9.2.2 The Quantifiers — Exists, ForAll, and notExists operators

The Quantifiers were introduced in SWRL axioms to handle incomplete facts, when used in the antecedent (body of the rule) and for removing facts, when used in the rule's consequent (head). In this set of expressiveness extensions, two types of fact quantification were introduced here. First, the Existential quantifiers — the Exists ($\exists$) and notExists($\nexists$) used to assert an existence and otherwise respectively, of at objects in a knowledge base (KB). Secondly, the Universal quantifier — ForAll ($\forall$), which generalizes some assertion on group of objects in a KB.

We discuss the *notExist* extension here. While the *Exists* and *ForAll* quantifiers with their example use cases, will be discussed in Section 2.9.4 as part of the SWRL existential extensions.

Expressiveness of 'notExist' Quantifier Due to the different semantic interpretations attributed to the antecedent and consequent sides of rule axioms, the notExist quantifiers can be used to achieve different assertion of facts (expressiveness) depending on which side they are placed in a SWRL rule. These assertions are presented in the following scenarios:

1. *notExists* Operator in the Antecedent — Checking for missing facts: To handle missing information as facts, the 'notExists' and 'Exist' quantifiers were proposed to allow some action to be taken where certain facts are not defined in the KB. This is useful because due to the OWA of SWRL KBs, it is not always feasible to write rules that, for example, enumerate or test-out for all individuals or properties. In such cases, asking for the existence or otherwise of a particular individual or property offers a simple solution. For example, the following rule (2.17) checks for the absence of participants with spouses in a booking register and asserts the status of the register.

$$Participant(?p) \wedge hasBooking(?p, ?b) \wedge \textbf{notExists}(hasSpouse(?p)) \\ \longrightarrow bookingStatus(?b, "SinglesOnly") \quad (2.17)$$

2. *notExists* Operator in the Consequent — Removal of Knowledge: Conversely, using the NotExist operator in the consequent of a SWRL rule,

results in the removal of knowledge. For example, rule 2.17 above may need to be retracted whenever a married participant made a booking and the ’hasSpouse’ property get added to the ontology. Instead of deleting the rule manually, a better option, would be to change the ’bookingStatus’ information using another rule (2.18), as follows:

$$\begin{aligned} &Participant(?p) \wedge hasBooking(?p, ?b) \wedge hasSpouse(?p) \\ &\qquad \longrightarrow \textbf{notExists}(bookingStatus(?b, "SinglesOnly")) \\ &\qquad\qquad \wedge bookingStatus(?b, "Mixed") \end{aligned} \tag{2.18}$$

A clearer example is also presented in the mail list update function shown in rule 2.19, where the ’notExist’ operator is used to test whether a member of a workgroup exist in a membership mailing list before adding them to an Alumni mail list:

$$\begin{aligned} &Workgroup(?g) \wedge Alumni(?a) \wedge hasMember(?g, ?a) \longrightarrow \\ &\qquad \textbf{notExists}(mailListMember(?a, ?g) \wedge AlumniMailList(?a) \end{aligned} \tag{2.19}$$

Syntax and Semantics of the ’notExist’ Quantifier In continuation of the definition of the Non-monotonic operators, the Table 2.3 below shows the notExists constructs and is used as is, in the human-readable syntax of SWRL. The semantics hold that S(P) becomes universally satisfiable for any unbound elements (A, B Z) associated to the notExist operator.

SWRL Elements	Pattern Matching (P)	Condition Test
$NotExist(A, B, ...Z)$	$(A), (B), ..., (Z)$	$S(P) = U$

Table 2.3: Syntax of *notExist* Extension to SWRL [5]

Implementation and Efficiency: As shown in the example rules 2.17, 2.18 and 2.19, implementation of the *notExists* operator is simple and usually depends on the side of the inference operator it appears. A notable efficient use of the operator is where there are conflicting facts in a knowledge base, the use of *notExists* operator help in retraction of a rule to restore consistency. As such, the concept of rule's 'precedence' and 'rule retraction' is strongly advised in the inference process of [5]. Consequently, this paves the way for our next SWRL extension — the *dominance* and *mutex* operators in the next section.

2.9.3 Rules Ordering and Priority Extensions

Introduced to define relationships between rules and their semantics, the 'dominance' and 'mutex' constructs are extensions meant to control the behavior of SWRL rules execution.

Expressiveness: Precisely, the dominance operator is used to specify the order of rule execution by assigning a precedence of one rule over another. In contrast, the mutex operator is non-symmetric and designed to assert the complete exclusion of a rule due to the execution of other rules.

Syntax and Semantics: As defined in [121], the dominance operator is defined as *dominance* $(R_x,\ R_y)$, where R_x and R_y are rule names, and with a semantic meaning that "rule R_x has more priority in the execution order than R_y". Furthermore, the dominance operator is designed to be transitive, thereby suitable for handling the addition of new rules into a rule base.

As an example, consider the execution of rules 2.17 and 2.18 above. We may want to run the 'Mixed-Status' rule (rule 2.18) first to check if there are married participants in the knowledge base before asserting the 'Singles' booking status in rule 2.17. To set this ordering, we can write the dominance operation as follows:

$$dominance\ (Rule_{10},\ Rule_9) \tag{2.20}$$

Whereas in its syntactic format, the mutex operator is written thus: *Mutex* $(R_x,\ R_y)$, asserting the fact that "rule R_y will not be executed in the event that rule R_x is already executed". Continuing to the hypothetical example above, we may want to completely skip executing the 'Singles Only' rule (2.17) in the event that the booking status is explicitly known to be 'Mixed'. This can be achieved that through thus:

$$Mutex\ (Rule_{10},\ Rule_{9}) \tag{2.21}$$

Implementation and Efficiency: However, these extensions requires specially modified Reasoners for inference and are yet to be formally accepted as part of standardized SWRL definition.

2.9.4 Existential Extensions — Dealing with Quantification of Individuals

As explained earlier, owing to DL-Safety restrictions, SWRL-enabled ontologies can not introduce new individuals (see Section 2.8.2.1). Moreover, with SWRL being the combination of OWL-DL and DataLog RuleML — which do not allow existential quantification in its consequent, it becomes even more difficult to assert new individuals into a classical OWL/SWRL ontology. However, as the authors of [122] puts out,

> ”It sometimes becomes necessary during inference and where certain conditions are met, to introduce new individuals to an ontology”

Addition of new individuals to a knowledge base is commonly referred to as ’existential quantification’ and in what follows, we discuss the various SWRL extensions proposed for achieving that.

2.9.4.1 The X-SWRL Extension

Expressiveness: To achieve existential quantification in SWRL, new operators were defined and the new extension referred to as extended semantic web rule language (XSWRL) [122]. The direct extension defines new operators to achieve the addition of new individuals to existing classes of SWRL ontologies.

Syntax and Semantics: XSWRL uses similar syntax and semantics of the classical SWRL with the only difference being that rules in XSWRL use an additional operator that allows for introducing new individuals. This is achieved through the use of ’existentially quantified variables’ in the rule’s consequent — represented by prefixing them with an exclamation mark (!). Universally quantified variables are represented with the usual question mark (?), as in the original SWRL definition. As an example, consider the rule assertion that *"All members of a project Workgroup should be Engineers"*.

$$Workgroup(?g) \longrightarrow Engineer(!e) \wedge hasMember(?g, ?e) \tag{2.22}$$

Implementation and Efficiency: From the XSWRL's prototype implementation, presented in [122], it can be understood that by defining the semantics of '!' (the existential operator) into the SWRL abstract semantics, reasoning over XSWRL rules can be achieved using existing DL-Reasoners such as Racer and Fact++. However, it remains to be seen of such implementation and whether the extension will be practically utilized by the semantic web community. This is because: contrary to the classical SWRL, XSWRL rules with non-atomic consequents and having joint existential variables cannot be split into multiple rules with atomic consequents. This, makes it difficult to assign rule-preference or prioritize rule execution of XSWRL rules. Recall, that the concept of DL-Safeness ensures decidability in SWRL rules by limiting the consequent variables to only those that previously occurs in the antecedent. As such, further restrictions must be placed on the use of the existential variables declared in XSWRL to ensure decidability. To this end, the authors declare the added restriction:

> "Do not construct a rule, which has existentially quantified variables, to form acyclic chain between its atoms or with atoms from other rules"

. Though this may restrict the expressiveness of the XSWRL language, but it ensures that infinite chains are safely avoided. Hence, XSWRL rules need to be tracked manually to ensure that their executions will not lead to a cyclic chain of existential quantification. However, this manual restriction, can be overly tedious or impracticable in very large ontologies.

2.9.4.2 SWRL First Order Logic Extension (SWRL-FOL)

Expressiveness: In an attempt to extend SWRL towards the expressiveness of First-Order Logic, notably to achieve the quantification of individuals, a SWRL-FOL extension was proposed in [123]. Analogous to the FOL Rule Markup Language [6], the proposal defines an abstract syntax and semantics for the SWRL-FOL extension. It shows how SWRL can be extended, to utilize the expressive powers of FOL, by extending the component OWL axioms to include function-free FOL assertion axioms.

Syntax and Semantics: SWRL FOL proposal defines an abstract syntax of the expressive extensions and further provided the model theoretical semantics for their interpretations. In their abstract syntax, SWRL-FOL ontologies contain sets of OWL axioms, facts, and horn-clause rules with additional FOL axioms or assertions. These assertions however, introduces some extensions to the original SWRL format, such as the limitless use of 'conjunctions' and/or 'disjunctions' in the FOL formula and the use of constructs such as *negation, 'ForAll', 'Exists'*, etc. over unary and binary predicates. However, the semantic interpretations of these assertions are defined as 'bindings' which maps every variable to an element in the domain. See SWRL built-ins extensions in Section 2.9.6 for more details on predicate bindings.

Example: Consider the Workgroup members assertion in rule 2.22. An alternative expression using the direct *Exists* element defined in FOL-SWRL can be written as:

$$Workgroup(?g) \longrightarrow \exists_m Member(?m) \ \wedge \ Engineer(?m) \tag{2.23}$$

Implementation and Efficiency: What we noted here is that the proposal in [123] is largely theoretical and as the authors of [122] puts out, its practical implementation is still open for discussion. This however, is due to the usual concerns of decidability and the lack of Reasoners that can achieve inferences over the FOL sentences. Nevertheless, considering its extensive definition and the general uti-

[6]http://ruleml.org/fol/

lization of FOLs, no doubt the SWRL-FOL extension can help to provide a good foundational framework for FOL-based SWRL extensions.

2.9.4.3 SWRL Constraints Interchange Format (CIF-SWRL) Extensio

In their motivation, the authors of CIF-SWRL [124] explains that knowledge fusion in an open distributed environments such as the semantic web, involves data gathering from various network sources, which also include the constraints on how the data can be used. As such, utilizing these constraints directly using the SWRL rules can help to achieve existential quantifications. This lead to the proposed CIF-SWRL — an extension of SWRL to express fully quantified constraints.

Expressiveness: the Constraint Interchange Format (CIF) extension of SWRL (CIF-SWRL) is an advancement of the SWRL formalism towards the constraint satisfaction problem (CSP). While in most cases, the target is simply to improve the SWRL formalism (rule layer) for better domain modeling, in CIF-SWRL extension, the objective is to improve SWRL to handle CSPs in the Logic layer of the Semantic Web and the aim is to allow the quantification of these constraints so that new individuals can be introduced into the knowledge base.

Syntax and Semantics: To allow expressing fully-quantified constraints using the CIF-SWRL rules, the proposal aligns the original features of CIF with the SWRL definition to form a single modeling language. In essence, the extension introduce constraints — defined as quantified implications, to solve the problem of existential quantification in the classical SWRL. These constraints are expressed using an interchange format, such as the First-order logic (FOL)-based CIF, to form relevant constraint satisfaction problems that serve as inputs to constraint solver. To illustrate the CIF/SWRL added syntax, we quote the following rule assertion: *”Every workgroup must contain at least 1 member who is a Professor”*

$$(\forall ?g \in Workgroup) \longrightarrow (\exists ?p \in Professor) \wedge hasMember(?g, ?p) \quad (2.24)$$

Meaning: For all Workgroup g, there exists a Professor p, who is also a member of the workgroup. The rule above shows the directly added symbols, 'ForAll (∀) and Exists (∃), which are both subsets of a 'Quantifiers' class — defined in the CIF-SWRL expressiveness extension.

Implementation and Efficiency: The CIF-SWRL extension comes about as an advancement of an existing CSP, which uses OWL ontology as a data model and SWRL rules for expressing constraints. Beside the use of quantification symbols, CIF/SWRL also claims to introduce nested quantified implications, which supposedly allows for multiple-quantification of individuals in a rule. The proposal features the technical details of CIF-SWRL extension, including the abstract syntax and semantics, with an illustrative application to a use case. However, the featured implementation does not fully explain a reasoning strategy for inference on CIF-SWRL rules. Other forms of non-monotonic extensions, precisely the representation of 'disjunction' and 'negation', were also mentioned as work-in-progress.

The Semantic Web Constraint Language (SWCL) Recognizing the need to improve the CIF/SWRL extension to handle more than just logical constraints, a complimentary extension of CIF-SWRL called the Semantic Web Constraint Language (SWCL) was proposed in [125] and subsequently applied in [126]. The rationale being that by incorporating other constraint satisfaction problems. such as the mathematical constraints, CIF-SWRL will can extend its applicability to problem areas beyond simple decision problems. The SWCL is then defined as an OWL-based extension for modeling mathematical constraints to solve basic optimization problems in the semantic web. It includes the proposal of a Unified logic and Constraint problem solver — alleged to be a semantic-web-based decision-making framework for implementing case scenarios using the SWCL.

2.9.4.4 SWRL Epsilon Extension

Proposed in [127], the *Epsilon* existential extension of SWRL is another extension to the SWRL syntax and semantics — designed to allow quantification of individuals into OWL ontologies using SWRL rules.

Expressiveness: In this SWRL existential extension, the authors uses a new operator the Hilbert's Epsilon (ϵ), to denote existential quantification. However, the main objective of the proposal is to achieve structural computation in OWL and subsequently, in the SWRL formalism.

Syntax and Semantics: The epsilon operator is employed to define a 'terminology constructor' ($\epsilon_X \phi(X)$), which on inference, is expected to return anonymous individuals as 'values of class X' as results of an existential formula function $\exists_X : \phi(X)$. The syntax of the epsilon extension is defined by simply extending the abstract syntaxes of the 'i-objects' and 'd-objects' of SWRL to represent the epsilon terminology. While highlighting the limitations of procedural attachments such as SWRL built-ins, and the inefficiencies involved in using first order logic extensions, the semantics of the epsilon extension were described in [127] with theories and technical case study showing the use of the new operator. However, the discussion failed to show the usability of the new operator in SWRL rules and no supporting reasoners were mentioned.

Implementation and Efficiency: In order to utilize the epsilon extension, an in-depth understanding of the theory of Hilberts Epsilon operator seems to be inevitable. As no example SWRL rules were presented on how to utilize the new operator, implementation of the epsilon extension is still open for discussion. Moreover, the new operator seems to only increase the complexity of the SWRL formalism in achieving existential quantification when compared to the previous existential extensions discussed.

In addition, it should be noted that a turn-around fashion of creating new individuals in SWRL rules is possible with the use of the followings: (i) the OWL class construct *owl:someValuesFrom* — thereby achieving existential quantification as class assertions. (ii) Alternatively, the SWRL built-in *swrlx:makeOWLThings* (discussed as part of SWRL Builtin extensions in Section 2.9.6), also allow direct creation of individuals in SWRL-enabled ontologies. Though, inference on the *swrlx:makeOWLThings* built-in will result in free individuals that cannot be classified into the ontology. (iii) SWRL can also achieve existential quantification using the restriction *owl:someValuesFrom* to directly creates new individuals into OWL classes. Even though the resulting quantification does conform to DL-safeness, such existential formulation using a restriction can be hard to implement in practice and has been criticized as being inconvenient.

2.9.5 SWRL Extension for Advanced Mathematical Support

As a semantic web rule language, modeling knowledge for, and across, all types of domains should be possible using the SWRL formalism. This includes complex mathematical equations, typically used in engineering applications. While SWRLs mathematical built-ins support the basic arithmetic operations such as addition, subtraction, comparison, string and Boolean operations, etc. There is, therefore, a need for extending SWRL to handle complex mathematical and engineering computations such as polynomials, integration, differentiation, summation, etc.

2.9.5.1 The OpenMath Extension of SWRL

To this end, Lopez and others in [128] propose the combination of SWRL built-ins with the 'OpenMath'[7] model to provide advanced mathematical support in SWRL. The OpenMath is an extensible representation standard as well as an evolving interchange framework for sharing and publishing mathematical objects and their semantics. It is basically, a markup language and a representation standard for mathematical objects.

Expressiveness: The aim of SWRL-Openmath extension is to allow expressing scientific knowledge using formulas. Using the functionalities of OpenMath, an additional SWRL built-in *swrlbext:mathext* (with three basic arguments) was designed to extend the SWRL built-ins ontology class. In essence, the 'SWRL-OpenMath' extension extends the SWRL formalism with a feature that evaluates and reason over mathematical expressions. By using the built-in (swrlbext: mathext) to define functions as instances of the Formula class, the OpenMath extension enables the representation of complex math operations such as integration, differentiation, polynomials, etc.

[7]http://www.openmath.org/

Syntax and Semantics: The mathematical expressions are represented using the OpenMath XML functions. The function parameters, which are assumed to be already defined in the OWL ontology, are then supplied as part of the SWRL rules. To allow reuse of the math functions, a 'Formula' class is defined to represent the OpenMath expressions as datatype values using a special datatype property called *hasOMExpression*. The result of the expression i.e. the parameters supplied and the formula itself, comprises the three arguments defined in the built-in extension *swrlbext:mathext* as described in the proposal.

Implementation and Efficiency: Apart from utilizing complex operators and their semantics, the use of OpenMath instead of the classical mathematical operators also helps to separate the mathematical and problem semantics in writing SWRL rules, thereby giving more clarity to rules representation.

However, the short proposal only gives an overview of the extension with a draft of a methodology showing how the combination can be achieved and mention of possible implementation using 'Bossam' and 'Mathematica'. Moreover, neither the implementation details nor testing of the extension can be found. In our opinion, the OpenMath extension introduces a rather complex approach to handle formulas in SWRL as compared to mathematical built-ins approach. In addition, the proposal failed to discuss reasoning supports for the OpenMath SWRL extension and is curiously silent on the overall decidability of the combination.

2.9.6 SWRL Built-in Extensions — Addressing SWRL limitations through Built-ins

Discussion on SWRL extensions can never be complete without mentioning the SWRL built-ins[8]. Simply put, SWRL built-ins are procedural attachments used to augment the expressive powers of the original SWRL language definitions. More formally, a SWRL built-in is defined as "a predicate that takes one or more variables as arguments and evaluates to true if the argument satisfies the predicate". SWRL built-ins consist of the 'core built-in libraries' for common operations involving constraints, lists, string, comparison, Boolean, URIs, and date operations

[8]http://www.daml.org/2004/04/swrl/builtins.html

— preceded by the namespace qualifier *swrlb:*. Moreover, SWRL built-ins are especially useful as they allow special definition of domain-specific, arbitrary methods called 'user-defined built-ins' — an important feature of the SWRL formalism that allows users to define new built-in libraries for special tasks. Core built-ins such as the mathematical operators, and built-ins for string and date operations were defined in the original SWRL specification.

Implementation: Defining SWRL built-in extension is possible either directly in OWL or through their corresponding java implementations made possible by the *SWRLBuiltInBbridge*. The *SWRLBuiltInBbridge* is a component of the open-sourced SWRLTab of the Protégé ontology editor, which allows the manipulation of SWRL built-ins using Java. User-defined OWL-based built-in definitions can be achieved by simply adding them as new instances to the *swrl:Builtin* class pre-defined in the SWRL definition ontology. Relevant built-ins are usually grouped together in a single OWL file, which can be imported into any domain ontology for utilization. A Java implementation of the built-ins, wrapped in a JAR file is however needed in the Protégé-OWL plugins directory for the Built-in bridge to make the necessary run-time linkages. A good example is the SWRL-Inference and Query tool, popularly termed as the SWRL-IQ — originally defined as a "plugin for Protégé version 3.x that allows users to edit, save, and submit queries to an underlying inference engine based on XSB Prolog." The built-in extension (swrl-extension.owl) basically contains user-defined predicates, implemented for use in the SWRL Inference and Query Tool (SWRLIQ).

In what follows, we briefly review some of the most popular as well as standardized SWRL built-in extensions, such as the temporal built-ins, the mathematical built-ins, the semantic web query language (SQWRL), and the Existential built-ins of SWRL:

2.9.6.1 SWRL Temporal Built-ins

Due to the limited temporal support in both OWL and SWRL, another notable example in the SWRL expressiveness extensions is the SWRL Temporal Built-in Library [129].

Expressiveness: Defined as part of the SWRL-API's built-in library, the temporal built-ins are hierarchically defined in the SWRL temporal ontology. The SWRL temporal model is designed to allow easy representation of temporal knowledge in SWRL-based ontologies. The temporal ontology provides a standard model for modeling the temporal domain facts. As a result, the built-ins allow temporal reasoning on OWL ontologies using SWRL rules.

Syntax and Semantics: The temporal built-ins provide a rich set of temporal operators such as *before, after, during, duration, contains, overlaps,* etc. and are normally preceded by the name-space qualifier 'temporal:'. Based on the time data they operate, the SWRL temporal built-ins were categorized into basic and advanced mode. In the basic mode, SWRL temporal built-ins operates on arguments supplied by the XML Schema's 'date' and 'dateTime' data types — supplied as *xsd:String* with values such as second, hour, day, time, week, month, year, etc. Whereas in the advanced mode, the SWRL temporal built-ins works on time information that is completely encoded using the 'valid-time' temporal model .

As an example, a rule that asserts the 'Fellow' membership rank by categorizing all registered Workgroup members with registration dates before the year 2000 can be written as:

$$WorkgroupMember(?m) \wedge hasRegDate(?m, ?rd) \wedge \\ \textbf{temporal:}before(?rd,'2000') \longrightarrow FellowMembers(?m) \quad (2.25)$$

2.9.6.2 SWRL-M Built-ins — A Complex Mathematical Built-in Library.

Expressiveness: Apart from the basic arithmetic operations available in the original SWRL definition, a SWRL-M built-in extension, was recommended to allow SWRL rules to handle more mathematical expressions. Defined as part of the SWRL Inference and Query Languages (SWRLIQ) [130], the extension introduces complex math operations as an advancement to the core mathematical built-in library. It is efined as part of the SWRL Mathematical Ontology and written with the prefix *swrlm:* as its pseudonym.

Syntax and Semantics: Common mathematical operations allowed in SWRLM built-in library include among others; the square root operation, *swrlm:sqrt(?x, a)* and the evaluate expressive function, *swrlm:eval(?x, "expression").* The latter is designed to support the evaluation of the ontology variable x against standard constants and functions such as pi (π), epsilon (ϵ), Lin ($\ln$), etc. For example, a SQWRL query that "returns a random number between 0 and 1" can be written as:

$$Swrlm : eval(?x, "rand()") \longrightarrow sqwrl : select(?x) \tag{2.26}$$

Efficiency: The SWRL-M built-in provides a simple and efficient way of dealing with complex math operations — especially when compared to the OpenMath extension (Section 2.9.5.1). However, the simplicity also has its price as the SWRL-M built-ins collection falls short in representing many advanced mathematical expressions, some of which are highlighted in Section 2.9.5.

The restriction is a classical issue of modeling languages, i.e. the need to balance between tractability and degree of expressiveness. And as mathematical extensions involving recursive functions in rules usually makes inference non-terminating — and thence entailment undecidable, there is the need for careful design of built-in functions to ensure that predicates introduced remain decidable.

2.9.6.3 SWRLX Built-ins — The SWRL Existentials Built-in Library

Considering the need to create new individuals using the classical SWRL definition and without relying on external extensions, the SWRL existential built-in (*swrlx:makeOWLThings*) is defined in [131]. It is defined in the SWRLX Ontology and written using the 'swrlx:' prefix. The built-in is designed to ease explorative modeling in SWRL. Specifically, the existential quantifications — see Section 2.9.4. Creating new individuals using SWRL built-in can be of particular importance, especially in the execution of mapping rules.

Syntax and Semantics: Using the SWRLX built-in model, creating new individuals is made possible using the *swrlx:makeOWLThings* method, which has at least one free variable as its argument. The semantics being that the the method will create a new individual of type *owl:Thing* and binds it to the free variable in the its argument. In essence, the axiom: *swrlx:makeOWLThings*(?x, ?y) — ”will cause a new individual to be created and bounded to ?x for every value of the matching variable ?y in the rule”.

Now consider an example SWRLX rule (2.27) that: *asserts a new individual for every membership ID of the WorkgroupMembers class and then asserts the new individual into the Editors class*

$$WorkgroupMember(?m) \wedge hasMemberID(?m, ?mID) \wedge \\ \textbf{swrlx:makeOWLThings}(?ed, ?mID) \longrightarrow Editors(?ed) \quad (2.27)$$

Efficiency and Decidability: Creating new individuals using SWRL built-in can be of particular importance, especially in the execution of mapping rules. The advantage of *swrlx:makeOWLThings* method over other existential quantifications is that it offers a simple and direct method of creating new individuals in SWRL-based ontologies. However, as the new individuals created will simply be of type ’owl:Thing’, it cannot be further classified by a reasoner into any particular class type. This may result in redundant unbounded individuals in the main ontology. As such an efficient and careful use of the built-in requires that a class be predefined in the ontology to collate the new individuals created. Another safe implementation, as the authors advised, will be to completely avoid storing the new individuals into the main ontology. However, SWRL built-ins still remains limited in extending the semantic web rule language to represent non-conventional domain knowledge. While those already defined, could benefit from well-documented constructs with more efficient syntax and semantics.

To summarize, SWRL built-in definitions no doubt increases the expressiveness of SWRL and the possibility of user-defined SWRL built-ins means that domain-specific extensions can always be defined to extend SWRL’s expressive powers. Other built-ins not expanded in our discussion here, includes the query extension built-ins such as the OWL-Axioms: the T-Box, A-Box, and R-Box built-in libraries, which allows querying knowledge stored in the: terminology box, the assertions box and the Relations sets of OWL ontologies. Note: special query built-ins were grouped together to form the Semantic Query Web Rule Language (SQWRL) built-ins [102]. In our opinion, improved syntax of the various built-in extensions with well-defined semantics that can be inferred within the DL-safety restriction, is highly desirable. As that will help to improve their usability.

2.9.7 Summary of SWRL Language Extensions

As categorically summarized in Table 2.4, various extensions to SWRL syntax and semantics have been proposed and justified as necessary expressiveness extensions of the rule language. While we present them as six categories for clarity, the extensions can be basically summarized into four categories, viz. the (i) Uncertainty management extensions — comprising of the fuzzy and probabilistic extensions, (ii) the Non-monotonic extensions — comprising of the negation, quantifiers, rules ordering and prioritization, as well as the existential extensions, (iii) the Advanced mathematical extension featuring the SWRL-OpenMath extension, and lastly, (iv) the SWRL Built-in extensions.

SWRL Extension	**Added Syntax**	**Added Semantics**	**Extension Type**
Fuzzy-SWRL (f-SWRL)	Fuzzy class assertion: C(x) * w. Fuzzy property assertion: P(x,y) * w. Where: w $\in [0,1]$.	Introduces a truth value 'w', to specify degree of confidence for individual membership in a Class or Property.	Fuzzy Extension
Vague-SWRL	Introduces a second-degree weight, w_2 to f-SWRL syntax.	Denote second degree of membership: w and w_2 specifies the upper and lower bounds of membership intervals in a Class or Property.	Fuzzy Extension

SWRL Extension	**Added Syntax**	**Added Semantics**	**Extension Type**
SWRL Fuzzy (SWRL-F)	Introduce a fuzzy matching operator: *fuzzymatch*(?x, 'fuzzy-value')	Matches corresponding fuzzy variables with designated fuzzy values from fuzzy sets.	Fuzzy Extension
Fuzzy Non-monotonic SWRL (f-NSWRL)	Fuzzy weight (w), 'Not' and '$\neg$' operators.	Same as the F-SWRL and Negation extensions.	Fuzzy-Non-monotonic Extension.
Bayes-SWRL	Probability weight (the p-variable, p) Class assertion: C(x) * p_x. Property assertion: P(x, y) * p_{xy}. Where: p [0, 1].	The p-variable (p) matches the probability of Class or Property assertions with predefined values in a conditional probability table (CPT).	Probabilistic Extension
Negation	The 'not' and '$\neg$' operators	Negation of existing concepts and asserting negative facts	Non-monotonic Extension
SWRL Quantifiers	Exists ($\exists_X$) operator	Asserts new instances of the quantified variable x.	Non-monotonic
	notExists ($\nexists_X$) operator in antecedent	Check for missing facts.	

SWRL Extension	**Added Syntax**	**Added Semantics**	**Extension Type**
	notExists ($\nexists_X$) operator in consequent	Removal of Knowledge	
	forAll ($\forall_X$)	Group-wise assertions	
Dominance	Dominance operator: $dominance(R_x, R_y)$, R = rule identifier	Implies: Rule R_x has more priority in the execution order than R_y	Rules Ordering $(Priority)$
Mutex	Mutex operator: $Mutex(R_x, R_y)$, R = rule identifier	Implies: Rule R_y will not be executed in the event that R_x is already executed	Rule Exclusion
Extended SWRL (XSWRL)	Existential operator'!' e.g. existentially quantified variable (!x)	Creates new individuals that satisfy the variable, x.	Non-monotonic Existential Extension
SWRL-FOL/ CIF-SWRL	The 'ForAll ($\forall$)' and 'Exists ($\exists$)' operators	Imposes first order logic (FOL) quantification operations on the SWRL variables	FOL Extensions
Epsilon Extension	The terminology constructor $\epsilon_X \phi(X)$	Creates new individuals as 'values of' class X.	Quantification extensions

SWRL Extension	**Added Syntax**	**Added Semantics**	**Extension Type**
SWRL OpenMath	The math built-in: $swrlbext : mathext$	Uses the functionalities of OpenMath to create additional built-ins	Advanced Math Ext.
SWRL Temporal	Duration operations, Allen's Temporal intervals, add/subtract operations e.g. temporal:before(T_1, T_2)	Implements temporal operations as predicates on valid-time data	Built-in Extensions
SWRLM	Complex Mathematical operations (evaluate, square-root, natural log) e.g. $swrlm : eval$(?area, "width * height", ?width, ?height)	Implements mathematical functions as predicates	Built-in Extensions
SWRLX	Existential Built-in e.g. $swrlx$:make-OWLThings(?x, ?y)	Creates new individuals of type $owl : Thing$	Built-in Extensions

Table 2.4: Table of Summary for `SWRL Extensions`

In the fuzzy extension category, we discussed the SWRL-F, Vague-SWRL, F-SWRL, and FNSWRL, all of which were aimed at extending SWRL to enable the representation of uncertainties or incomplete information. The extensions basically involve the use of fuzzy logic principles, specifically the fuzzy and vague sets, to extend the description logic of the SWRL formalism. Their similarity is apparent in their use of the fuzzy weight to denote the degree of membership for individual instances of the fuzzy class or property within a SWRL axiom.

An exception to this similarity is the 'F-SWRL extension' — which introduces a *fuzzymatch* operator for representing uncertainties based on the principles of fuzzy control system. A probabilistic extension of the SWRL formalism, 'Bayes-SWRL', was also discussed as another uncertainty-handling or predictive modeling extension of SWRL. Bayes-SWRL uses the Bayesian Networks of probability theory to manage predictive knowledge modeling using SWRL rules. See Section 2.9.1.5 for details.

Various non-monotonic expressiveness extensions of SWRL have been geared towards solving the classical negation and the negation as failure (NAF). The negation extensions were included here only for completeness as the issues have been thoroughly addressed both in the mainstream OWL and the SWRL's abstract syntaxes. It is safe to assume that these numerous proposals with their justifications lead to the advancement of the abstract definitions to handle the knowledge negation. Other expressiveness extensions discussed in the non-monotonic category include the existential extensions — introducing existential operators with syntax such as: the exclamation mark '!' in the *X-SWRL extension*, the existential ($\exists$) and Universal ($\forall$) quantifiers in Section 2.9.2, the 'ϵ operator in the SWRL epsilon extension, and the *makeOWLThing* operator proposed in the SWRLX built-in definition. All these extensions were recognized to allow a safe creation of individual instances to an OWL class or property, except the *makeOWLThing* built-in — where the resulting instance is beyond the inference of existing OWL reasoners.

As domain knowledge is ever-evolving, there is sometimes the need to control the execution of rules or even retract some facts based on new found information. These challenges were addressed through rules ordering (using the 'dominance' operation), rule exclusion (using the 'mutex' operation) and facts removal, using the notExists operator ($\nexists$). These extensions (See Sections 2.9.2—2.9.4), which obviously entails the essence of non-monotonicity, were thence summarized in Table 2.4 as non-monotonic extensions of SWRL.

Advanced mathematical extensions of SWRL language through built-ins were also discussed. The category introduces a special built-in extension based on the OpenMath language syntax and semantics. The 'SWRL-OpenMath' extension involves the use of *OpenMath* functionalities to enable expressing mathematical formulas and scientific equations in SWRL. Another more classical approach in this categoryis the 'SWRLM extension', which introduces advanced math operations, beyond those in the core SWRL definition. SWRLM extension handle operations such as the square-root (swrlm:sqrt), natural log (swrlm:ln), etc. Another important built-in extension is the SWRL Temporal extension, which do not directly falls under any of the above four categories, was also discussed to highlight how time-related domain information and temporal facts can be expressed using SWRL rules. See Sections 2.9.5—2.9.6 for details. We present a compacted summary of these extensions and their component syntax and semantics in Table 2.4 below.

2.9.8 Decidability and Completeness of the SWRL Expressiveness Extensions

From the foregoing review of the various expressiveness extensions added to SWRL, one may be interested in asking the question, 'What could be the largest decidable extension of the semantic web rule language?' and while an obvious answer could be 'the DL-Safe extensions', there is still the need to evaluate which assemblage of these extensions can still remain decidable. Where the combination is no longer DL-safe, then what restrictions could be imposed for the combination to remain decidable. This is particularly important in achieving inference and to ensure usability of these extensions.

In practical terms, *Decidability* refers to the ability of a Reasoner to classify and achieve inference over a given piece of ontology within a finite time. While basic SWRL rules can be kept decidable through the DL-Safety restriction, most of the SWRL expressiveness extensions can hardly be kept decidable by employing similar restriction. Such extensions that work within the fringes of decidability, such as the existential extensions, need to be carefully tailored to ensure that their usage do not cause inconsistencies to the resulting ontology. For example, the XSWRL extension [122], which adds new individuals using existentially quantified variables is clearly non Dl-safe and therefore, undecidable. This is because, unbounded variables are bound to be introduced into the ontology, possibly creating anonymous individuals or cyclic chains in rule execution. As such for the

XSWRL extension to be decidable, the authors make the following suggestion:

i. ”Do not construct a rule which has existentially quantified variables to form acyclic chain between its atoms”.

ii. ”Do not construct a rule which has existentially quantified variables and other rules to form cyclic chain among their atoms”.

In what follows, we briefly discuss the conditions needed to maintain the decidability of the reviewed SWRL extensions:

On Decidability of the Non-monotonic SWRL Extensions: As the authors of [5] described, decidability issues involving unbound variables can be resolved through careful management of their syntax or semantics during the inference process. For example, in the case of the ’not’ Operator — negation as failure (see Table 2.2). Here, decidability is achieved by controlling the appearance of the unbound variable in the consequent of the SWRL rule. In essence, the unbound variable $(?z)$, whose inference can result in anonymous negative property assertions from the new elements $not(P(x;y))$ and $not(Q(x;y))$, was deliberately introduced so as not to use the bounded variables $'?x'$ or $'?y'$ in the consequent of the extended SWRL syntax. In effect, while the negation of the object or datatype property assertions P or Q holds, the resulting unbound variable $(?z)$ that hold this momentary value will not appear in any other case or rule atoms and therefore the knowledge base remains decidable.

Similarly, in order to preserve the decidability of the main ontology while using the proposed NotExists ($\nexists$) quantifier (see Table 2.3), the unbounded variable introduced is semantically defined to match all the individual IDs available in the ontology. In other words, the free variable is interpreted to be ’universally satisfiable’. This condition ensures that the execution of the operator does not result in any modification of the existing ontology and therefore remains decidable.

Other forms of the non-monotonic SWRL extensions, such as the ’Mutex’ and ’Dominance’ operators, do not have a direct consequence on the content of the ontology and therefore may not affect its decidability or otherwise. However, the Mutex and Dominance operators, which controls and prioritizes the sequence of rule executions respectively, can be used to control the Decidability of SWRL ontologies during the inference process. This is because, by enabling the Reasoner to block and/or schedule rule execution plan, they mimic the feature of the inference process. As such, undecidable rule fragments can be dominated or muted where necessary to achieve consistency.

Decidability of Fuzzy SWRL Extensions: Theoretically, the decidability of f-SWRL extension is not debatable considering that the extension simply introduces weighted values (see Section 2.9.1.1) to justify the importance of certain facts over others. Specifically, the class assertion, property assertion and rules axioms of SWRL were extended to show the degree of confidence of such assertions. As such, f-SWRL ontology — and by extension the Vague-SWRL ontology, can be assumed decidable as long as their OWL class and property axioms conform to the DL-safety restrictions. While the semantics of the fuzzy axioms, such as class and property inclusion axioms may have different interpretations from the original SWRL, the concepts introduced in most of the fuzzy-based SWRL extensions reviewed, do not seem to invalidate the DL-Safeness principle and can therefore, be expected to remain decidable. This can be seen from the fact that no new concepts or anonymous individuals are expected from these extensions nor the interpretation of their syntax — which in most cases is closely similar to the original SWRL. For example, the comparable SWRL-F extension (see Section 2.9.1.3) introduces the concept of fuzzy ontology as a separate entity to maintain consistency. As such its inference has no direct influence to the consistency of the main ontology. Inherently, the Decidability of this extension also follows the DL-Safety restriction of the SWRL rules.

Decidability of SWRL First Order Logic (FOL) Extensions: FOL-based languages are generally undecidable and logical formalisms extending FOL, such as the SWRL-FOL, are usually so — unless where they are restricted by some semantic or syntactic completeness theorems. The SWRL FOL extension(see Section 2.9.4.2) introduces assertion axioms that contain first-order formulae. However, the restriction is that quantified variables must be bound to their corresponding OWL typed quantifiers — meaning that no ’free’, ’anonymous’ or

’unbounded’ variables should exists in the rules. In such cases, the decidability of SWRL-FOL extension can also be said to be inherent in the DL-safety restriction of the component SWRL ontology. As the authors of SWRL introductory paper [70], expressed regarding the semantic interpretation of FOL assertions; ”An ontology is consistent if-and-only-if it is satisfied by at least one interpretation” Moreover, neither n-ary predicates nor functions were directly included in the SWRL-FOL abstract syntax as they do not fit with OWL and by extension the SWRL paradigm.

Decidability of SWRL Built-in Extensions: Built-in extensions basically means that the extension utilizes the abstract syntax and semantics of the SWRL rule language. As such, poses no decidability issues as long as the extensions are used within the DL-safety limits.

Most of the SWRL mathematical and temporal extensions were added as built-in predicates to the original SWRL definition in order to avoid the decidability and complexity overheads. Other cases, where major extensions were added, such as the ’OpenMath’ extension discussed in Section 2.9.5.1, the functionalities were strategically added based on a distinct separation between the mathematical and problem semantics. This is possible considering the fact that the mathematical functions operates on existing domain knowledge as input. Moreover, as the OpenMath extension also uses the built-in (*swrlbext: mathext*) to define functions as instances of the Formula class, which are then solved using a constraint solver. Due to the modularity, the main ontology is thus free of the inconsistencies that may arise during math operations. Hence the extension basically follows the decidability of basic SWRL built-ins, the DL-safety.

2.10 Ontology Utilization: Ontology-based Semantic Search Engines

Information Retrieval (IR), which involves searching and/or browsing of stored data, is the process of discovering specific portion of a large collection of stored information that satisfies certain user requirements [132]. Various forms of information searching for logically stored data do exist and are described in various terms.

2.10.1 Keyword-based Information Retrieval

The commonest IR method is the *keyword search*, a strategy frequently employed by traditional search engines, where indexed documents can be retrieved based on their lexical matches to the search keyword. Another form of search approach, often employed to compliment keyword-based searches, is the *faceted search* [133]. In this approach, to searching or browsing to be precise, information discovery is achieved through filtering of facet values. A faceted search combines the techniques of direct keyword search and a navigational search. A *Navigational search* is a form of directory-based search approach and as argued by the authors of [133]:

> ”The use of faceted searches for querying RDF data is known to have solid background and various theoretical frameworks that work fairly well in the life-sciences domain”

This has lead to the evolution of semantic semantic searching where more specialized domain ontologies are employed to serve as guide to the search engines by defining relevant concepts instead of rigidly matching keywords in the knowledge pool. Semantic search approach is discussed below.

2.10.2 Semantic Searching

Semantic search is an application under the Semantic Web where search engines try to understand the meaning of search terms before exploring the knowledge base for relevant results. By adding semantic tags into documents, and using standard definition of domain concepts usually provided by ontologies, a semantic search is able to return precise search results as it understands the meaning of search keywords and queries performed by the end users. While the general idea of a semantic web is to allow users to search for information from various sources and domains, the concept of semantic search is used to define intelligent searching of information from a single domain [134]. In essence, the *ontology-based* *semantic search* involves defining a possible metadata for the stored contents using ontologies or other semantic annotations to retrieve matching concepts, their relationships and instances via unique identifiers (URIs).

A combination of these two search approaches was proposed in [135], and termed as *Hybrid Search*. The hybrid search simply combines the functionalities by employing the semantic search where search terms are explicitly defined in the ontology and where the ontology falls short of describing the search keyword, the system basically works as a traditional search engine. As our knowledge base also involves RDF datasets that are semantically enhanced and linked together using a global OWL ontology (the UC-ONTO), we simply employ the semantic search approach as detailed in Section 3.4.

2.10.2.1 Ontology-based Semantic Search — Relevant Approaches

Various tools for semantic data exploration and browsing have been developed to allow utilization of available RDF datasets and ontological knowledge bases. In the field of crops and life sciences domain, tools such as CO Curation tool [52], SemFacet [133], do exist among many others. The crop ontology curation tool is a curation and annotation website that provides a search engine for exploring the CO including various URL lists for browsing the ontology. In addition, the tool also allows uploading and creating ontologies in the OBO format with available RDF dumps and a web service API for download. While the CO curation tool may have been motivated by the need to disseminate CO knowledge, a stand-alone tool, SemFacet was developed by Zhou and others, for querying arbitrary life sciences ontologies. According to the authors, the tool allows the use of keyword search as well as faceted navigation for querying ”ontology-enhanced RDF datasets”. The tool’s interface shows a navigation map, which enables search refocusing from one object to another and search results can be filtered based on relevant facet names and values.

Another domain-specific, ontology-based semantic search engine is presented in [136], which is developed for querying agricultural information. The search engine relies on a domain ontology developed in RDF and a web-based interface consisting of a keyword based search engine. With little or no structure, text-based query results are also returned in another text box on the interface. Other more generic semantic search engines include SemSearch [137], SHOE [138], SWSE [139] and Swoogle [140]. The Swoogle search engine, which is one of the early search and metadata engine for the Semantic Web, is able to discover matching keywords from thousands of ontologies, documents, and terms. It allows end users to select preferred data sources before submitting a query and construct relations between resulting documents.

In line with the basic features provided by these systems, the Onto-CropBase semantic search engine presented in Section 4.3 employs the keyword based search engine and complement it with a faceted navigation for browsing the initial search results from the keyword search. Initial search results from the knowledgebase are returned as lists of subjects with navigational links categorized for each data source. In doing so, the user is given a choice to navigate the search results based on available datasets. Details of the Onto-CropBase design and development approaches were described in Section 3.4.

2.11 Ontology Language Extensions Review: Managing Temporal Uncertainties in OWL Ontologies

Conceptual domain modeling in the field of the Semantic Web is often achieved through Description Logic (DL)-based ontology languages such as OWL and is typically guided based on the basic set theory. Modeling imprecise temporal expressions (ITEs) that depend on unstructured vague time data is still a challenge and due to their limited syntax and semantics, current domain modeling languages require that temporal information is asserted as definite time-points. However, such limitations may lead to unfounded approximations and quickly translate to a loss of information in modeling the real-world, especially in application ontologies. Hence, there is a need for a comprehensive ontology language for handling temporal uncertainties (vague expressions of time) commonly found in the real-world narration of domain facts.

In this section, we review the relevant research approaches that handle representation and reasoning about fuzzy-temporal expressions, as well as language extensions that have been proposed to manage temporal expressions with uncertainties in the semantic web community. The aim of which is to extend the semantic web rule language with necessary constructs to manage the modeling of temporal uncertainties in OWL ontologies. We discuss more on our motivations for the Fuzzy Temporal Extension of SWRL (FT-SWRL) in Section 3.5.1.

2.11.1 Fuzzy Temporal Knowledge Representation and Reasoning

On the issue of modeling temporal uncertainties, there are two types of temporal uncertainties to be modeled: the imprecise dating of events and vague descriptions of time data [141]. As a domain modeling knowledge, the fuzzy temporal extension of SWRL aims to manage the latter by providing a formalism for representing imprecise time expressions. Even though there is yet a formally accepted and standardized fuzzy-temporal reasoning system for the ontological knowledge bases, other logically validated fuzzy temporal reasoning systems have been proposed and developed over time. The following works on fuzzy temporal representation and reasoning offer an extensive literature and guide to developing new formalisms such as FT-SWRL.

A fuzzy temporal constraint satisfaction problem is defined in [142] as a new formalism for modeling flexibility and managing uncertainty into the interval-based temporal logic originally defined by J. Allen in [9]. The authors describe a reasoner based on Interval Constraint Network (ICN), which they claimed, can manage both the crisp and fuzzy temporal information containing uncertainties. Basic reasoning tasks described in this work involves the temporal consistency management and temporal query answering. The issue of consistency, as mentioned elsewhere, is important in all logical networks to achieve inference and this includes the fuzzy temporal networks.

A similar approach was presented in [143], where the authors proposed a modeling and reasoning system for managing fuzzy temporal information commonly found in medical records. Here, an existing temporal reasoning system called

’TimeText’, which allow representation of temporal information in clinical texts or narratives was extended to allow uncertain temporal data. The extension also proposed the use of fuzzy temporal constraint network (FTCN) with a proposed solution involving the three-state, staircase possibility distribution function. By exploring the complexity of possibility distributions in solving fuzzy temporal reasoning problems, the work relies heavily on the advances of [144] — where the authors defined a propositional temporal language based on the fuzzy temporal constraints (FTCs). The proposal in [144] describes formal syntax and semantics based on possibilistic models and an inference mechanism based on FTC inference rules, with cited use cases in the medical domain. For more on FTRR systems, we refer interested readers to [145, 146, 147, 148] among others.

2.11.2 Fuzzy Temporal Knowledge Representation in OWL Ontologies

Relevant works on temporal reasoning in the semantic web have focused more on the representation and reasoning of definite temporal information. Basic temporal ontologies are representations of time-stamps to allow modeling time-specific domain information. Most common temporal ontologies represent the basic metadata about time information; specifically, the ’points in time’ data and others are simply domain-specific models for modeling crisp temporal data. Though, fewer efforts have aimed at providing consistent standards for reasoning on the temporal data. Notable efforts in this category includes: the SWRL Temporal [129], temporal OWL (tOWL) [149], the OWL Time Ontology [150] and its extensions such as in [151, 152, 153]. Others include: the Clinical Narratives Temporal Ontology (CNTRO) [154] and CHRONOS [155] — which, as the authors claimed, handles both qualitative as well as quantitate temporal facts. These works (and numerous others) have extensively discusses temporal modeling on the semantic web.

However, real-life temporal information (including expert narratives) are usually inundated with non-crisp temporal description of events, procedures, etc. Nevertheless, the basic temporal models have provided groundworks for defining much of the fuzzy temporal representation models.

A notable fuzzy model for representing temporal uncertainties in ontologies is presented in [156] — which basically discusses the application of fuzzy temporal reasoning over historic data. Following a modular semantic approach, fuzzy set operations were employed to achieve the fuzzification of basic Allen intervals [157] into fuzzy temporal intervals. However, due to its added flexibilities of second-order theory, KAON2 [9] is selected as the target framework for the extension and not OWL directly. Moreover, to efficiently model temporal specifications using the model, an in-depth expertise in fuzzy interval logic is necessary. A factor we try to avoid by introducing the fuzzy temporal built-ins in our FT-SWRL specification. Thus giving users the simple natural language terminologies for modeling imprecise temporal information while hiding the technical implementation details. Moreover, importing the SWRL-FT ontology helps to achieve modeling consistency in the use of the fuzzy temporal SWRL rules.

2.12 Summary

In this chapter, the ontology generation and standardization approaches were explored and the integration of OWL ontologies with LP rules for added expressiveness. It discusses available literatures ranging from Semantic web technologies and the relevant works in ontological knowledge modeling to language extensions of OWL and SWRL formalism.

Based on the reviewed works, ontologies have been widely used as knowledge modeling tools and have been effectively used in the field of life-sciences as Knowledge-sharing and integration tools. Regarding the techniques for ontology generation, we briefly highlight the situations and process of creating ontologies from scratch, followed by the benefits of ontology reuse. Generating ontologies from existing data sources is also discussed and the practice of generating domain-specific concepts through competency questions. These techniques were put to the test in our first case study in Section 4.2, where we give detailed description with examples of our underutilized crops ontology development and standardization. While the ontological knowledge modeling is imperative, of equal importance is

[9]http://kaon2.semanticweb.org/

the standardization of this knowledge. Commonly employed techniques in the early development stage involves reuse of existing standards and alignment of newly generated ontologies with foundational ontologies for standardization. Involving domain experts during the ontology development also greatly improves the quality of the knowledge model and also minimize the inconsistencies in nomenclature. For later stages, continuous update and merging of new concepts into existing ontologies and providing natural language definition for ease of reference, help to ensure ontology standardization. As mentioned earlier, despite the expressive powers of ontologies, rules are necessary for modeling declarative knowledge and for expressing complex roles — such as composite relations between concepts that are not easily expressible with OWL alone. Logic program rules were widely used for integrating ontologies and SWRL rules are shown to be widely accepted for integrating OWL ontologies. Other reasons for the combination of ontologies with rules is the advantages offered by rule formalisms, such as their wide acceptance, the commercialization of rule-based applications and the availability of reasoning engines to provide inferences over their knowledge bases.

From the Semantic Web point of view, which is the driving force for most ontology developments, the use of rules to integrate ontologies was embedded in the imprint of the semantic web stack. This may be a driving point for many researchers and Knowledge engineers to consider embedding rules into ontologies, so as to utilize in full, the technologies offered by the semantic web. Moreover, query writing as well as ease of detailed specification of factual domain knowledge may be considered other driving force for this harmonious integration. When selecting suitable rule languages for integrating with OWL ontology, compatibility is the main concern. The most compatible rule language for extending OWL ontologies is the SWRL, which is basically a rule extension of the OWL itself. Compatible languages result in simplicity in the knowledge modeling and validation process since there is a tight semantic inter-operability between the ontology and the rule language.

The chapter further introduces the SWRL formalism, highlighted its expressive limitations and present a review of the state-of-the-art of expressiveness extensions proposed to tackle these language limitations. The review discusses the various: fuzzy and probabilistic extensions of SWRL, such as the non-monotonic extensions, existentials, advanced mathematic extensions, as well as the SWRL built-in extensions. From the explored literature, it is evident that much of the available SWRL extensions have been directed towards solving three major issues, namely: (i) the uncertainties in domain knowledge, (ii) the non-monotonicity — for handling evolving knowledge bases, and (iii) the modeling of mathematical formulas and scientific knowledge. As the selected rule formalism for extending the underutilized crops ontology (UC-ONTO) [45], the review of common SWRL expressiveness extensions is undertaken with the aim of asserting the SWRL's overall expressiveness and to consider enhancing the SWRL language with further expressiveness for managing temporal uncertainties — imprecise temporal expressions commonly found in (crops) domain knowledge descriptions. To this end, Section 3.5 proposes the pioneer fuzzy temporal extension of the Semantic Web Rule Language.

In the following chapter, research methodologies and contributions in the experimental development of the Underutilized Crops Ontology (UC-ONTO) are presented. This is followed by methodologies for ontology utilization through the Onto-cropBase semantic search tool designed to utilize the UC-ONTO as its knowledge base. Lastly, the modeling approaches and preliminaries of the proposed Fuzzy-temporal extension of SWRL were presented.

Chapter 3

Research Methodology

3.1 Introduction

This chapter presents the ontology engineering methodologies of the Underutilized Crops Ontology (UC-ONTO), followed by the design approaches of the Onto-CropBase semantic search tool and lastly, the model design of the Fuzzy-Temporal Semantic Web Rule Language (the FT-SWRL model). The chapter is organized as follows: Section 3.2 presents the UC-ONTO development methodology highlighting our approach and novel contributions in generating domain ontologies using Competency Questions (CQs). This is followed by the detailed ontology development approaches and design choices in Section 3.3, with illustrations from the domain of discourse and the Protégé Ontology Development Editor (ODE). Section 3.4, presents the notable design choices considered in developing the Onto-CropBase tool, highlighting its knowledge base integration approach, architectural framework, interface design and search engine approach as well as the selection of the mediator component. In Section 3.5, the FT-SWRL model design is presented with preliminary technologies and the pioneer FT-SWRL ontology design highlighting its various entities and hierarchy. Lastly, the chapter is summarized in Section 3.6.

3.2 The UC-ONTO Development Methodology

Comparable to software engineering methodologies, various ontology design strategies have been proposed and practiced over the last decade. Common examples include the *Methontology, Enterprise Ontology development approach, the Onto-Knowledge, and DILIGENT* methodologies [158]. These approaches were presented in details in Section 2.5 among others. In theory, ontology development follows a standard engineering approach that heavily utilizes 'reusability'. However, in practice, considering the indirect nature of knowledge acquisition and modeling, these development methodologies can best be applied according to the ontology requirements and nature of the development approach. As such, other ontology generation approaches were proposed, tested and utilized while others adopted. In what follows, the methodological framework for our ontology engineering was presented highlighting the different conceptualization strategies utilized for efficient domain knowledge gathering. The Protégé ontology editor illustrates the various stages in the ontological knowledge modeling in Section 3.3.

The ontology engineering methodologies discussed in Section 2.5 of the literature include the generic approaches to ontology engineering from the scratch, which were found to be more ideal in theories than in practice. Others are ontology generation from competency questions as well as ontology reuses. Based on the general guidelines advised in the work of Noy and Mcguinnes [23], the METHONTOLOGY [159], DILIGENT [160], and the Onto-Knowledge methodology, a comprehensive ontology modeling strategy is summarized here. The UC-ONTO development methodology takes into effect, the practicalities involved in domain knowledge elicitation as well as the current technologies available for ontology development. The process of the UC-ONTO development, which can also be replicated for any domain ontology engineering, can be summarized as follows:

(i) Ontology requirements specification.

(ii) Domain knowledge gathering and conceptualization.

(iii) Modular implementation of Ontology fragments.

(iv) Standardization of Ontology fragments.

(v) Versioning and Assembly.

However, a more detailed description of the methodology highlighting the numerous activities for each stage is given below:

(i) Ontology requirements specification.

   a. Identify Users, Scope and Purpose of the Ontology.

   b. Decide on the Nature of Ontology Development — heavily reuse or from Scratch.

   c. Ontology Language Selection and relevant technologies. Note: For a taxonomy an OWL language is ideal, while for Data manipulation vocabulary, an RDF Schema is sufficient.

   d. Development Tool selection — For an open-source project and experienced Ontologists, the Protege ODE is ideal. While for non-ontologist, a licensed ODE such as TopBraid Composer may be considered.

(ii) Domain knowledge gathering and conceptualization.

   a. Conceptualization from Data sources.

   b. Conceptualization from Competency questions.

   c. Reuse: importing other Ontologies.

   Note: 'Conceptualization' is any simplified version of knowledge that we wish to represent for some purpose.

(iii) Modular Implementation of Ontology fragments.

   a. Generate Sub-ontologies (for Applications, non-domain concepts) as fragments of larger Domain Ontology.

(iv) Standardization of Ontology fragments.

   a. Alignment and Merging of Ontology fragments.

  b. Concepts alignment with foundational ontologies.
  c. Annotations: natural language definition of domain concepts.

(v) Versioning and Assembly.
  a. Create link-points for Ontology fragments.
  b. Import (or manually assemble) all Modular fragments into Larger Ontology.
  c. Create a version tracking for each Ontology development round through Annotations and Datatype creators.
  d. Create/Update ontology header to include versioning info.

(vi) Ontology Extension with Logic Programming Rules (Optional).

These ontology development guidelines help to structure the ontology engineering process by identifying important but non-obvious aspects, such as the target users of the ontology, supporting tools, and specifying what values can be allowed for properties. Other aspects that are apparent and also common to all methodologies — such as defining domain terms and roles, asserting their hierarchy, and filling the concept slots with individual instances — performed iteratively for each source of data to populate the underutilized crops ontology. Similarly, our user-defined SWRL rules were added iteratively while ensuring the consistency of the ontology by invoking the Reasoner.

3.2.1 Generating Ontologies from Competency Questions

As the major purpose of a knowledge base is likely to answer user queries for ease of decision making, queries can be designed to identify such basic questions that an ontology model must answer. These questions are commonly referred to as Competency Questions (CQs) and since domain-specific ontology development process involves both the knowledge engineers as well as domain experts, the latter can then provide the required responses that can serve as knowledge input to the ontology model. The resulting set of CQs and their answers can thus serve as the starting point from which typical domain concepts can be generated and their relationships and instances outlined.

The basic steps involved in generating ontologies from competency questions can be outlined thus:

(i) Identify Competency Questions — the possible queries that the tool will answer based on its Users' (Researchers, Farmers, general public, Agronomists, etc.) perception.

(ii) Eliciting the answers to competency questions with the help of domain experts.

(iii) List out the terms or concepts identified from the competency questions and answers.

   a. Classify the terms.
   b. Identify the relationship between them.
   c. Assert repetitive facts (Individual instances) as class members.

In practice moreover, CQs can be used to support the entire ontology development life-cycle, with new concepts added to the ontology as the question-and-answer base grows. As such, CQs provide a good starting point for generating ontologies by extracting specific domain concepts and their relationships from the questions and answers — usually from domain experts and data stores. As noted in [23], the CQs being representatives of knowledgebase requirements, serve as pointers to the overall scope of ontologies. Moreover, the lists of CQs can provide a good starting point for generating ontologies by extracting specific domain concepts found in both the questions and their possible answers (usually from the domain experts).

As they are in natural language and less formal when compared to the software engineering requirement specifications, CQs helps non-logicians to approach ontology development, understand its scope and subsequently evaluate the final domain ontology. Furthermore, CQs can be posed as queries during ontology evaluation, to ascertain whether the knowledgebase achieves its intended purpose. A framework for evaluating ontologies against their corresponding CQs was proposed in [161].

3.2.1.1 List of Competency Questions for UC-ONTO Development

Though the list of competency questions can never be exhaustive, we present some examples of the CQs used to elicit the underutilized crops domain knowledge. The questions focuses on the main underutilized crop Bambara Groundnut as a reflection of the data availability from the domain experts.

- What are the commonly cited Underutilized Crops?
- Which UCs characteristics should a Farmer consider when planting a particular UC?
- Is Bambara groundnut a Seasonal or an Annual plant?
- What are the Soil types/Temperature/Rainfall requirements for an optimum yield of an UC?
- What are the best conditions for Bambara groundnut farming?
- What other Regions of Cultivation match the farming centers of Bambara Groundnut?
- Which Country has the highest yield of Bambara Groundnut in the year 2016?
- What are the good Landraces of Bambara groundnut for Nigeria?
- Does Bambara groundnut nutrients make up a complete food?
- What are the nutrition components of a particular Crop?
- Which part of Bambara Groundnut stores the food?
- How many days does it take for Bambara groundnut to flower/produce food/harvest?

- What happens to the yield of Bambara groundnut if the annual or Seasonal rainfall is moderate?
- What type of root does Bambara groundnut possess to survive the draught?
- What are the unique features of Bambara groundnut as compared to other Legumes?
- What are the commercial products of Bambara groundnut?
- Does Bambara groundnut grows better in a Mixed farming or Crop Rotation?
- What are the best candidates for mixed farming with Bambara groundnut?
- What are the common Pests and Diseases of Bambara groundnut?
- What other names is Bambara groundnut called around the World?
- Which farm input or fertilizer is often used for Bambara groundnut?

It should be noted here that our competency queries and responses for the underutilized crops ontology (see Section 4.2.6), which were elicited in the form of ”if (question) then (answer)” statements, sets the stage for designing the SWRL rules used to extend the UC-ONTO. That is, where the conceptual relationship cannot be asserted as a hierarchy, we simply adopt the rule assertion format of the CQs. The approaches of extending ontologies with logic programming rules were previously discussed in Section 2.8 and the example SWRL rules added to the UC-ONTO were presented in Section 4.2.7.

In essence, the ontology engineering methodologies were put into practice in the next Chapter, which discussed in Section 4.2, the development process of the Underutilized Crops Ontology (UC-ONTO). In the next sub-section however, we discussed the ontology modeling using the Protégé Ontology Development Environment (ODE).

3.3 Ontology Development with Protégé ODE

3.3.1 Naming Conventions

Best practices for naming conventions were adhered to during the ontology development with class names beginning with a capital letter and no spaces e.g. *Underutilized Crops, BambaraGroundnut etc.* and properties or relations beginning with a small letters, also with no spaces e.g. *hasLocalName, hasFeatures*. Whereas, Individual instances and concept names that are not class or property names, begin with a capital letter but use a hyphen or underscores (Protégé's default) in some cases to make it readable e.g. *Leaf-spot*.

The complete ontology with classes, properties, facts, rules and individuals instances rendered in OWL is available for download (as .owl file) in the below link[1].

3.3.2 Entities Assertion — Classes and Hierarchy Definition

The basic ontology components are Classes and OWL has a predefined Class ancestor called the 'Top' concept class or 'Thing' in which all concepts (Classes, properties and individuals) in an OWL ontology belongs as sub-entities. Domain concepts or Classes are added as successors of the 'owl:Thing' super class.

Defining a class in Protégé is achieved either using the 'Entities' or 'Classes' tab as shown in Figure 3.1. The Entities tab allows a generic overview of a Concept — Classes (top-left) or Properties (bottom-left) with their most relevant concepts such as the 'Annotations and Usage' features in the top-right corner and 'Description' of the concept in the bottom-right. An 'Entity' in Protégé ODE is analogous to a 'Concept' in Ontologists terms — as they involves all meaningful terms in an ontology such as Class, Property (Object and Datatypes), Individual instances and Annotations.

For compactness, all Crops related terminologies in the Underutilized Crops ontology will be defined under the 'DomainConcepts' Class as an ancestral class, followed by the 'UnderutilizedCrops' class to contain only those terminologies of crops that are categorized as underutilized. Due to the research focus of CfFF as

[1]https://drive.google.com/file/d/0B1FHDywt7JQlWWNhMzRjWmRuNVk/view

the data-source and the domain data availability, a focus class for the Ontology is the ’BambaraGroundnut’ class added as a direct subclass of the ’Underutilized-Crops’.

Classification and Hierarchy definition of concepts can be achieved through the Class hierarchy tab or through ’Restrictions’ added as ”Domain’ and ’Ranges’. Other method of achieving hierarchical representation is through ’Rules’ added as constraints and which will be inferred once a Reasoner is invoked to check the consistency of an ontology — See section 5.2.5. The class hierarchy tab allows adding classes either as subclass of another or as sibling classes as shown by the plus signs directly under the class hierarchy tab in Figure 3.1.

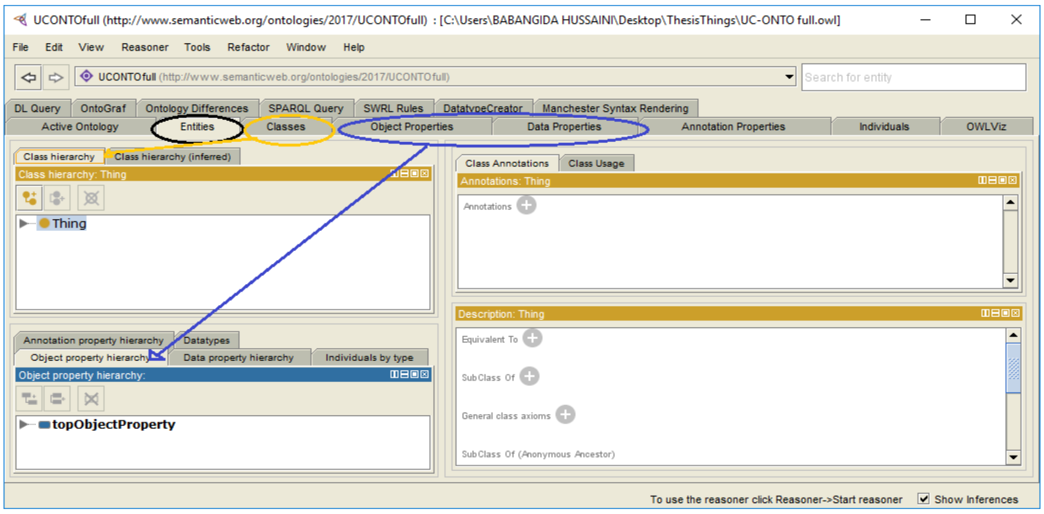

UCONTOfull (http://www.semanticweb.org/ontologies/2017/UCONTOfull) : [C:\Users\BABANGIDA HUSSAINI\Desktop\ThesisThings\UC-ONTO full.owl]
File Edit View Reasoner Tools Refactor Window Help
UCONTOfull (http://www.semanticweb.org/ontologies/2017/UCONTOfull)
Search for entity
DL Query OntoGraf Ontology Differences SPARQL Query SWRL Rules DatatypeCreator Manchester Syntax Rendering
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz
Class hierarchy Class hierarchy (inferred)
Class hierarchy: Thing
Thing
Annotation property hierarchy Datatypes
Object property hierarchy Data property hierarchy Individuals by type
Object property hierarchy:
topObjectProperty
Class Annotations Class Usage
Annotations: Thing
Annotations
Description: Thing
Equivalent To
SubClass Of
General class axioms
SubClass Of (Anonymous Ancestor)
To use the reasoner click Reasoner->Start reasoner
Show Inferences

Figure 3.1: Entities Assertions tab showing Class Definition

3.3.3 Entity Assertions — Object and Datatype Properties

Properties, roles or relationships in OWL ontology is of two types, the ’Object Property’, which describes the relationship between two Classes and the ’Datatype Property’, which describes a relationship between Individual instances and their respective primitive data values such as strings, integer, etc. As shown in Fig. 3.2, the Entities tab in Protégé or the respective Property tabs can be used to assert object or datatype properties. Another important modeling parameter for managing relations between objects in OWL ontologies is the ’Property Characteristics’ tab (bottom-middle), which allows asserting the nature or restriction of the property type.

Based on the common logical conclusions of relationships between concepts, the following Property characteristics are allowed in the Protégé Property assertions tab:

1. Functional vs Inverse Functional property — where the value of the relationship is a one-to-one (unique) between concepts or where the role is a one-to-many relationship.

2. Transitive property — where the relationship between concepts involves a sandwich or transferable roles. It is usually enforced by the reasoner to generate transitivity where they exist between concepts.

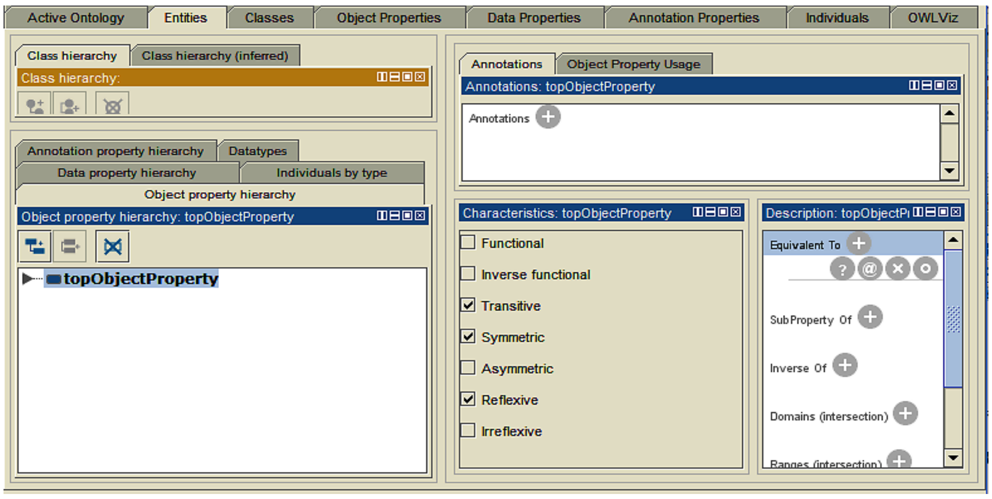
3. Symmetricity vs Asymmetric property — where relations are mirror images between concepts (e.g. spouse) or the reverse is the case, the antisymmetric characteristic is imposed by the reasoner.

4. Reflexive vs Irreflexive property — where the role or relationship is between concepts that are interchangeable or subclass of themselves (e.g. Person is-a Person). Irreflexivity is asserted where concepts or individual can not be interchanged.

Example: The property *hasLocalName* which describes the typical names of a crop (e.g. Bambara Groundnut) in its locality, can be asserted as 'functional' because the receiving class *LocalNames* (Property Range) can contain only those names for *BambaraGroundnut* (the Domain) class. Therefore any member of the class should be a unique local name for Bambara Groundnut as related by the inverse property *isLocalNameOf*.

3.3.3.1 Property Domain and Ranges

The OWL property Domain and Range as described above have different interpretation with the mathematical domain and range. Instead of restricting individuals into a given domain and range or used as data-type checks as traditionally known, in OWL modeling they are simply used as axioms during reasoning process to



- [ ] Functional
- [ ] Inverse functional
- [x] Transitive
- [x] Symmetric
- [ ] Asymmetric
- [x] Reflexive
- [ ] Irreflexive

Figure 3.2: Entities Assertion tab showing the Top Object Property and its Characteristics

notify the Reasoner that only those classes that belongs to the domain (originating class) and range (the recipient class) of the property can be inferred. As an example, consider the use of domain and range on the *hasLocalName* property above. Now, should we add the malay local name of say 'Orange' as another fact into our ontology, i.e. "Orange *hasLocalName* Oren". Then without defining a domain and range of the *hasLocalName* property, it will be inferred at reasoning time that Oren is also a local name of BambaraGroundnut class because it is will be found in the LocalName class. However, by adding that the domain and range of a property, such inconsistent knowledge modeling are easily avoided.

3.3.4 Entity Assertion — Individual Instances

Individuals or Class members list can be asserted in the 'Individuals tab' or from the 'Class tab' added directly as members, by clicking the plus sign beside the 'Members' sub-tab in the 'Descriptions tab — see Fig. 3.1. In the Individual tab however, untyped instances can be added and their Property assertions such as Object and Datatype properties or their negations. Equivalent instances that shares similar class types, e.g. Human and Person can be described using the *Same Individual As* sub tab. While disjoint individuals can be described using the *Different Individuals* sub-tab.

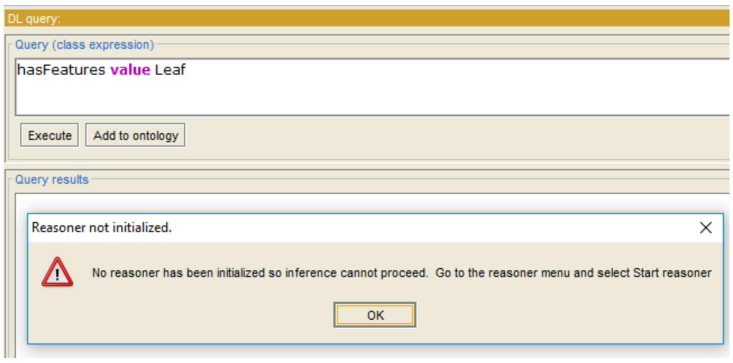
3.3.5 Reasoning and Queries

3.3.5.1 Description Logic (DL) Queries

DL Query is a simple and powerful ontology searching mechanism used to search for class expressions in a consistent ontology. The Ontology needs to be classified (by invoking a Reasoner) before running a DL Query over it. DL queries are written using an easy to use syntax called ’Manchester OWL Syntax’ which is a user-friendly, human readable syntax that allows writing class or individual expressions as queries and the results will be displayed in the ’Query Result’ tab. Unlike the XML/RDF and OWL/XML syntaxes for OWL 2, it is a frame-based format where complete information about a Class, Individual, or property is collected in a single assembly called frame.

An example DL query in Manchester syntax to retrieve ”a class with at least one feature” and :another class with exactly a ’leaf’ as its feature”, from our Crop Ontology in DL will be thus:

- hasFeatures some Features
- hasFeatures value ’Leaf’



```
hasFeatures value Leaf
```

Reasoner not initialized.

No reasoner has been initialized so inference cannot proceed. Go to the reasoner menu and select Start reasoner

Figure 3.3: DL Query tab showing Reasoner not Initialized error

As shown in the figure 3.3, a reasoner need to be initialized and inference achieved before a DL query can be written. This is to ensure that the ontology is consistent before answering, which helps to avoid incomplete information retrieval. See section 3.3.5.3 for details on ontology inference and consistency checking.

3.3.5.2 SPARQL Queries — for Large and Federated Queries

As earlier explained, the building block of Semantic Web data is the Resource Description Framework (RDF) triple graph, and SPARQL is the query language for manipulating the RDF graph data. SPARQL is a recursive acronym which stands for: 'SPARQL Protocol and RDF Query Language'. SPARQL queries can contain a combination of RDF triples with disjunctions and/or conjunctions of classes and properties. The subject, object, and predicate triples in SPARQL queries contain variables which will be used to match existing graph patterns in the Ontology knowledge base.

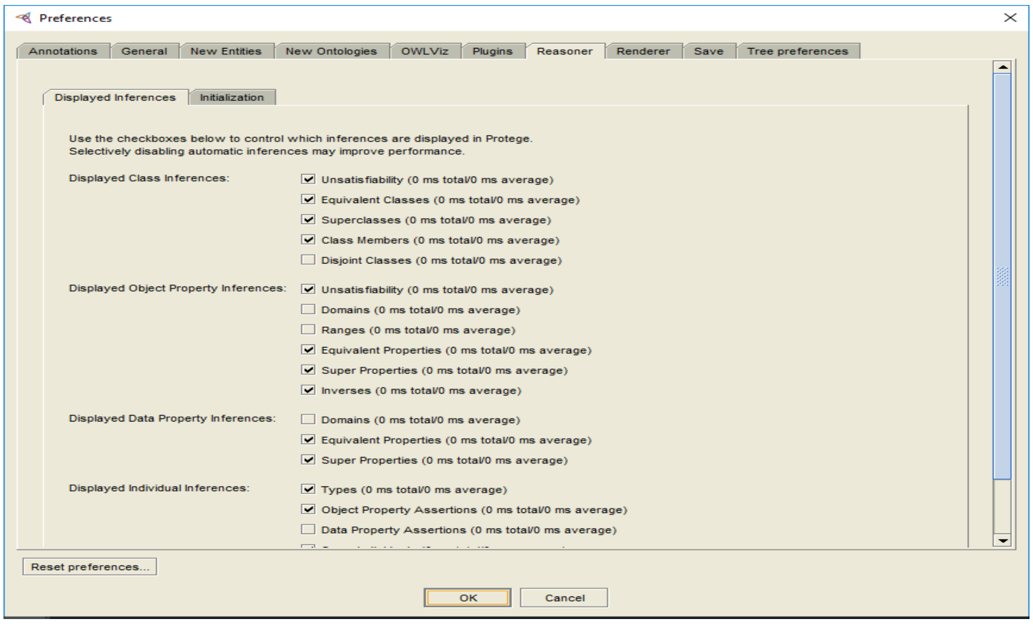
SPARQL Query Format: In its basic form, the SPARQL contains a 'SELECT' and 'WHERE' clauses, just as in SQL Queries, see the Listing below:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subject ?object

        WHERE {?subject rdfs:subClassOf ?object }

ORDER BY
GROUP BY ...
```

Note that the PREFIXes are imports for the RDF syntax namespaces (rdf-syntax-ns), OWL syntax (owl), XML Schema (XMLSchema) and RDF Schema (rdf-schema) respectively. They are standard prefixes needed to access, retrieve and presents the ontology data correctly. Other prefixes are added below these standard prefixes to show the path to any participating ontology or data source for the SELECT statement. Hence, various data sources can be searched at once through the federated SPARQL query feature. We highlight the resulting case study of the SPARQL searching in our Onto-CropBase tool in Chapter 4.

3.3.5.3 Consistency Checking

As earlier discussed, developing ontologies requires a Reasoner to be invoked to ensure consistency and infer additional knowledge from asserted axioms. Commonly used reasoners available in the Protégé ODE includes the Pellet, HermiT, Fact++, and Pellet Incremental [14]. To check for the consistency of ontologies, the Reasoner tab provided in the Protégé menu provides the list of default reasoners and their reasoning strategy. For more explicit reasoning, the 'Reasoner Configuration' sub tab is selected to configure what pre-computation tasks needs to be done. For a simple ontology development, it is recommended that the default setting be allowed, were the Reasoner plug-ins decides the pre-computation task such as 'Initialization, Classification, and Realization'.



Use the checkboxes below to control which inferences are displayed in Protege.
Selectively disabling automatic inferences may improve performance.

Displayed Class Inferences:
- ✓ Unsatisfiability (0 ms total/0 ms average)
- ✓ Equivalent Classes (0 ms total/0 ms average)
- ✓ Superclasses (0 ms total/0 ms average)
- ✓ Class Members (0 ms total/0 ms average)
- ☐ Disjoint Classes (0 ms total/0 ms average)

Displayed Object Property Inferences:
- ✓ Unsatisfiability (0 ms total/0 ms average)
- ☐ Domains (0 ms total/0 ms average)
- ☐ Ranges (0 ms total/0 ms average)
- ✓ Equivalent Properties (0 ms total/0 ms average)
- ✓ Super Properties (0 ms total/0 ms average)
- ✓ Inverses (0 ms total/0 ms average)

Displayed Data Property Inferences:
- ☐ Domains (0 ms total/0 ms average)
- ✓ Equivalent Properties (0 ms total/0 ms average)
- ✓ Super Properties (0 ms total/0 ms average)

Displayed Individual Inferences:
- ✓ Types (0 ms total/0 ms average)
- ✓ Object Property Assertions (0 ms total/0 ms average)
- ☐ Data Property Assertions (0 ms total/0 ms average)

Figure 3.4: Reasoner Inferences Selection in Protégé ODE

In the preferences display, users can chose from check-boxes (See Fig. 3.4) which inferences will be realized and displayed in the ODE. Selectively un-checking some of the boxes to disable the automatic inference of unnecessary inferences, such as disjoint classes, super-properties, etc, will go a long way in improving the Reasoner performance and speed. Moreover, as shown in Fig. 3.3, an active reasoner is also needed to allow writing SWRL rules and for DL query execution. As such, it is strongly advisable that the reasoner is always active during ontology development to avoid introducing intractable inconsistencies.

3.3.5.4 Adding Query Results to Ontologies

Due to the complex and eliciting nature of queries, an ’Add to Ontology’ option is provided during DL query (See Fig. 3.3) writing to enable the assertion of the exact query results into the ontology — as a new class. This helps to avoid repeated query writing and allow for efficient knowledge extraction from ontologies.

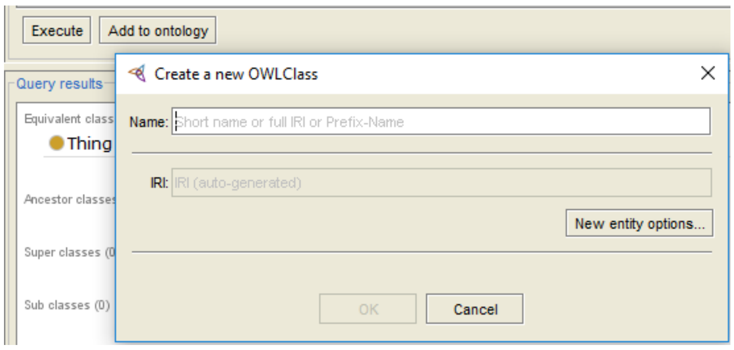


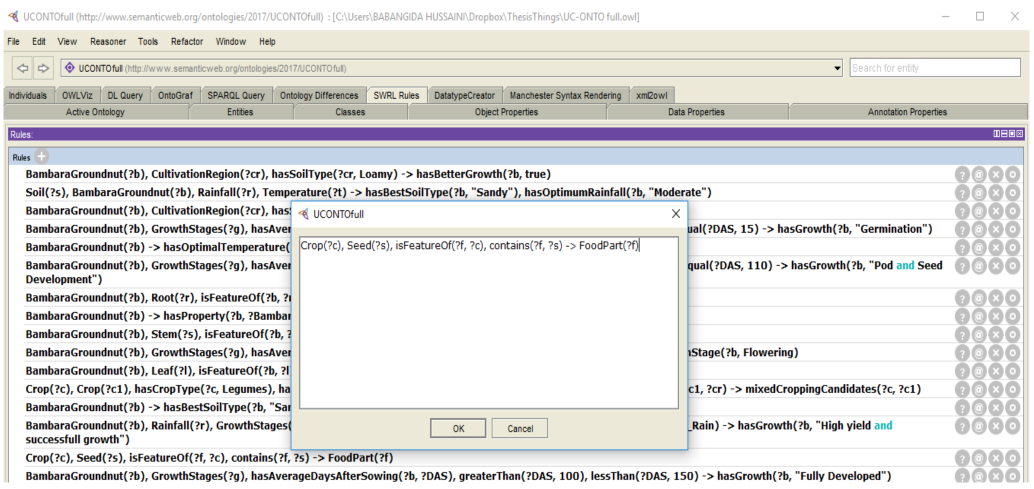
Figure 3.5: Reflexive Ontology Generation - Adding Query Results in Protégé ODE

This reflexive ontology generation approach, though rarely mentioned, is highly efficient in enhancing the knowledge elicitation process in domain ontologies. However, as shown in Fig. 3.5, a new class need to be created which can lead to further alignment of the main ontology. For easier utilization of this feature, a separate class ”QueryResults” can be created as a sibling class to the ’Domain-Concept’ class to collate all new queries. This will help to avoid inconsistencies and ease of management of the main ontology.

3.3.6 Rules Assertion: Adding User-defined SWRL Rules in OWL Ontologies

Writing SWRL rules in the Protégé ODE requires creating the Rules tab as the default installation does not provide the rules assertion option. However, the ”Rule” View is already available in the Window menu $->$ Views $->$ Ontology Views $->$ Rules.

As shown in Figures 3.6 and 4.9, A plus sign is provided in the rules tab to allow adding user defined rules in SWRL. Rules are added as OWL axioms by us-



```
UCONTOfull (http://www.semanticweb.org/ontologies/2017/UCONTOfull) : [C:\Users\BABANGIDA HUSSAINI\Dropbox\ThesisThings\UC-ONTO full.owl]

Rules:
BambaraGroundnut(?b), CultivationRegion(?cr), hasSoilType(?cr, Loamy) -> hasBetterGrowth(?b, true)
Soil(?s), BambaraGroundnut(?b), Rainfall(?r), Temperature(?t) -> hasBestSoilType(?b, "Sandy"), hasOptimumRainfall(?b, "Moderate")
BambaraGroundnut(?b), CultivationRegion(?cr), has
BambaraGroundnut(?b), GrowthStages(?g), hasAver          ual(?DAS, 15) -> hasGrowth(?b, "Germination")
BambaraGroundnut(?b) -> hasOptimalTemperature(
BambaraGroundnut(?b), GrowthStages(?g), hasAver          qual(?DAS, 110) -> hasGrowth(?b, "Pod and Seed
Development")
BambaraGroundnut(?b), Root(?r), isFeatureOf(?b, ?r
BambaraGroundnut(?b) -> hasProperty(?b, ?Bambar
BambaraGroundnut(?b), Stem(?s), isFeatureOf(?b, ?
BambaraGroundnut(?b), GrowthStages(?g), hasAver          Stage(?b, Flowering)
BambaraGroundnut(?b), Leaf(?l), isFeatureOf(?b, ?l
Crop(?c), Crop(?c1), hasCropType(?c, Legumes), ha          c1, ?cr) -> mixedCroppingCandidates(?c, ?c1)
BambaraGroundnut(?b) -> hasBestSoilType(?b, "Sa
BambaraGroundnut(?b), Rainfall(?r), GrowthStages(          _Rain) -> hasGrowth(?b, "High yield and
successfull growth")
Crop(?c), Seed(?s), isFeatureOf(?f, ?c), contains(?f, ?s) -> FoodPart(?f)
BambaraGroundnut(?b), GrowthStages(?g), hasAverageDaysAfterSowing(?b, ?DAS), greaterThan(?DAS, 100), lessThan(?DAS, 150) -> hasGrowth(?b, "Fully Developed")

UCONTOfull
Crop(?c), Seed(?s), isFeatureOf(?f, ?c), contains(?f, ?s) -> FoodPart(?f)
```

Figure 3.6: SWRL Rules Assertion tab in Protégé showing SWRL Rule format

ing known classes, object properties, data types and individual names. These were added as conjunctions, separated by a comma, to represent an if-then declarative assertion of domain facts. The antecedent is thus separated from the consequence using the dash and the greater than signs ($- >$) to show the direction of the inference. Rules added can be 'edited, deleted, explained by asserting annotations and their inferences can be questioned' using the 'O, X, @ and ?' symbols respectively as seen in the right-hand side Fig. 3.6. While any class names can be used in writing SWRL rules, known classes are encouraged in accordance to the DL-Safety restriction (See Section 2.8.2.1) to avoid inconsistencies and decidability of the main ontology.

3.3.7 Ontology Reuses: Ontology Imports in Protégé Ontology Editor

Importing an ontology or set of ontologies into an open ontology file in Protégé is achieved through the 'Direct Imports' sub-menu in the Active ontology tab or using the File menu tab. Clicking the plus sign of the Imports menu starts the Ontology import wizard which helps to semi-automate the ontology import process. Following options are available for importing ontologies: (i) from a local file or from ontologies located somewhere on the web. For the latter, the URL that points to the file that contains the ontology needs to be specified, i.e.

the physical URL rather than the ontology URI. (ii) Other import options include importing ontologies from the ontology libraries, which goes through all available ontology repositories (in the local host) and presents their IRIs for the user to chose from. (iii) Lastly, loaded ontologies already in the Protégé workspace can also be imported into a particular ontology — called Ontology gathering. Indirect imports of ontologies can also be achieved by directly copying ontology concepts of one ontology into another.

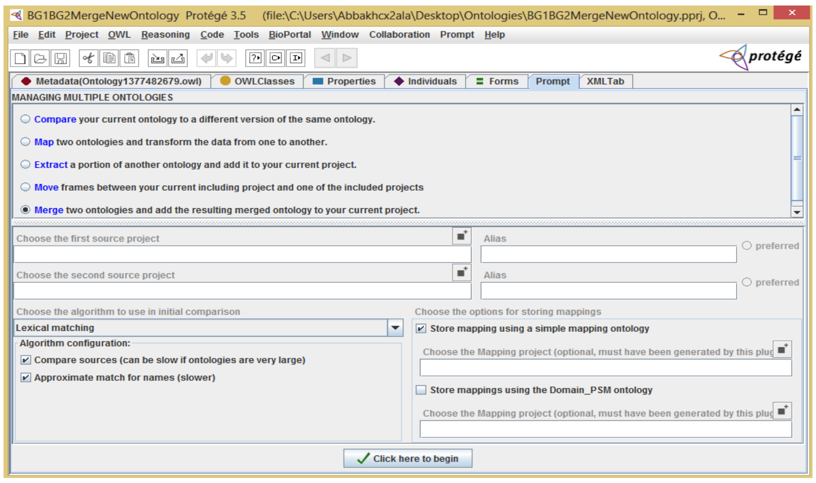
Once an ontology is imported, the next task involves creating conceptual links between the different ontology entities. Where the imported ontology is needed without modification or alignment, the required classes, properties or data values needed are then asserted and the Reasoner invoked to check for consistency. However, as most ontologies are only needed inorder to reuse some of their concepts in another existing ontology, there is a need to sometimes 'merge' ontologies after importing them. We discuss the ontology standardization approaches in Section 2.5.4.

3.3.8 Ontology Merging and Alignment

3.3.8.1 Ontology Merging with PROMPT

Built initially as a plug-in to Protg 2000 Knowledge Base Tool, PROMPT [162] is a comprehensive Ontology Mapping and Merging algorithm with the options to 'Compare' ontology versions, 'Map' two ontologies, 'Extract' or import a part of one ontology to another and 'Merge' together two ontologies. It also has a 'Move frames' option as an added support for Frame-based ontologies (PROMPT is a feature of Protg 3.5 which have support for OWL using Protg-OWL knowledge model). Moreover, PROMPT provides useful suggestions and feedback to the user during the process and where a conflict exists, PROMPT will direct the user to the affected Classes. Another useful feature of PROMPT is that it allows preference during merging such that one of the two ontologies can be selected as preferred and the merging process will be favored towards the preferred ontology and PROMPT will automatically resolve conflicts (naming, restrictions, etc) in its favor.

On the more recent versions of the Protégé ontology editor however, merging


BG1BG2MergeNewOntology Protégé 3.5 (file:\C:\Users\Abbakhcx2ala\Desktop\Ontologies\BG1BG2MergeNewOntology.pprj, O...

File Edit Project OWL Reasoning Code Tools BioPortal Window Collaboration Prompt Help
protégé
Metadata(Ontology1377482679.owl) OWLClasses Properties Individuals Forms Prompt XMLTab

MANAGING MULTIPLE ONTOLOGIES

- ○ Compare your current ontology to a different version of the same ontology.
- ○ Map two ontologies and transform the data from one to another.
- ○ Extract a portion of another ontology and add it to your current project.
- ○ Move frames between your current including project and one of the included projects
- ● Merge two ontologies and add the resulting merged ontology to your current project.

Choose the first source project — Alias — ○ preferred

Choose the second source project — Alias — ○ preferred

Choose the algorithm to use in initial comparison: Lexical matching

Algorithm configuration:
- ☑ Compare sources (can be slow if ontologies are very large)
- ☑ Approximate match for names (slower)

Choose the options for storing mappings
- ☑ Store mapping using a simple mapping ontology
  - Choose the Mapping project (optional, must have been generated by this plug
- ☐ Store mappings using the Domain_PSM ontology
  - Choose the Mapping project (optional, must have been generated by this plug

Figure 3.7: PROMPT Interface for Managing Multiple Ontologies in Protégé 3.5

ontologies is achieved through the 'Refactor' tab — see Fig. 3.8. This allows similar process of merging ontologies as described with the PROMPT plug-in, with the options to merge two or more ontologies to create a new ontology or simply merge into an existing ontology. At the final stage of the ontology merging process, a different ontology rendering format can be selected.

It should be noted however, ontology merging is usually semi-automated as user intervention is necessary from start to finish — especially where there is a conflict in naming convention of concepts. The user then has to choose which concept name to keep or discard and from which ontology. The process is more tedious for merging very large ontologies and it becomes overly tedious if the ontologies have many comparable class names.

We present our case study of using the merging tool including the challenges involved in Section 4.2.6.

3.3.9 Annotations and Natural Language Definition of Concepts

Due to the technical jargons and specialized keywords associated to their terminologies, many domain-specific ontologies are not easily comprehensible by others (non-experts in the domain), leading to low reusability and conformism. However, a natural language definition of domain-specific concepts or terminolo-

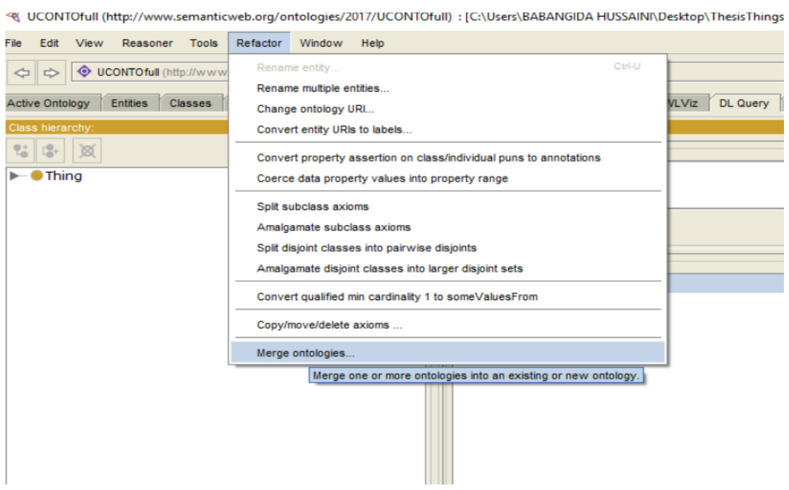

UCONTOfull (http://www.semanticweb.org/ontologies/2017/UCONTOfull) : [C:\Users\BABANGIDA HUSSAINI\Desktop\ThesisThings
File
Edit
View
Reasoner
Tools
Refactor
Window
Help
UCONTOfull (http://www
Active Ontology
Entities
Classes
VLViz
DL Query
Class hierarchy:
Thing
Rename entity... Ctrl-U
Rename multiple entities...
Change ontology URI...
Convert entity URIs to labels...
Convert property assertion on class/individual puns to annotations
Coerce data property values into property range
Split subclass axioms
Amalgamate subclass axioms
Split disjoint classes into pairwise disjoints
Amalgamate disjoint classes into larger disjoint sets
Convert qualified min cardinality 1 to someValuesFrom
Copy/move/delete axioms ...
Merge ontologies...
Merge one or more ontologies into an existing or new ontology.

Figure 3.8: Ontology Merging Process in Protégé ODE

gies can help in alleviating most of the technical ambiguities and increase the usability and interoperability of such ontologies. Simply put, the lack of natural language definitions of concepts is a serious hindrance to understanding much domain-specific ontologies, thousands of which are available online. Two main approaches were suggested in [163] for improving domain ontologies using the natural language definition of concepts, namely — the use of Annotations (such as labels, comments etc) and standardized naming patterns.

Creating New Annotations: User-defined Annotations In the Annotation design pattern, use of *rdfs:label* and *rdfs:comment* constructs — for annotating domain concepts with widely known keywords and providing comments on ambiguous terms, is highly advised in ontologies. Whereas, the Naming design pattern approach, advises on the use of standard naming conventions for domain terminologies. Such as, the use of capital letter to begin names of classes and small case letter for properties, among other naming conventions as advised and agreed upon by professionals in the domain of discourse.

In essence, providing natural language definition of ontological concepts greatly improves its acceptance and usability among humans and machines alike. We highlight with examples on using natural language annotations for standardizing our UC-ONTO in Section 4.2.6.

3.4 The Onto-CropBase Development Methodology

Utilizing the rapidly growing collection of semantic data is one of the contemporary challenges in the field of life sciences. In the last decade, with the advancement of semantic technologies in design and development of domain-specific decision support systems, various RDF datasets, SPARQL endpoints, OWL ontologies as well as generic taxonomies have been developed. Ontologies being explicit specifications of domain knowledge, as widely quoted in [12], are known to enhance the semantics of data so that it can be easily interpreted by machines and often enable consistent, precise and human-understandable queries over such data. While these developments strengthen semantic knowledge modeling and representation, of equal importance is the retrieval and dissemination of the knowledge to non-technical stakeholders for informed decision-making. As such, user-friendly tools with communal access and customized presentation views needs to be considered beyond the usability offered by desktop applications. To this end, a web interface for exploring the UC-ONTO through concept-based semantic search engine was designed and developed. The tool is termed as the *Onto-CropBase* — which is a web-app developed to serve as an interactive gateway to the underutilized crops ontological knowledgebase, enhanced with additional linked-datasets in RDF[2].

In this section, the design strategy and methodology employed in the development of the Onto-CropBase tool were discussed and as explained earlier, the Onto-CropBase tool was designed to comprise three major components: (i) the *domain ontologies* component—consisting of a global ontology developed in OWL 2 and local ontologies in RDFS, (ii) the *mediator* component—provided by the Jena API, and (iii) the *ontology-based semantic search engine*—developed using J2EE[3]. In what follows, the ontology-based integration strategies employed in developing the Onto-CropBase tool were discussed.

Details of the design and development of the domain ontology has been presented in Sections 4.2 and 3.2 respectively. The methodology discussed includes the process of converting available crops data and metadata from XML to OWL

[2]Details of the Onto-CropBase design and development is published in our paper: *The Onto-CropBase — A Semantic Web Application for Querying Crops Linked-Data [11].*

[3]http://www.oracle.com/technetwork/java/javaee/overview/index.html

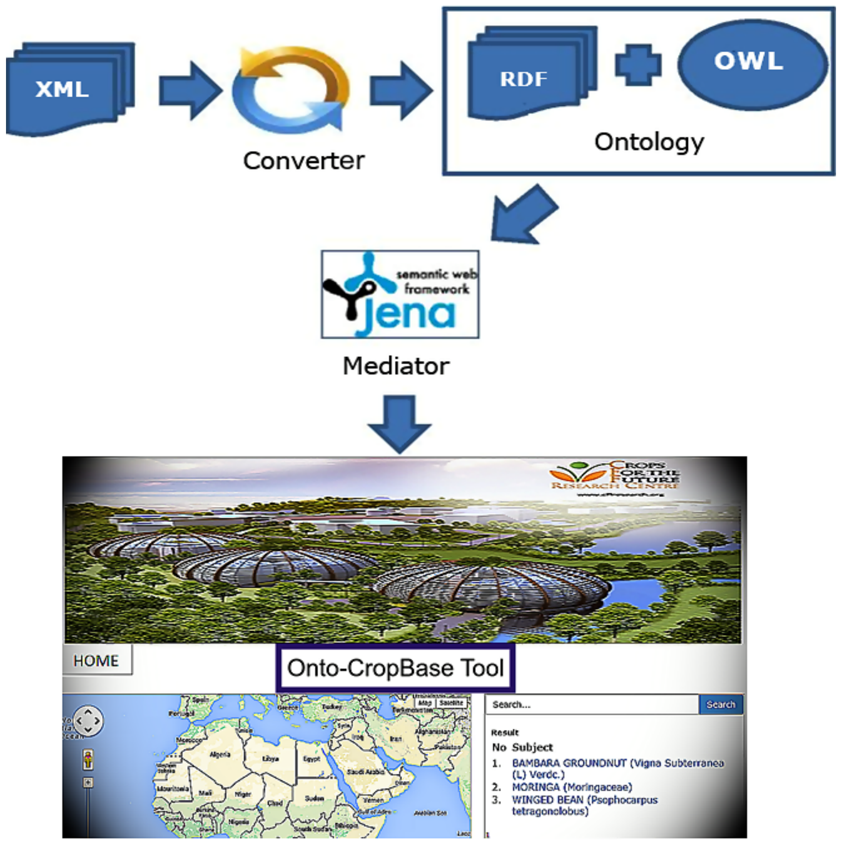


Figure 3.9: Onto-CropBase architecture showing the tool's components

format in order to easily generate relevant domain concepts. This is also presented with practical details in our paper [61]. Hence, the overall framework of the Onto-CropBase tool is based on an open-source solution as shown in Fig. 3.9. The methodology and detailed description of the Onto-CropBase components are presented in the following subsections.

3.4.1 Knowledge base Integration: Linking UC-ONTO with Relevant Data sources

To allow interoperability between the UC-ONTO and other crops data involved in the knowledgebase and due to the modularized nature of the ontologies involved, a connection needs to be made between the various components data sources to allow writing the federated queries. This is particularly useful when answering user queries that require pulling information from two or more data sources. Common procedure of integration involves the use of *mappings* between terms in the relevant ontologies, i.e., where two or more ontologies are involved, termed as, inter-ontology mapping. Another approach involves the *linking* of the ontologies with the actual information within the data sources, i.e., where data sources other than ontologies are involved in the integration process. By definition, *ontology mapping* is a directed matching of two or more ontologies in which entities of one ontology are mapped to at-most one entity of another ontology [164].

Ontology-based integration of data sources can be generally categorized into three approaches, namely, *the single ontology approach, multiple ontologies approach* and *the hybrid approach* [165]. The *single ontology* method follows the centralization of all data sources and shared vocabulary into a global ontology. This method is suitable when all data sources used for the integration share the same view of the domain. While in the *multiple ontologies* approach, each data source is separately described by its own local ontology with inter-ontology mapping between relevant terms. This method is suitable for decentralized data sources that do not need a common vocabulary. The third approach, *hybrid ontology integration*, involves a combination of the single ontology and multiple ontology approaches. In this approach, the ontology of each data source is developed separately and mapped to a shared vocabulary (global ontology) to allow interoperability.

For the Onto-CropBase development, the *hybrid* approach was employed to integrate our UC-ONTO with the RDF data sources or ontologies due to the particular advantage that new sources can be easily added without modifying existing mappings. This suits comfortably with our ever-evolving underutilized crops knowledgebase, the UC-ONTO, as it ensures that the knowledgebase can be easily extended in the future to accommodate data from other heterogeneous sources like RDFs, relational databases, excel and web documents. The architecture of the hybrid ontology approach is shown in Fig. 3.10 (left) and a textbook discussion on the three approaches with example use cases is presented in [165]. It should be noted that apart from achieving interoperability between data sources, ontology-based integration has the following added advantages: *global conceptualization, mapping support, metadata representation*, as well as *support for high-*

level queries, among others [166].

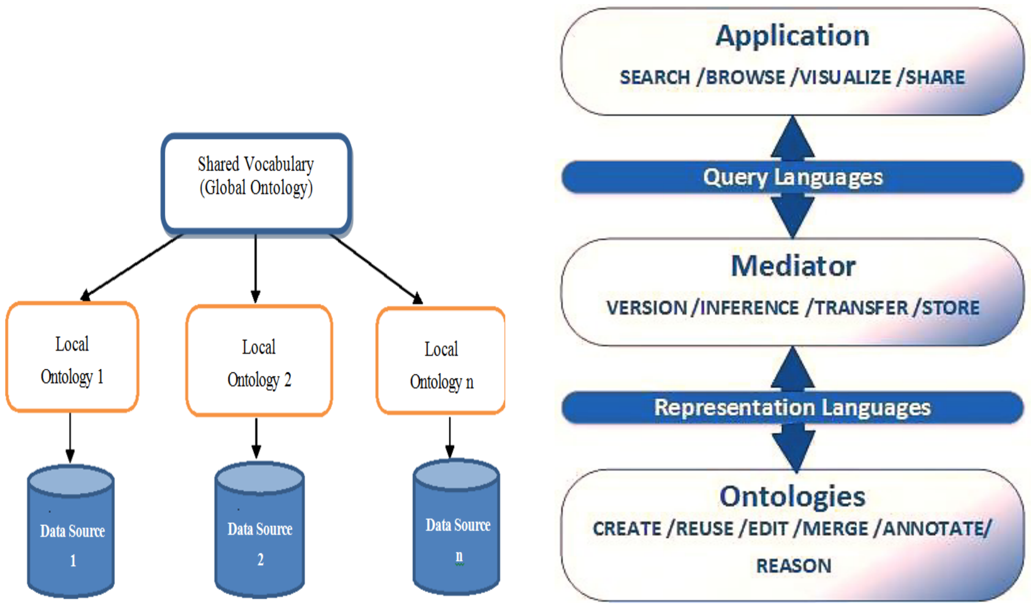


Figure 3.10: Hybrid approach to ontology integration (left) and an MVC adapted design concept for Ontology-based Applications (right).

3.4.2 Interface Design and Search Engine Approach

Main features in the Onto-CropBase interface include a search space, a map for location info and a results panel. These three main features are available in all sections of the tool. The home page of the Onto-CropBase tool is designed with a form-based search engine and as location data is critical to crop-based knowledge systems especially that of the underutilized crops, a map interface is provided to display the crops location data. The map interface is embedded using the Google Maps JavaScript API version 3, which enables map features in a web application, including styled maps, place data, 3D buildings, and geocoding, among other features. The map information is extracted from the named location's *east-west bound longitude* and *north-south bound latitude* elements. The following design approaches makes up for the Onto-CropBase features:

3.4.2.1 Search Approach.

In the Onto-CropBase search engine, a search begins with a keyword entered in the Search area and the query results are returned as a set of navigational links based on the subjects of the data sources—using their title annotation provided in the corresponding local ontology, see Fig. 4.12. Users can then browse the list of subject titles to explore the remaining information. Clicking on a particular subject will present the RDF assertions as subjects and objects pairs. For example, in Fig. 4.12, Subject number 3 is a caption for *Carbohydrate* of Bambara groundnut and the corresponding object asserted as its value is *High (65%)*.

3.4.2.2 Query Language.

SPARQL is used as the query language, which is a recursive acronym for SPARQL Protocol And RDF Query Language (SPARQL), and a standard query language for retrieving information stored in RDF graph or triples [167]. A basic structure of a SPARQL query includes the SELECT, CONSTRUCT, ASK, and DESCRIBE statements followed by the WHERE clauses and GROUPBY clauses where applicable, see example in Listing 3.2. As mentioned earlier, we employ the Jena-ARQ [168], a query engine for Jena that supports SPARQL RDF Query language. This allows for federated user queries across the local ontologies using the corresponding terminologies of concepts found in the global ontology as search phrases. Recall that the local ontologies are linked to the global ontology using their URIs which is also supplied into the SPARQL queries.

3.4.2.3 Query Design.

Queries entered in the Onto-CropBase search space are embedded in a Java code containing a SPARQL query and the concepts are parsed as ’text strings’ to the *QueryFactory* method. The SPARQL’s SELECT query is first executed on the global ontology (the UC-ONTO) to retrieve the relevant concepts matching the search keywords in the global ontology. An OPTIONAL query pattern, as shown in **Listing 3.2**, is provided to recursively explore all the RDF datasets linked to the UC-ONTO through their URIs. The use of the OPTIONAL construct allows the Jena-ARQ query engine to search for all relevant triples with inferences from the local RDF ontologies without a query execution failure — i.e. it acts as an exception condition in the event that the optional data does not exist. Moreover, the use of OPTIONAL query also helps to ensure that all non-optional facts are returned from at least, the global ontology. For example in the map query, in the event that one of the RDF subjects returned is a location data, a *setMap* object is then used to obtain the set of *longitudes* and *latitudes* of the location, for display on the map interface. The size of the resulting subject-predicate-object (SPO) triples, is also calculated for each dataset to determine the number of pages to be returned for each search result. In the case of Onto-CropBase, we set the capacity to 10 triple sets per page so as to allow the map view stay in focus.

```
1. "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"+
2. "PREFIX rdf: <http://.../1999/02/22-rdf-syntax-ns#> "+
3. "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> "+
4. "PREFIX ucnames: <http://.../ontologies/2015/Naming#>"+
5. "PREFIX agronomy: <http://.../ontologies/2015/agrono#>"+

6. SELECT distinct ?subject ?object " +
7.  WHERE { " + " " + "
8.   OPTIONAL " +  " {?subject rdf:value ?object ." + "
9.  ?subject rdf:type ucnames:"+ className +".} "+" "+"
10.  OPTIONAL " + " {?subject rdf:value ?object . " + "
11. ?subject rdf:type agronomy: "+ className + ".}"}
```

Listing 3.1: SPARQL query showing the use of SELECT and OPTIONAL constructors

The above listing shows example uses of the SELECT and OPTIONAL constructors in a SPARQL query to retrieve distinct subject-object pairs from two local ontologies: $ucnames$ and $agronomy$. Note that the prefix declarations, lines 1 through 6, allow for abbreviating URIs, so that the short names can be used instead of the URIs in the query body.

3.4.2.4 The Query Processing.

As mentioned earlier, Jena-ARQ is employed as a query engine for executing our SPARQL queries. This is achieved through a sequence of five iterative steps as follows:

1. String to Query parsing: where the text string parsed to the $QueryFactory$ method is structured from a query string to a Query object.

2. Algebra Generation: Next step is the algebra generation, which involves translation of the Query object to a SPARQL algebra expression using the SPARQL specification algorithm.

3. High-level Optimization: Third step is the optimization of the algebra expression generated in (2) and is called high-level optimization and transformation. Here, a Transformer class applies a transform code to convert or replace the algebra expression tree with more efficient expressions. Example of transform code function is in replacing the equality filter with more efficient graph pattern in algebraic expressions.

4. Low-level Optimization: In the next step, the final query plan is determined and is called the Low-level optimization. This involves deciding the order in which to evaluate the basic graph patters transformed earlier. However, this stage can be carried out concurrently with the fifth step,

5. Evaluation of the query plan: this involves executing the algebra expressions to generate the solution graph patterns, returned as sets of facts in SPO triples.

However, these steps can be extended and modified to allow searching for different graph-pattern implementations. Moreover, the final step (evaluation of the query plan), can also be enhanced to suit specific application requirements.

3.4.3 Selection of the Mediator Component

In compliance with the standards of Model-View-Controller (MVC) software architecture [169], the Onto-CropBase was also designed to have a robust control module handling the interaction between the thin web interface and the ontology-based semantic data model. Considering the design paradigm for ontology based applications in Fig. 3.10 (right), it is possible to make the following assertion: 'In order to develop a semantic web application (SWAP), a mediator component—*the controller*, is needed as the programmatic environment that can interact with the semantic data—*the model*, to allow visualization on the states of the model—*the views*'. Common open-source choices for a mediator component in developing

SWAPs include the OWL API [170], Protégé-OWL API [171], and the Apache Jena framework [172], among others. Detailed comparison of these common tools and applications can be found in [173].

3.4.3.1 The Jena API Distribution

The Apache Jena [172], is a java-based, free, as well as open-source programming tool for developing semantic web and linked-data applications. Firmly rooted in RDF, the framework consists of inter-connected APIs that can be invoked into an application code to manipulate an RDF data or knowledge model. Basic Jena APIs include Ontology, RDF, SPARQL, Inference and Storage APIs. The generic nature of the Jena's Ontology API, *OntModel*, makes it capable of handling other ontology languages that can be serialized in RDF format. A scalable Triple Store Database (TDB), which implements SPARQL specifications, is also provided in the Jena distribution for storing and retrieving RDF data. With regards to inference, a set of predefined Reasoners such as: the RDFS rule reasoner, Transitive reasoner, OWL-Lite Reasoners, and a Generic rule reasoner are available in the Jena distribution [168]. Moreover, Reasoners need not always be explicitly specified while working with the Jena Ontology API, as appropriate reasoner can be accessed to generate inferences by the query engine based on existing ontology configuration.

3.4.3.2 The OWL API

This is also an open source and java-based reference tool for creating as well as manipulation of OWL ontologies [174]. Centered on the web ontology language (OWL), it allows working directly with OWL axioms without having to be serialized in to RDF, as required by the Jena API. Moreover, the OWL API provides interfaces for OWL 2 with *parsers* and *writers* available for: RDF/XML, OWL/XML, OWL Functional Syntax, Turtle, KRSS and the OBO format. Various Reasoners such as Fact++, HermiT, Pellet, etc. are also embedded for inferencing [170].

3.4.3.3 The Protégé-OWL API

Considered as an extension to the OWL API, the Protégé-OWL API [171], is another open source java-based tool designed for manipulating both OWL and RDF models. The Protégé-OWL API also uses Jena for *parsing*, thereby providing some of the available Jena services. The API is provided in a standard Protégé installation. Notable improvements in the OWL API over its predecessors, is its ease of implementing graphical user interfaces (GUIs) for users based on their working ontology or data models and also the ability to control internal representation of the ontologies using the GUIs. Various DL-based Reasoners are also available in the Protégé-OWL API.

Consequently, due to its rich documentation, stability in handling RDF data and the availability of rule-based and non-rule-based Reasoners, we chose the 'Apache-Jena' to serve as the connecting point between our ontology-based model and the user interfaces or views. Moreover, our primary choice of a java-based development for the Onto-CropBase tool, is another deciding factor for choosing Apache Jena as the mediator. Nevertheless, there are other java-based tools that provide a complete development environment for SWAPs, by shielding the user from the need to bother with the above connections. Commonly known example include the Stardog RDF database [175] among others.

3.5 The Fuzzy Temporal Semantic Web Rule Language (FT-SWRL) Extension — Design Methodology

Despite the advancements of ontology languages, there still exists a wide research gap in achieving consistent representation formalisms for managing temporal uncertainties in domain ontologies. Modeling imprecise temporal expressions that depend on unstructured vague time data is still a challenge in the field of semantic web. Moreover, considering the diverse nature of information on the web (and by extension, the semantic web) — which to some considerable proportion involves experts as well as novice's narratives of events, the advancement of the semantic web no doubt requires an even more expressive modeling formalisms. The purpose of this research, therefore, is to bridge this gap by introducing a consistent fuzzy-temporal extension that can be used to represent and reason over uncertain-temporal domain knowledge in the semantic web. As described in [141], there are two types of temporal uncertainties — the imprecise dating of events and the fuzzy description of temporal data. The FT-SWRL aims to provide a new formalism for representing the latter in ontologies (i.e. modeling fuzzy temporal data) using SWRL rules. The new extension is defined as a new fragment of the existing SWRL Temporal formalism, where a fuzzy temporal ontology has been developed to extend the temporal ontology and new set of built-ins defined to represent the semantics of the imprecise temporal expressions for the reasoning purposes.

By defining the fuzzy temporal SWRL ontology (SWRL-FTO) as a reference model and designing relevant built-in operators, the FT-SWRL extension will surely improve the usability of the existing SWRL temporal formalism. While the basic temporal SWRL rules can represent interval operators such as those described as Allens temporal operators [157], utilizing such operators will be incomplete without some degree of fuzziness in the knowledge base. Moreover, fuzzification of imprecise temporal expressions from a single time stamp or interval to a more realistic set of possible time intervals, usually results in a wider range of temporal operations, such as *overlaps, meets*, and *contains*. Regardless of the overheads, such modeling scenario confirms the assertion that FT-SWRL does not only help to represent fuzzy temporal information in OWL ontologies but can also help to improve the utilization of its existing temporal model operators.

3.5.1 FT-SWRL Model Design Approach and Motivation

As mentioned earlier, the aim of proposing the FT-SWRL extension is to provide a consistent representation model for managing both temporal data and the imprecise temporal expressions commonly found in domain facts. This is achieved by extending the formalisms of the Semantic Web Rule Language with fuzzy temporal constructs, syntactically defined as in a fuzzy temporal SWRL ontology model and built-ins. As such, FT-SWRL enabled ontologies are expected to consistently model such dynamic and uncertain phenomena that are otherwise difficult to manage using the basic OWL/SWRL constructs or temporal models. For instance, in the current OWL/SWRL definitions, expressions involving approximate temporal assertions such as *around 4pm, about 4 hours, few weeks ago,* etc. cannot be easily represented nor efficiently extracted from OWL ontologies.

In essence, the FT-SWRL extension is designed to provide the syntactical and semantic extensions in the existing SWRL formalism for handling imprecise temporal information by providing fuzzy-grounded classes and property constructs, built-ins, and annotation properties to encode fuzzy-temporal information in the SWRL formalism. We extend the SWRL-Temporal model with such commonly utilized Imprecise Temporal Expressions (ITEs) found in descriptions of domain facts. This is particularly important where the knowledge to be represented is in the form of expert opinions or natural language narratives of experts, with no prior knowledge of ontological domain modeling. As in the motivational case study [45], where we use SWRL-enabled ontologies to model the farming practices of underutilized crops — a field that significantly relies on local farmers expertise alongside the scientific knowledge. Consider the representation of following facts on underutilized crops (Bambaranut) domain:

- Bambara groundnut requires a growth period of *about 110 to 150 days* for the crop to be developed.
- Bambara beans take *around 7 to 15 days* to germinate. Seed stored for *about 12 months* germinate well, but longer storage results in loss of viability.
- Flowering *starts 30 to 35 days* after sowing and may continue *until* the end of the plants life.
- Pod and seed development take place *approximately 30 to 40 days* after fertilization. This takes *up to 30 days after* fertilization. The seed develops *during a further 10 days*. [176]

While handling time-related data alone or managing uncertainties in a domain knowledge are in themselves difficult tasks. Nonetheless, there is a need to model such real-world issues that require the representation of time changes and the uncertainties brought about by these changes or imprecise temporal relations between events. For example, a given class (GrowthStage) in our Underutilized-Crops ontology[45] can have different individual instances asserted depending on the planting date of a crop (represented as *hasDateOfSowing* datatype property). However, temporal measurements and time itself are not fixed or definite data and therefore the time inputs are merely estimates. As such the use of descriptors, such as *about*, *approximately*, *around* followed by a time value merely confirms that such information is imprecise — and hence the need to be modeled as such. Other common application areas of fuzzy-temporal modeling for handling imprecise temporal expressions include the medical domain, multimedia, market trends analysis, and natural language applications in virtual assistants (e.g., Siri in Apple, Cortana by Microsoft and Google-Now), among others.

3.5.2 Preliminaries

As the use of ontologies in enterprise applications is becoming pervasive, effective representation and communicating of fuzzy domain facts cannot be overestimated. As such, various language extensions have been inspired by the fuzzy set theory to enable representing non-crisp or vague facts into ontology models. These extensions (discussed earlier in Section 2.11) were mainly rooted in the fuzzy extension of the underlying description logic, leading to the serial evolution of both OWL and SWRL fuzzy language extensions, including the Fuzzy-OWL[177], Fuzzy-SWRL [178], SWRL-Fuzzy [108], and Vague-SWRL [113] among others.

Similarly, though with a much lesser magnitude, temporal ontologies and language extensions have been proposed for handling time-related information in the semantic web. As presented in Section 2.11, available ontologies include the OWL Time ontology [150], which basically describes the general concepts of time as entities. This is due to the logic-based function-free approach of OWL – meaning that temporal arguments cannot be added to the supported binary relationships. Whereas, available temporal language extensions include the Temporal SWRL among others, which aim to provide basic constructs to describe temporal facts in a knowledge domain with some degree of consistency [129]. The SWRL-Temporal model provides a standard mechanism for representing and managing temporal information based on the Valid-time temporal model — commonly used to represent temporal information in knowledge-based systems [179].

3.5.2.1 The Valid-Time Temporal Model

A valid time temporal model [179] helps to provide a simple and consistent approach for modeling temporal information (called facts or propositions). In this model, a temporal proposition is either true or valid as specified in its associated timestamp — called the valid time. Such timestamps can be either specific time instants or intervals (time period between time instants). A special time-interval called Duration is characterized by two arguments: the Granularity — which is a unit measure for temporal data e.g,. days, hours, seconds, etc. and the Duration count — usually an integer. However, Duration can also be specified using two 'time instants' (with xsd:dateTime as arguments).

From the literature, basic temporal objects have been commonly classified into three distinctive references as follows:

(i) Points in time which defines a single temporal point on the timeline. Example: 13:00, now, date, etc.

(ii) Time Intervals defining the temporal relationship between two time-points. Example: 13:00 14:00, 23 December, etc.

(iii) Duration or relative expression of intervals. Example: 2 weeks, 6 years, many hours, etc. usually represented as counts of a time granularity.

Based on the valid-time temporal model, the Fuzzy Temporal SWRL model was proposed in [129] by defining two important entities: the SWRL Temporal Ontology and the SWRL Temporal Built-ins. We briefly discuss them below.

3.5.2.2 SWRL Temporal Ontology

The SWRL temporal ontology[4] defines the OWL constructs that can be used to represent the valid-time temporal model. It hierarchically defined the collection of entities that allow modeling interval-based temporal information in OWL ontologies. It has the default prefix: temporal and in its complete form, the ontology

[4]http://swrl.stanford.edu/ontologies/built-ins/3.3/temporal.owl

also defines the built-ins (SWRL temporal built-ins) for processing and reasoning about the SWRL valid-time temporal model. In the ontology (see fragment view in Fig. 3.11), a temporal fact is represented as an 'Extended Proposition' to separate the temporal ontology from the domain ontology. This helps to maintain consistency and allows easy manipulation of the temporal fragment of the ontology without affecting the main ontology.

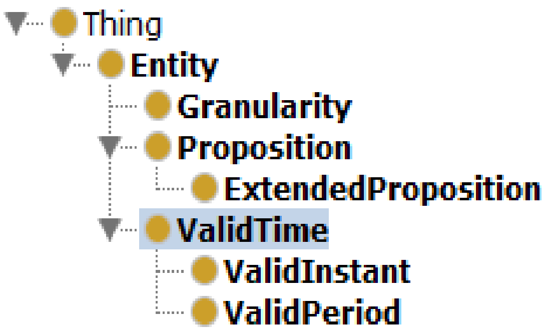


Figure 3.11: SWRL Temporal Ontology showing ValidTime class hierarchy

The ontology further defines individuals for the granularity class to include (Years, months, days, hours, minutes, seconds and milliseconds) and a set of built-ins that can be used in the SWRL rules to perform temporal reasoning in OWL ontologies. Built-ins defined in the ontology can be classified into three categories: (i) The duration operators — for reasoning about time durations, with constructs: *duration, durationLessThan, durationEqualTo, durationGreaterThan* and their inverses), (ii) the standard Allen temporal operators — for reasoning about qualitative temporal information using the calculus of binary relations on intervals. The built-in implementation has constructs, such as *equals, before, after, meets, metBy, overlaps, overlappedBy, contains, during, starts, startedBy, finishes, finishedBy,* and their inverses, and (iii) the add operator — implemented to achieve addition and subtraction comparison of time intervals. The construct is written as *temporal:add*. It should be noted here that, in the standard interval calculus, all intervals are assumed to have proper and distinct time, with clear-cut beginnings and ends. This limitation forms the basis of our study, as temporal events and time-related processes in the real-world are usually non-crisp and inexact. And representing them as exact facts will no doubt result in logically inefficient models and knowledge-based systems relying on such information can never be sufficiently intelligent.

3.5.2.3 SWRL Temporal Built-ins

A powerful feature of the SWRL formalism, is the ability to extend its definition with user-defined methods for writing application-specific rules. Similar to functions used in rule engines, the SWRL built-ins are predicates that accepts one or more arguments and operate on them during rule execution. Due to the limited temporal support in both OWL and SWRL, the SWRL Temporal Built-in Library is added as an expressiveness extension to the original SWRL definition. Defined as part of the SWRL-API's built-in libraries, the temporal built-ins are hierarchically defined as part of the SWRL temporal ontology. The temporal built-ins provide a rich set of temporal operators designed to allow temporal operations on information described using the temporal ontology. Thus, the built-ins allow temporal reasoning about OWL ontologies using SWRL rules.

Syntax and Semantics: In the basic mode, SWRL temporal built-ins operates on arguments supplied by the XML Schemas 'date' and 'dateTime' data types provided as xsd:String with values, such as second, hour, day, time, week, month, and year. These were also defined in the basic OWL temporal ontology (OWL Time)[5]. Whereas, in the advanced mode, the SWRL temporal built-ins work on time information that is completely encoded using the valid-time temporal model. As an example, a rule that asserts a 'Fellow' membership rank to existing workgroup members, with registration dates before the year 2000, can be written as:

Workgroupmember(?m), hasRegDate(?m, ?rgd), ***temporal****:before(?rgd, '2000')* $\longrightarrow$ *FellowMembers(?m).*

3.5.2.4 Fuzzy Sets and Membership Functions

In contrast to probability theory, the Fuzzy theory is a generalization that studies and facilitates analysis of uncertainties in systems where such uncertainties are

[5]https://www.w3.org/TR/2016/WD-owl-time-20160712/

born due to vagueness (fuzziness) in the available domain knowledge — rather than due to randomness (probability) alone [180, 181]. The logic being that assertion of ’truth’ or otherwise of a given fact can be represented by a varying degree on the closed interval [0, 1] — denoting the classical false and true values. A pool of real numbers denoted by (0, 1) in-between the interval represents the varying degrees of truth (w). Consequently, Fuzzy Sets [181] have been widely used for modeling uncertainties, where knowledge of a domain is incomplete or marred with vagueness. In contrast to crisp set theory, where an object simply belonged to a given set or otherwise, in fuzzy set theory, membership to a set is subjected to the given weight or degrees of truth (w). In fuzzy conceptualization, objects can belong (or otherwise) to a given class with some degree of certainty. We briefly highlight the formal definitions of fuzzy sets as follows:

Definition 1: In a classical set theory, the membership function (μ or MF) of an element (x) belonging to a given set (A) is represented thus:

$$\mu A(x) = 1 \iff X \in A, 0 \iff X \notin A. \tag{3.1}$$

However, in a fuzzy set theory, there is more to this crisp representation, where an additional information is provided to denote the degree of certainty that the element (x) belonged or otherwise to the given set (A). In such cases, the fuzzy membership function is written as:

$$\mu A(x) = 1 \iff X \in A,\ 0 \iff X \notin A,\ w\, if\ X\ partially\ belongs\ to\ A. \tag{3.2}$$

Where w is a weighted degree function such that $0 < w < 1$.

In essence, the membership function μA(x) is continuous in a fuzzy set theory with a range of [0, 1] = w called the *degree of truth for the membership*. An arbitrary curve is usually designed to represent the input space, also called *the universe of discourse* and the mapping of the membership value on this input space is the membership function (μ). The following also hold true in a fuzzy set theory:

$$\mu_A^-(x) = 1 - \mu_A^-(x). \tag{3.3}$$

$$\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x)). \tag{3.4}$$

$$\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x)). \tag{3.5}$$

We define a Fuzzy Temporal extension of SWRL as a fusion of these basic concepts with the aim of introducing a fuzzy representation and reasoning model for temporal information using the SWRL rules. Our work focused on the complete stages of the fuzzification process, as defined in [182], where we first define the linguistic terminologies and variables in the SWRL-FT ontology and new set of fuzzy temporal Built-ins — presented in Section 4.4. We chose suitable membership functions for some selected imprecise temporal expressions and demonstrate how we can generate their corresponding fuzzy values based on the membership functions as presented in Section 4.4.3. In the next sub-section, we present the state-of-the-art of the fuzzy temporal representation and reasoning system.

3.5.2.5 The FT-SWRL Model Scope

Since it is such an enormous task to generalize all fuzzy set theories into an ontology rule language extension, we adopt the bottom-up approach where we begin with introducing fuzziness from the peripherals of SWRL by extending existing temporal model. This has the advantage of working with existing tools during implementation and without introducing inconsistencies to main ontologies. As such, our proposal does not focus on modeling rudimentary time concepts and terminologies but relies mainly on existing standards that are already compatible with SWRL, such as the OWL time ontology and XML schema temporal data types — the xsd:(date, dateTime, and duration), among others. Moreover, the SWRL temporal model, presented in [129], serves as the basis for the adopted crisp valid-temporal model. In essence, our primary focus is providing an extended abstract syntax and semantics for fuzzy temporal representation in the SWRL language. This allows modeling imprecise temporal facts using fuzzy temporal modifiers (a collection of fuzzy terms and variables) defined as constructs. We focused on the Semantic Web Rule Language (SWRL) and its fuzzy temporal extension largely due to its semantic integration with OWL and the ability of SWRL to assert domain knowledge into ontologies, as well as extract them using its query functionality available as an SQWRL (SWRL query language) [183].

3.5.3 The FT-SWRL Ontology: Representing Fuzzy-temporal Facts using OWL/SWRL

The fuzzy-temporal extension of the SWRL language is designed to support modeling and reasoning with imprecise temporal expressions (ITEs) in OWL ontologies. To this end, a fuzzy temporal ontology is developed to define a consistent model that can be used to represent all fuzzy-temporal facts. The ontology also includes the definition of relevant SWRL built-ins, which extends the existing SWRL built-in library, to allow reasoning operations about the modeled fuzzy-temporal information. As explained earlier, the original SWRL temporal model follows the valid-time temporal model — where temporal facts are modeled as intervals of time-points. Hence, as the name implies, the FT-SWRL extension basically extends the syntax and semantics of the SWRL temporal model (more specifically the 'Advanced SWRL Temporal model') with appropriate fuzzy syntax and semantics.

While classical SWRL temporal ontology serves as a reference standard for modeling crisp temporal information, the SWRL-FT ontology is its carefully extended version with the ability to handle imprecise temporal expressions in domain knowledge representation. The ontology has a default prefix: *fuzzytemporal*, and the hierarchical representation of the ontology is presented in Listing 2 below:

Listing 3.2: The SWRL Fuzzy-Temporal Ontology (SWRL-FTO)

```
owl:Thing
owl:Entity
owl:Proposition
temporal:ExtendedProposition \equiv TemporalProposition (Time related event class
    )
temporal:hasValidTime           //object property
temporal:hasDuration            //object property
temporal:ValidTime       (Valid Crisp times of events: instants or period)
temporal:hasDuration     //object property
temporal:hasGranularity //object property
temporal:ValidInstants   (event occurs at a single instant)

temporal:hasTime(xml:dateTime)   //Datatype property
```

```
temporal:ValidPeriod (event occurs over an interval of time)
temporal:hasStart(xml:dateTime) //Datatype property
temporal:hasFinish(xml:dateTime) //Datatype property
temporal:Duration       (Temporal Expressions denoting interval-based temporal
    information)
temporal:hasCount (xml:Integer)
temporal:hasGranularity
temporal:Granularity    (years, months, days, hours, minutes, secs, milliseconds)
(Temporal)
________________________________________________________________________________
(Fuzzy temporal)
fuzzytemporal:FuzzyTemporalProposition  (Vague temporal fact)
fuzzytemporal:hasFuzzyTime      //object property
fuzzytemporal:hasFuzzyModifier  //object property
fuzzytemporal:hasFuzzyDuration  //object property
fuzzytemporal:FuzzyTime (Vaguely known time data)
fuzzytemporal:hasFuzzyDuration  //object property
fuzzytemporal:FuzzyTimeInstant \equiv FuzzyTimePoint
fuzzytemporal:FuzzyTimePeriod
fuzzytemporal:minFuzzyTime
fuzzytemporal:maxFuzzyTime
fuzzytemporal:FuzzyDuration     (Vague interval-based temporal information)
fuzzytemporal:hasFuzzyCount     //object property
fuzzytemporal:hasFuzzyGranularity       //object property
fuzzytemporal:FuzzyCount        (cycles, times, twice, several, many, long-time,
    this, next, last, etc.)
temporal:hasCount (XML:Integer) //Datatype property
temporal:hasGranularity (temporal:Granularity)  //Datatype property
fuzzytemporal:FuzzyGranularity  (weeks, weekend, fortnight, quarter, noon, etc.)
fuzzytemporal:SetGranularity    (Yearly, Monthly, Weekly, daily, hourly,
    perMinute,
perSeconds,  perHour, perWeek, perYear)
fuzzytemporal:DateGranularity   (past, present, currently, recently, nowadays,
    ago, since, lately, earlier, etc.)
fuzzytemporal:FuzzyModifiers    (Imprecise Temporal Expressions - ITEs e.g. about
    , around, approx, within, a few, several, many, until, always, very).
fuzzytemporal:hasWeightedValue \equiv hasWeightDegree)[0,1]     // Functional
    datatype prop.
fuzzytemporal:hasMembershipFunction (args: weighted sum)        //datatype prop.
fuzzytemporal:WeightValues              (0 < w < 1) // Possible Weight Intervals
fuzzytemporal:MembershipFunction
fuzzytemporal:mfName    (gaussmf, sigmoidmf, gbellmf, etc.)
fuzzytemporal:mfCurve   (plots of membership functions)
```

The Listing 3.2 above defines the OWL entities as a reference model for representing fuzzy temporal domain knowledge in OWL ontologies. The listing also shows the hierarchical layout of the OWL entities with the type of relationships that exists between them. For obvious reasons, it began with the original temporal entities defined in the SWRL temporal model followed by the extended fuzzy-temporal ones defined as the SWRL fuzzy-temporal model. These include the fuzzy temporal proposition, the fuzzy modifiers and their membership function, fuzzy granularity, fuzzy counts, fuzzy time instants and fuzzy durations.

3.5.3.1 FT-SWRL Model Entities: Classes, Properties, Domain and Ranges

A summary of the fuzzy temporal entities is presented in Table 3.1: Fuzzy Temporal Classification with their properties, and in Table 3.2: Fuzzy Temporal Relations highlighting their types, domain, and range.

Fuzzy Temporal Class	Sub-classes	Properties
FuzzyTemporal-Proposition		hasFuzzyTime hasFuzzyModifier hasFuzzyDuration
FuzzyTime	FuzzyTimeInstant FuzzyTimePeriod	hasFuzzyDuration
FuzzyTimePeriod	minFuzzyTime maxFuzzyTime	
FuzzyDuration		hasFuzzyCount has-FuzzyGranularity
FuzzyCount		hasCount hasGranular-ity
FuzzyGranularity	SetGranularity Date-Granularity	
FuzzyModifiers		hasWeightedValue has-MembershipFunction
WeightValues		
MembershipFunction	mfName mfCurve	

Table 3.1: Summary of SWRL Fuzzy Temporal Entities

Note: Even though the essence of the FT-SWRL extension is to handle imprecise temporal extensions found in domain language narratives, we still introduce some added concepts (e.g. week, quarter, and fortnight) to the original SWRL temporal ontology. Moreover, to accommodate our new constructs, new container classes need to be added leading to the design of a new fuzzy-temporal ontology from scratch. Hence the above description is that of the fuzzy temporal ontology containing an extended temporal ontology that is set towards modeling natural language description of domain knowledge in OWL ontologies. This approach, will no doubt allow flexible modeling of time-related events.

3.6 Summary

In this chapter, a framework for representing domain knowledge using OWL and SWRL rules was presented in Section 3.2. The framework describes the methodologies involved in the UC-ONTO engineering and standardization approaches as well as the use of competency questions to generate ontology concepts. The experimentation of ontology development with Protégé is also presented highlighting the modeling, standardization, extension with rules and querying the ontological knowledge model. Ontology utilization approach through ontology-based semantic search engine is further presented in Section 3.4. The methodology highlights the Onto-CropBase development approaches, which includes knowledge base integration, interface and search engine design as well as the selection of the mediator component for managing the front and back ends of the tool. Lastly, the chapter presented the modeling approaches of the FT-SWRL model in Section 3.5, highlighting the model design, motivational examples from the crops domain followed by the fuzzy temporal ontology model and its entity hierarchies.

In the following chapter, experimentation results are discussed describing the underutilized crops ontology (UC-ONTO), the Onto-CropBase tool and the FT-SWRL model as case studies.

Fuzzy Temporal Relation	Role Type	Domain	Range
hasFuzzyTime	Object property	FuzzyTemporal-Proposition	FuzzyTime
hasFuzzyModifier	Object property	FuzzyTemporal-Proposition	FuzzyModifiers
hasFuzzyDuration	Object property	FuzzyTemporal-Proposition FuzzyTime	FuzzyDuration
hasFuzzyCount	Object property	FuzzyDuration FuzzyCount	
hasFuzzyGranu-larity	Object property	FuzzyDuration	FuzzyGranularity
hasWeightedValue	Datatype prop. (Functional)	FuzzyModifiers	WeightValue (xml:Decimal)
hasMembership-Function	Datatype prop.	FuzzyModifiers	MembershipFunction
hasCount	Datatype prop.	FuzzyCount	xml:Integer
hasGranularity	Datatype prop.	FuzzyCount	temporal:Granularity

Table 3.2: SWRL Fuzzy Temporal Relations summary showing Domain and Range

Chapter 4

Results: Case Studies

4.1 Introduction

This chapter presents case study implementations of the underutilized crops ontology (UC-ONTO), the Onto-CropBase semantic search engine and the FT-SWRL model ontology implementation. The chapter is structured into the following main sections: Section 4.2, presents the UC-ONTO[1] model highlighting its knowledge gathering and conceptualization of relevant terminologies. UC-ONTO standardization techniques are also presented in Section 4.2.6 followed by its extension with the SWRL rules in Section 4.2.7.

This is followed by the ontology utilization case study in Section 4.3, which presents the Onto-CropBase semantic search engine with an overview and pictorial presentation of its basic functionalities in Section 4.3.1. The FT-SWRL model implementation is then presented in Section 4.4 highlighting the pioneer SWRL fuzzy temporal ontology implementation in Section 4.4.1 and the fuzzy temporal built-ins providing the semantic definition of the ontology concepts in Section 4.4.2. The reasoning paradigm of the model is presented in Section 4.4.3 and lastly conclude in Section 4.5 with the chapter summary.

[1]Details of UC-ONTO design and development is published in our papers: *(i)Advancing Underutilized Crops Knowledge using SWRL-enabled Ontologies — A survey and early experiment* [184] and *(ii) An Ontological Approach for Knowledge Modeling and Reasoning Over Heterogeneous Crop Data Sources* [61].

4.2 Case Study 1: Ontology Engineering — The Underutilized Crops Ontology (UC-ONTO) Development, Standardization and Extension

4.2.1 The UC-ONTO Development: Domain Knowledge Gathering and Conceptualization

As earlier mentioned, underutilized crops are also significantly under-represented [8]. As such, standardized information on the underutilized crops is generally scarce and considering that major crop ontologies are usually in the OBO format, a knowledge gathering process from scratch has to be considered. This is to allow modeling of those specialized concepts and data-values that are most relevant to the Underutilized Crops. For the development of the Underutilized crops ontology as case study, various conceptualization approaches were proposed or tested and eventually utilized to generate the pioneer OWL2-based Crops ontology with added user-defined rules in Semantic Web Rule Language (SWRL). These conceptualization approaches, as discussed in details in Chapter 3 include: (i) Conceptualization from unstructured data such as PDF, Word files, Excel sheets and research notebooks. (ii) Conceptualization from structured heterogeneous data sources such as XML and relational databases, and (iii) Conceptualization from Competency questions, which describes a pre-requisite queries for the ontological knowledge base. In what follows here, the resulting knowledge gathered through these approaches is presented in details.

4.2.2 Conceptualization from Scratch

The knowledge gathering task was thus structured into three major stages as follows: first going through the available data sources to extract the domain concept names and relevant terminologies (e.g. Bambara Groundnut) that will be asserted as individuals of the $UnderutilizedCrops$ class. At the same time, noting other related terminologies such as *Rainfall, Temperature, Landrace,* etc. The second stage involves assigning the relationships that exists between these concepts — the object properties and between concepts and their data values — the datatype properties. Lastly, a hierarchy is defined and individual instances that shares common object properties into classes and subclasses. In collaboration with the domain experts from CFF[2], these steps were repeated to create the initial version of the underutilized crops ontology-based knowledge model, which was continuously enhanced as the cropbase knowledge evolved.

A summarized Bambara Groundnut vocabulary, extracted from unstructured data sources, is shown in Table 4.1 and the hierarchical representation of the concepts is depicted in Fig. 4.1.

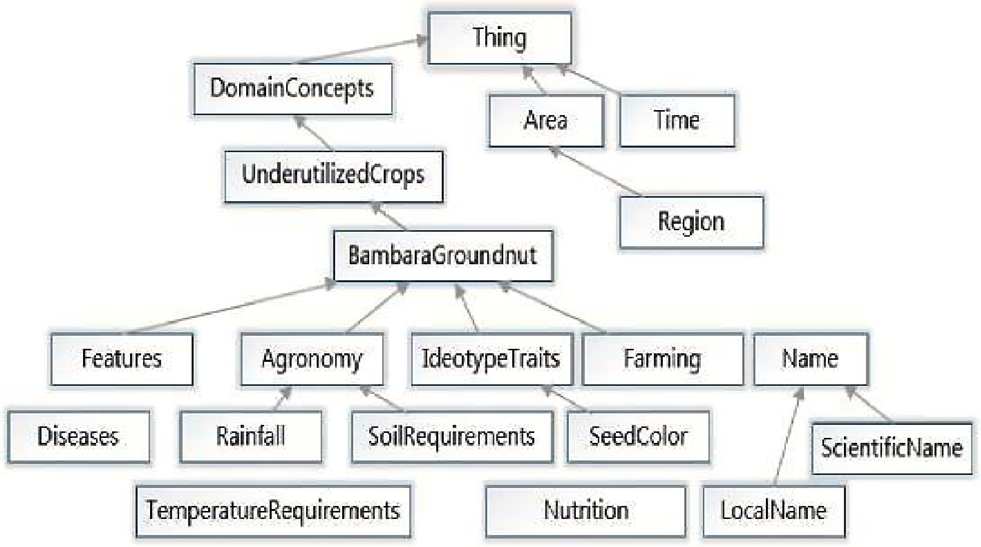


Figure 4.1: Key Concepts of the Underutilized Crops Ontology

4.2.3 Conceptualization from Heterogeneous data sources

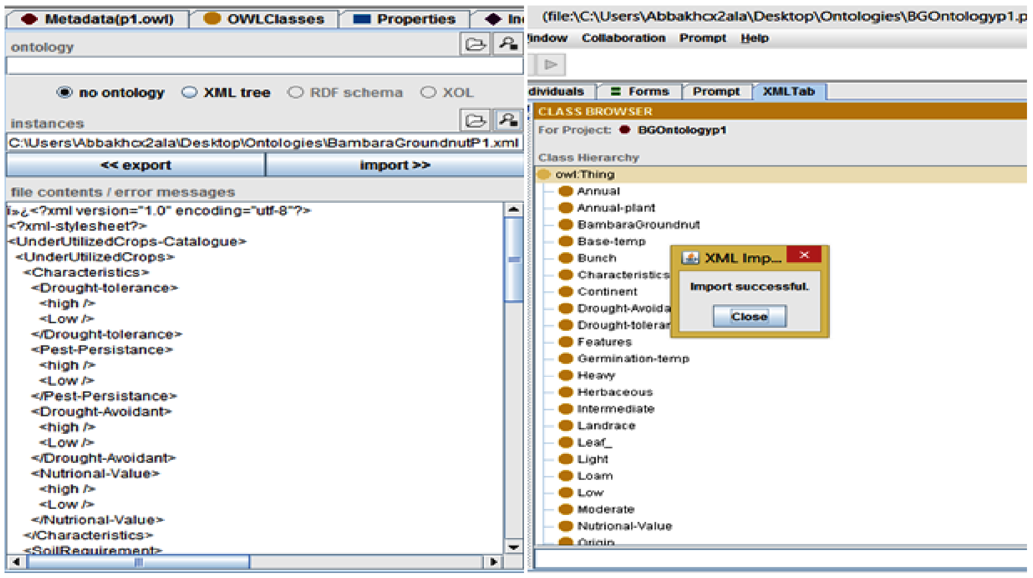
As explained in Section 4.2.2, available information on underutilized crops is usually scattered across various sources and formats — both structured and unstructured. Moreover, as majority of the available knowledge exist in the form of published articles and XML-based web documents, an 'XML to OWL conversion' was used where possible to generate domain-related concepts from XML documents to the web ontology format. This is achieved through the *XML2OWL*

[2]Crops for the future: http://www.cropsforthefuture.org/

Bambara Groundnut (BG) Vocabulary, Source: CFFRC	
Class	$BambaraGroundnut$
Super Class	$UnderutilizedCrops$
Ancestor Class	Family Legumes
Type	Bunch Seed Crop
Alias	Vigna Subterranea
Region	West Africa
Properties	High nutritional value, Pest persistent Crop, Highly tolerant
Features	Leaf, Stem, Roots, Pods, Seed
Soil Requirements	**PH level**:$5.0 - 6.5$, **Soil Type**:Loamy (heavy loam, light loams), Sandy soil
Rainfall	Moderate ($500 - 1200$ mm) Seasonal rain
Temperature	**Optimum temp.**:$20 - 28^oC$ **Base temp.**:$10 - 12.3^oC$ **Germination temp.**:$30 - 35^oC$
Flowering	takes $30 - 50$ days and depends on: day length, temperature and Landrace. BG is a 'short-day' crop (grows at elevation up to 1600 m).
Germination: **Harvesting**	Emergence takes about $7 - 15$ days. usually between $90 - 170$ DAS (days after sowing).
Pests **Pests Control**	Spidermites Spidermites (Tetranyches Cinnabarinnus), can be controlled by Pesticides e.g. Phytosiulus Persimilis.
Nutrients and Minerals	Protein: $16 - 25\%$, Carbohydrate: $42 - 65\%$, Lipid (oil): 6%. Dominant minerals — Ca, K, Mg, Na, P, Cu, Fe, Zn.
Growth Phases:	Vegetative phase, Reproductive phase(Phase has stages).
Cultivation	BG is traditionally cultivated by small-scale farmers (majority Women Farmers) mostly in extreme tropical environments without access to irrigation and/or fertilizers.
Growth	Depends on Landrace and Environmental Condition (e.g. Drought, Cold, Heat, Soil Moisture (same as Soil water), Evapotranspiration) BG is drought tolerant (not drought escape or avoidant) i.e. Maintains positive tugor at low water potential. BG needs moderate Soil moisture.
Life Span	Averagely 4 months after sowing (120 DAS) or when leaves begin to turn yellow in color.
Purpose/Uses	For Human consumption

Table 4.1: Summarised Bambara Groundnut Vocabulary

tab available in the Protégé ontology editor, which can generate OWL concepts directly from XML files. As shown in Fig. 4.2, the conversion process simply represent every XML node in the XML tree as a class in the resulting .owl file and the lists or slots of the XML nodes, are asserted as relations. However, due to the diverse nomenclature of the resulting concepts, we import some upper-domain ontologies and then undertake the process of ontology merging, which incorporates the matching and alignment of the resulting OWL concepts with standardized terminologies from the upper-domain ontologies.



C:\Users\Abbakhcx2ala\Desktop\Ontologies\BambaraGroundnutP1.xml

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet?>
<UnderUtilizedCrops-Catalogue>
 <UnderUtilizedCrops>
  <Characteristics>
   <Drought-tolerance>
    <high />
    <Low />
   </Drought-tolerance>
   <Pest-Persistance>
    <high />
    <Low />
   </Pest-Persistance>
   <Drought-Avoidant>
    <high />
    <Low />
   </Drought-Avoidant>
   <Nutrional-Value>
    <high />
    <Low />
   </Nutrional-Value>
  </Characteristics>
  <SoilRequirement>
```

```
owl:Thing
  Annual
  Annual-plant
  BambaraGroundnut
  Base-temp
  Bunch
  Characteristics
  Continent
  Drought-Avoida
  Drought-tolerar
  Features
  Germination-temp
  Heavy
  Herbaceous
  Intermediate
  Landrace
  Leaf_
  Light
  Loam
  Low
  Moderate
  Nutrional-Value
  Origin
```

Import successful.

Figure 4.2: Generating OWL concepts from XML data source using XML2OWL

In the case of dealing with other sources such as text documents and relational database (RDB) tables, we further utilize the features of *XML-Tab* — another plug-in available in Protégé. The 'XML tab' help to convert the source documents into XML before being passed into the *XML-to-OWL* converter. Though 'RDB to OWL' conversion may be possible, the process was found to be overly tedious and crude. As such, the XML files have to be created from the data stored in text files and relational tables using the XML-Tab plug-in. In the article, *An ontological approach for knowledge modeling and reasoning over heterogeneous crop data sources* [61], we discuss the XML-to-OWL conversion of XML documents to OWL concepts in great details, highlighting the step-wise procedures as well as the merging process of the resulting ontology fragments.

4.2.4 Conceptualization from Competency Questions

As the goal of developing the UC-ONTO is partly to serve as a knowledgebase for a farming decision support system, there is a need to identify the basic questions that the intended knowledgebase must answer. To achieve this, we identify the competency questions for our UC-ONTO based on the possible queries that a farming system should answer from target users, i.e. farmers, researchers, and interested public. As such, possible competency question can be elicited thus: *Which underutilized crop can be grown with the least seasonal rainfall?*. Similarly, the competency questions designed in the early stage of our ontology development were used to elicit the domain concepts while interacting with domain experts as respondents. However, for effective knowledge elicitation and ease of modeling, the questions were limited to a particular crop, the Bambara groundnut.

To further highlight the conceptualization process, some of the classes, properties and example individuals that can be immediately deduced from each question were shown in the following table. Note that for brevity, possible classes and properties already defined were not repeated as the questions list progress.

Competency Question	**Possible Classes**	**Object Property**	**Datatype Property**
What are the commonly cited Underutilized Crops (UC)?	Crops, UnderutilizedCrops	hasMembers	
Which UCs characteristics should a Farmer consider when planting a particular UC?	UC-Characteristics, Farmer	has-Characteristics, isGrownBy	hasPlantingDate
Is Bambara groundnut a Seasonal or an Annual plant?	CropType, Bambara-Groundnut	hasCropType, is-a, isMemberOf	hasType

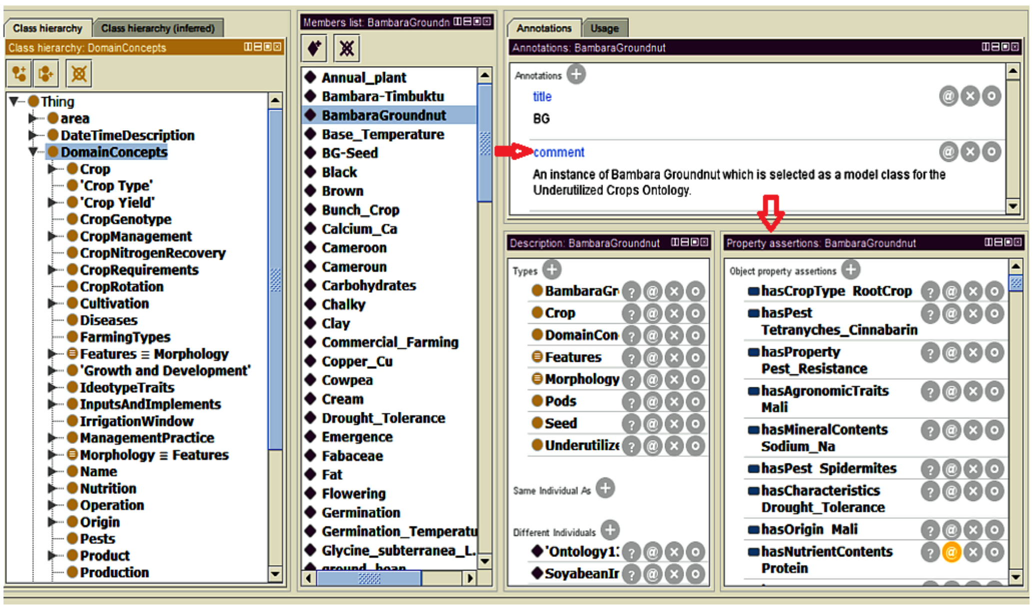
Competency Question	**Possible Classes**	**Object Property**	**Datatype Property**
What are the Soil types/Temperature/Rainfall requirements for an optimum yield of an UC?	Soil, Temperature, Rainfall, Yield	hasRequirements, hasSoilTypes, hasTemperatureRequirement, hasRainfallRequirement	
What are the best conditions for BG farming			hasOptimumTemp, hasOptimumRainfall, hasOptimumSoilType
What other Regions of Cultivation match the farming centers of Bambara Groundnut?	CultivatnRegion, FarmingCenters, EconomicRegion	hasCultivationRegion	hasFarmingCenter
Which Country has the highest yield of Bambara Groundnut in the year 2016?	Area, Country, Territory, Time, Year, Duration, Week, Calendar	hasYieldCount, belongsTo	hasCountry, hasDuration
What are the good Landraces of Bambara groundnut for Nigeria?	Landraces	hasLandrace	hasCountry
Does Bambara groundnut nutrients make up a complete food?	Nutrition, Food	hasNutrientContents	hasCount, hasUses

Competency Question	Possible Classes	Object Property	Datatype Property
What are the nutrition components of a particular Crop?	Carbohydrate, Minerals, Protein, Fats, Lipid		
Which part of Bambara Groundnut stores the food?	FoodPart, Features	hasPart, hasFeatures	
How many days does it take for Bambara groundnut to flower/produce food/harvest?	GrowthStages, Harvest, Days	hasGrowthStage, hasHarvestType	hasDaysAfterSowing
What happens to the yield of Bambara groundnut if the annual or Seasonal rainfall is moderate?	Rainfall (Seasonal, Annual), Category (Moderate, Heavy, Low)	hasRainfall, hasYield	hasAnnualYieldCount, hasSeasonal-YieldCount
What type of root does Bambara groundnut possess to survive the drought?	CropFeatures (Leaf, Root, Stem), Draught	hasFeature	hasRootType, hasLeafType, hasStemType, hasWeather-Problems
What are the unique features of Bambara groundnut as compared to other Legumes?	CropType (Legume, Herbaceous, etc)	hasCropType	
What are the commercial products of Bambara groundnut?	Products, Uses, Commercial-Products	hasUses	hasCommercialProduct, hasCommer-cialValue

Competency Question	**Possible Classes**	**Object Property**	**Datatype Property**
Does Bambara groundnut grows better in a Mixed farming or Crop Rotation?	FarmingType (Mixed, CropRotation, etc)	hasFarmingType	
What are the best candidates for mixed farming with Bambara groundnut?			hasMixedFarming-Candidate
What are the common Pests and Diseases of Bambara groundnut?	Pests, Disesaes	hasPests, hasDiseases	isDiseaseOf, isPestOf
What other names is Bambara groundnut called around the World?	Name (ScientificName, CommonName, etc)	hasName	hasScientificName, hasOther-Names, isLocalNameOf, isScientific-NameOf
Which farm input or fertilizer is often used for Bambara groundnut?	FarmInputs (Fertilizer, Implements)	hasFarmInputs	hasFertilizer, hasBestInput

Table 4.2: Competency Questions for UC-ONTO

With the help of domain experts and existing knowledge sources, we explore the questions and answers to list out relevant terminologies and concept names as possible classes and properties for the ontology. Finally, the listed concepts were then hierarchically organized and their relationships asserted to complete the development of the ontology fragments. Example fragments for our UC-ONTO include the *Naming ontology, Nutrition ontology, Agronomy, Cultivation* and *Production ontology*, among others. These modular fragment of ontologies were assembled together to form the UC-ONTO.


Class hierarchy
Class hierarchy (inferred)
Class hierarchy: DomainConcepts
Members list: BambaraGroundn
Annotations
Usage
Annotations: BambaraGroundnut
Description: BambaraGroundnut
Property assertions: BambaraGroundnut
Types
Same Individual As
Different Individuals
Object property assertions

Thing
- area
- DateTimeDescription
- DomainConcepts
  - Crop
  - 'Crop Type'
  - 'Crop Yield'
  - CropGenotype
  - CropManagement
  - CropNitrogenRecovery
  - CropRequirements
  - CropRotation
  - Cultivation
  - Diseases
  - FarmingTypes
  - Features ≡ Morphology
  - 'Growth and Development'
  - IdeotypeTraits
  - InputsAndImplements
  - IrrigationWindow
  - ManagementPractice
  - Morphology ≡ Features
  - Name
  - Nutrition
  - Operation
  - Origin
  - Pests
  - Product
  - Production

Annual_plant
Bambara-Timbuktu
BambaraGroundnut
Base_Temperature
BG-Seed
Black
Brown
Bunch_Crop
Calcium_Ca
Cameroon
Cameroun
Carbohydrates
Chalky
Clay
Commercial_Farming
Copper_Cu
Cowpea
Cream
Drought_Tolerance
Emergence
Fabaceae
Fat
Flowering
Germination
Germination_Temperatu
Glycine_subterranea_L.
ground_bean

Annotations
title
BG
comment
An instance of Bambara Groundnut which is selected as a model class for the Underutilized Crops Ontology.

Types
BambaraGr
Crop
DomainCon
Features
Morphology
Pods
Seed
Underutilize

Different Individuals
'Ontology1:
SoybeanIr

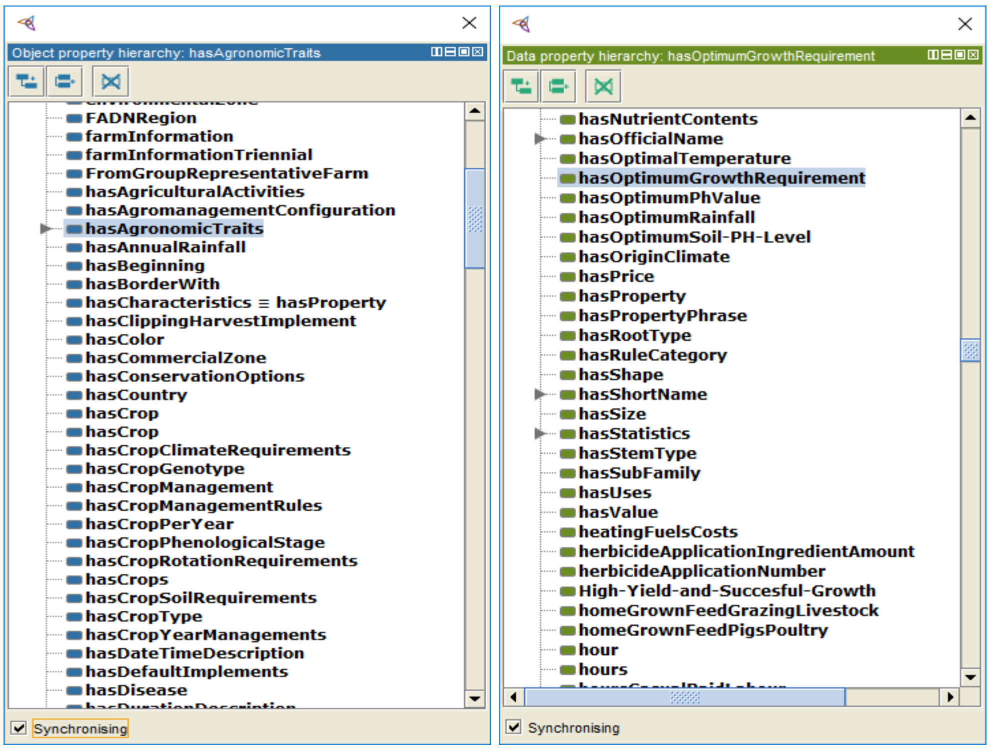
Object property assertions
hasCropType RootCrop
hasPest Tetranyches_Cinnabarin
hasProperty Pest_Resistance
hasAgronomicTraits Mali
hasMineralContents Sodium_Na
hasPest Spidermites
hasCharacteristics Drought_Tolerance
hasOrigin Mali
hasNutrientContents Protein

Figure 4.3: Fragment of UC-ONTO in the Protégé ODE showing class hierarchy (left tab), Individual instances (middle tab), Annotations (top-right), Individual Descriptions and Properties (bottom-right tab).

From Fig. 4.3, the DomainConcept class is shown implemented as the ancestral class for all the crop-related concepts. The class hierarchy or terminology assertion box (T-Box) contains all the major classes and their sub-classes. The arrow besides each concept highlights that it is a super-class with sub-classes under it. The UC-ONTO contains underutilized crops knowledge ranging from crop features, cultivation requirements, crop management practices, Pests and Diseases, Origin, Production, Uses, etc. Relationships between the concepts were added to express their roles and membership restrictions as shown in the 'Object properties list' of Fig. 4.4 (left). Where as data types that express the property values of Individual instances is shown in the 'Datatype Properties list' of Fig. 4.4 — on the right.

4.2.4.1 UC-ONTO Visualization

Due to the nature of logical axiomatization of ontology assertions, visualization of concepts is of great importance. The large collection of entities and individu-



Object property hierarchy: hasAgronomicTraits

- FADNRegion
- farmInformation
- farmInformationTriennial
- FromGroupRepresentativeFarm
- hasAgriculturalActivities
- hasAgromanagementConfiguration
- hasAgronomicTraits
- hasAnnualRainfall
- hasBeginning
- hasBorderWith
- hasCharacteristics ≡ hasProperty
- hasClippingHarvestImplement
- hasColor
- hasCommercialZone
- hasConservationOptions
- hasCountry
- hasCrop
- hasCrop
- hasCropClimateRequirements
- hasCropGenotype
- hasCropManagement
- hasCropManagementRules
- hasCropPerYear
- hasCropPhenologicalStage
- hasCropRotationRequirements
- hasCrops
- hasCropSoilRequirements
- hasCropType
- hasCropYearManagements
- hasDateTimeDescription
- hasDefaultImplements
- hasDisease

Synchronising

Data property hierarchy: hasOptimumGrowthRequirement

- hasNutrientContents
- hasOfficialName
- hasOptimalTemperature
- hasOptimumGrowthRequirement
- hasOptimumPhValue
- hasOptimumRainfall
- hasOptimumSoil-PH-Level
- hasOriginClimate
- hasPrice
- hasProperty
- hasPropertyPhrase
- hasRootType
- hasRuleCategory
- hasShape
- hasShortName
- hasSize
- hasStatistics
- hasStemType
- hasSubFamily
- hasUses
- hasValue
- heatingFuelsCosts
- herbicideApplicationIngredientAmount
- herbicideApplicationNumber
- High-Yield-and-Succesful-Growth
- homeGrownFeedGrazingLivestock
- homeGrownFeedPigsPoultry
- hour
- hours

Synchronising

Figure 4.4: UC-ONTO Objects and Datatype Properties Implementation in Protégé ODE

als (e.g. 15,000 different crops-related concepts are available in the UC-ONTO) makes it necessary to provide single-entity views of ontology concepts inorder to make meaningful use of the results. This is in addition to the search systems designed to utilized the UC-ONTO — See Section 4.3 below. The Protégé ODE allows for efficient visualization of ontologies using two different plug-ins, the Onto-graph and OWL-viz as shown in the interface in Fig. 2.11.

An Onto-graph visualization of UC-ONTO concepts hierarchy is shown in Fig. 4.5. While a single-entity visualization of the 'BambaraGroundnut' instance highlighting its logically asserted object and datatype properties is shown in Fig. 4.6. As shown in both figures, each directed line represents a relationship between the two objects — similar to the RDF relation of subject-predicate-object. Relationships are color-coded and grouped together for ease of referencing. Such flexibility and logical structuring of the ontology knowledge model is one of its many advantages over traditional databases as discussed in the Introduction chapter. An OWL-viz visualization of the UC-ONTO is also provided in Fig. 5.17. More results of the UC-ONTO implementation and visualizations are presented in the UC-ONTO evaluation in Section 5.2 and the Appendix A1.

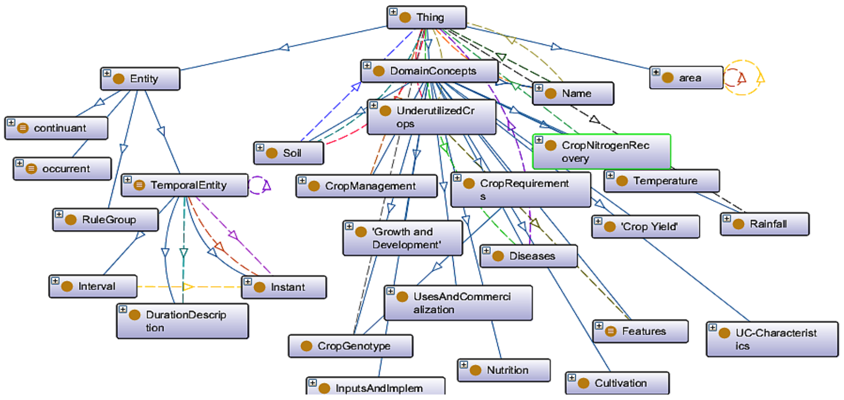


Figure 4.5: UC-ONTO visualization showing the Crops and related concepts hierarchy.

4.2.5 Ontology Reuses for UC-ONTO

As stated earlier, one of the benefits of developing ontology for a domain is knowledge reuse. Considering the relevant ontologies available for the crops domain (see Section 2.4), we utilize some existing ontologies by importing such ontologies having shared concepts with the underutilized-crop domain. Similar approach has been proposed in [185], where AGROVOC is used as a base vocabulary to develop the $CropOnt$ — describing crop production life cycle for individual farmers.

Relevant ontologies aligned to the UC-ONTO include the $Crop$ and $Farming$ Ontologies. These ontologies were merely utilized for their standard concepts in the agricultural domain and since they are of various versions and mostly very large, we import only small fragments where necessary. In cases, where the concepts from our initial knowledge gathering stage (Section 4.2.2) are sufficient, we proceed to simply align these concepts by using standardized terms from imported ontologies to add meaning to our existing underutilized crops concepts. Whereas, upper-level ontologies added as direct imports to the UC-ONTO include the *OWL-*

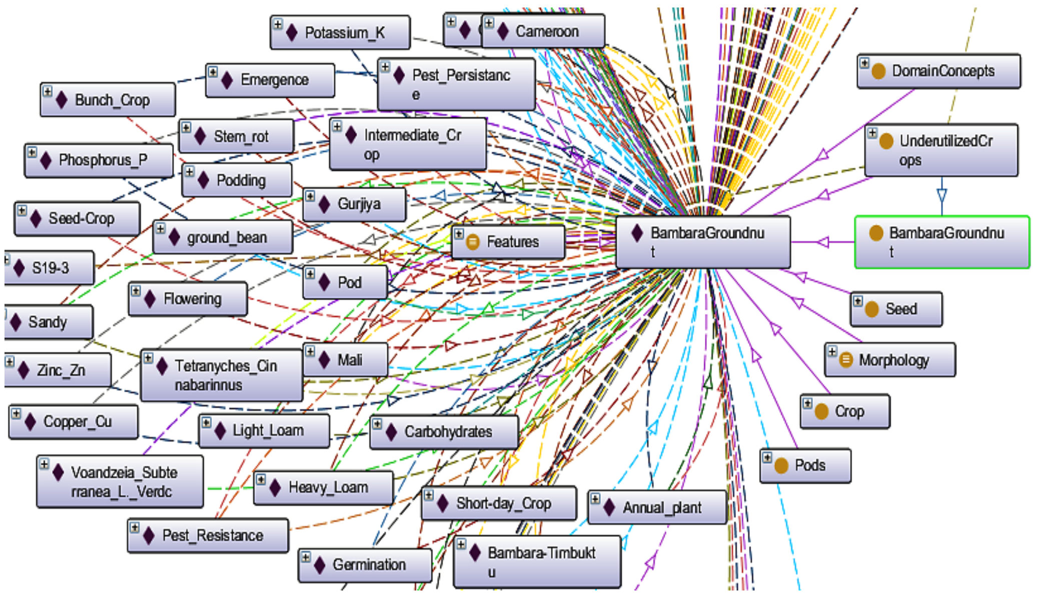


Figure 4.6: UC-ONTO visualization showing the Crops and related concepts hierarchy.

temporal ontology [150], SWRL built-ins [186], and the FAOs geopolitical ontology [187], among others. These ontologies, being significant in describing crops knowledge and mostly in the OWL format, can be easily integrated without posing compatibility problems or inconsistencies. Information integration from the reused OWL Time ontology, SWRL built-ins and FAO Geo-ontology for example, can be of the following scenarios: while expressing the DAS (Days of Sowing) of a crop in days or weeks, using the comparison operators (e.g. $>, <, =$, etc.) and when expressing the geolocation data of a given crop or related concept (e.g. region, country or place of cultivation), respectively.

4.2.6 UC-ONTO Standardization

Following the development approaches discussed in Section 4.2.1, various fragments of the UC-ONTO are developed and that means overlapping concepts are bound to exist among other minimal inconsistencies. As such there is a need for standardizing these ontology fragments into a coherent standard that can serve as a reference model for the underutilized crops domain. We first employ the ontology merging technique for streamlining the various versions of the UC-ONTO using the upper domain ontologies for referencing and alignment. We then annotate the domain-specific concepts with natural language definition of terms, adding annotations such as $labels$ and $comments$ to ambiguous domain concepts. This is followed by augmenting the ontology with expressive $if - then$ LP rules (using SWRL) to express deeper relationships and indirect conditions necessary for modeling domain knowledge. In what follows, we briefly summarize our approaches to the UC-ONTO standardization. The ontology integration with SWRL rules is presented in the next section.

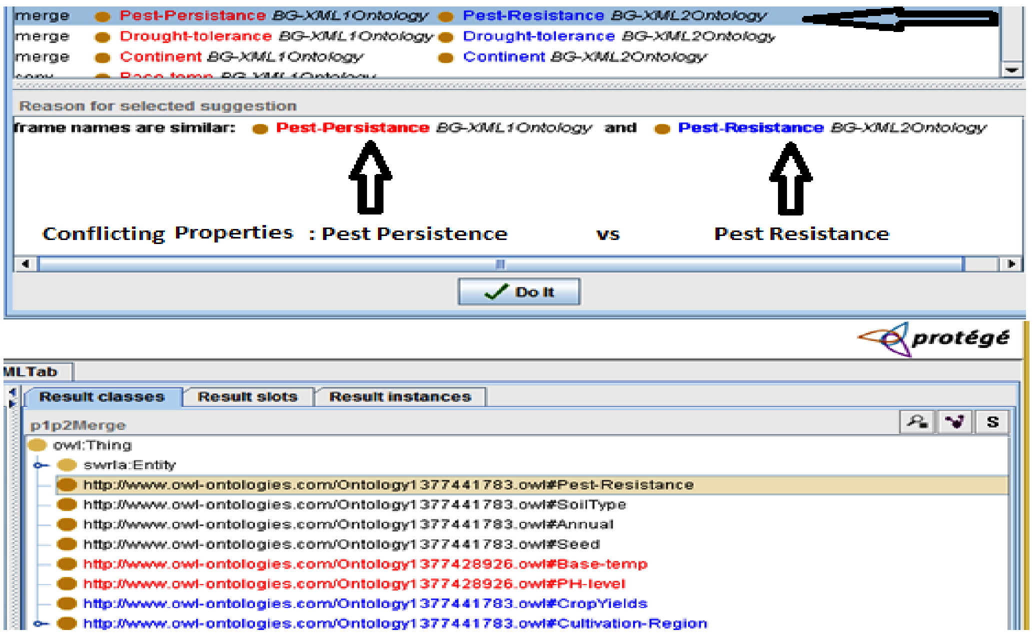
4.2.6.1 Alignment and Merging of UC-ONTO Fragments

During ontology development, a need may arise to merge two or more versions or fragments into a single ontology. As a means of dealing with heterogeneous ontologies, Protégé 3.5 provide the PROMPT [162] Plugin, a tool for merging, mapping, and or aligning two ontologies. In our case, we are concerned only with merging two ontologies to achieve homogeneous and coherent ontology. We employ the Prompt tool only for the sake of merging the ontologies resulting from XML sources. However, for the later stage of development, the merging is much simpler as less user intervention is required using the later versions of Protégé (e.g., Protégé 4.2). We discuss the theory and tools for ontology merging, matching and alignment in Section 2.5.4.1 of the Literature.

Merging Ontologies in Prompt involve a series of semi-automated steps, since it requires user intervention when there is a conflict. It is a tedious process for large source ontologies and a very tedious one if those ontologies have many similar classes. In its simplest form, the process involve three stages as follows: (i) Loading source ontologies and selecting the merging algorithm — we select the lexical matching algorithm, which detect the lexical similarities between classes (as shown in Fig. 4.7). (ii) The matching process — after scanning the two ontologies or when there is a conflict (such as identical class names with different properties), Protégé presents the user with merging suggestions for further actions. Finally, (iii) the Conflict resolution step — this is continuously repeated whenever Protégé encountered a conflict until all classes are merged. We present detailed descriptions including results and discussion of the merging process for

UC-ONTO development in [61]. In its simplest form, the merging process can be summarized thus:

- Loading the source ontologies.
- Selecting a merging algorithm — usually the lexical matching: which detects the lexical similarities between concepts, and then
- Authorization — to drive the merging process.
- Conflict Resolution — where there is a conflict (such as identical class names with different properties), the tool presents the various merge-options for user selection.



merge ● Pest-Persistance BG-XML1Ontology ● Pest-Resistance BG-XML2Ontology
merge ● Drought-tolerance BG-XML1Ontology ● Drought-tolerance BG-XML2Ontology
merge ● Continent BG-XML1Ontology ● Continent BG-XML2Ontology

Reason for selected suggestion
frame names are similar: ● Pest-Persistance BG-XML1Ontology and ● Pest-Resistance BG-XML2Ontology

Conflicting Properties : Pest Persistence vs Pest Resistance

p1p2Merge
owl:Thing
swrla:Entity
http://www.owl-ontologies.com/Ontology1377441783.owl#Pest-Resistance
http://www.owl-ontologies.com/Ontology1377441783.owl#SoilType
http://www.owl-ontologies.com/Ontology1377441783.owl#Annual
http://www.owl-ontologies.com/Ontology1377441783.owl#Seed
http://www.owl-ontologies.com/Ontology1377428926.owl#Base-temp
http://www.owl-ontologies.com/Ontology1377428926.owl#PH-level
http://www.owl-ontologies.com/Ontology1377441783.owl#CropYields
http://www.owl-ontologies.com/Ontology1377441783.owl#Cultivation-Region

Figure 4.7: Ontograph showing fragment of UC-ONTO hierarchy

Figure 4.7 shows the merging conflict presented to the user for intervention. The tool detects a similarity between two frames: 'Pest Persistence' and 'Pest Resistance'. The merging suggestions are shown in the lower tab, where the first entry shows the suggested name by the Protégé Prompt tool, where Pest Persistence should be merged into the Pest Resistance.

Note that, from our experience, we identify the following as common challenges while using the Protégé merging tool:

(i) Ensuring that all classes have been copied or merged successfully, which is difficult for large ontologies.

(ii) Realignment of the resulting ontology — this involves reclassification of merged concepts and setting the correct object properties.

(iii) Where the ontology is part of a hybrid-KB, it will need to be re-integrated and checked against inconsistencies with the concepts in the SWRL rule base.

4.2.6.2 Alignment of UC-ONTO with Upper-Crop Ontologies

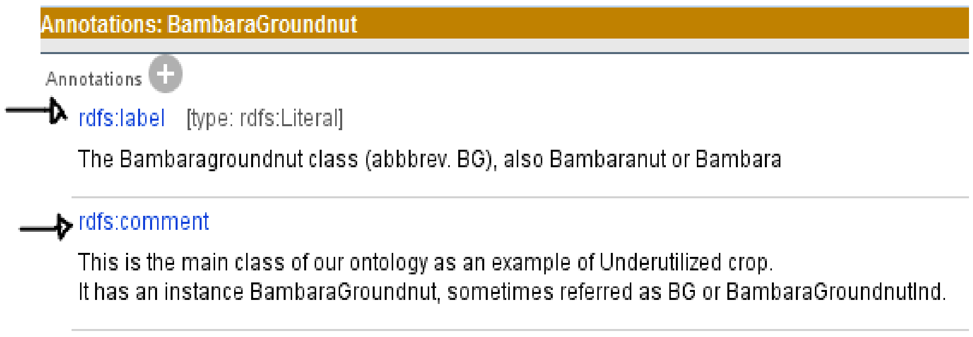
Due to their wide acceptance, upper-domain ontologies can be employed to provide standard concept definitions into lower ontologies. As mentioned earlier, relevant ontologies imported into our UC-ONTO include the farming and crops ontologies from 'Seamless project'[3], the OWL-time ontology, and FAOs geopolitical ontology, added as direct imports. As these ontologies are domain-independent and available in the OWL format, integrating them into the UC-ONTO poses neither compatibility problems nor introduce inconsistencies. However, other popular ontologies such as Plant Ontology [188] and GCPs crop ontology [189] are found to be incompatible with our ontology, due to the differences in format (OBO) and ontological commitment. Nevertheless, their vocabularies were frequently consulted for standardizing common concepts and term definitions, such as the growth stages of Legume crops (being the family containing Bambara groundnut) among others.

4.2.6.3 Natural Language Annotations for UC-ONTO

As discussed earlier, a natural language definition of domain-specific terms can help in alleviating most of the ambiguities and increase the usability, interoperability of domain ontologies. In our approach, we utilize the annotation design

[3]http://www.seamless-ip.org/

pattern by collaborating with the underutilized crops domain experts for basic definition and explanation of domain-specific concepts in common language — see Fig. 4.8. Moreover, regarding the naming conventions of concepts, domain experts as well as relevant standard ontologies such as the crop ontology [189], were frequently consulted for standardization. For example, in the ontology, all crops related concepts are grouped together under the *DomainConcepts* as super class and all other concepts such as 'Area', 'TimeZone', etc. offered by imported ontologies, are composed as siblings. The *UnderutilizedCrops* class contains the subclass *BambaraGroundnut*, which dominates most of the object properties and data-type property modeling in the current version of UC-ONTO.


Annotations: BambaraGroundnut

Annotations

rdfs:label [type: rdfs:Literal]

The Bambaragroundnut class (abbbrev. BG), also Bambaranut or Bambara

rdfs:comment

This is the main class of our ontology as an example of Underutilized crop.
It has an instance BambaraGroundnut, sometimes referred as BG or BambaraGroundnutInd.

Figure 4.8: Annotations for Bambaragroundnut Concept

4.2.7 The UC-ONTO Extension with SWRL Rules

As explained earlier, the integration of OWL ontologies and LP rules provides many advantages that cannot be achieved using either OWL or rules alone. In this section, we present the implementation of SWRL rules into the underutilized crop ontology. However, as SWRL rules cannot introduce new terms into the ontology — a safety requirement for its decidability, the addition of our SWRL rules was made to conform to the DL-safety restriction. By using only those concepts already defined in the OWL ontology, the rules were added to enable more complex domain modeling and to express such conditions that are otherwise difficult or even impossible using OWL constructs. An example is shown in the use of SWRL rule to determine the current development stage of a crop based on the planting date, also called days after sowing (DAS). We use the SWRL mathematical built-ins (*swrlb:lessThanOrEqual* and *swrlb:lessThanOrEqual*) for comparison of the datatype property DAS with the number of days for each stage of development, as depicted in rule 4.1. A fragment of our SWRL rule base in shown in Fig. 4.9:

$$\begin{gathered} BambaraGroundnut(?bg) \ \wedge \ DAS(?das) \ \wedge \ GrowthStage(?gs) \wedge \\ hasAverageDAS(?bg,?d) \ \wedge \ greaterThanOrEqual(?das,30) \wedge \\ lessThanOrEqual(?das,50) \longrightarrow hasGrowthStage(?bg,"Flowering") \end{gathered} \tag{4.1}$$

This rule, which is hard to be expressed using OWL, will assert a new property (hasGrowthStage)to the Bambara groundnut instance with a datatype value of ”Flowering”.

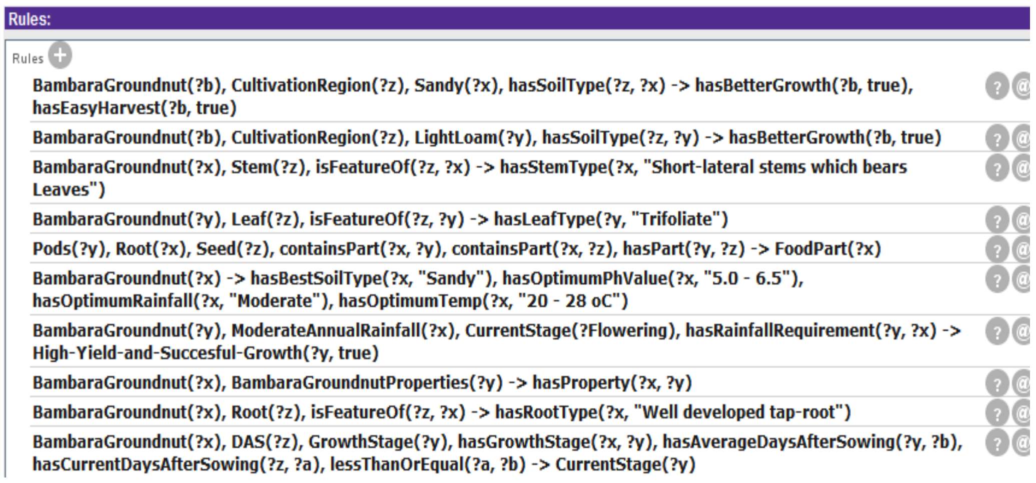
Moreover, if this growth stage needs to be added into the GrowthStage class, then the following rule can do the trich:

$$\begin{gathered} BambaraGroundnut(?bg) \ \wedge \ GrowthStage(?gs) \wedge \\ hasGrowthStage(?bg,"Flowering") \wedge swrlx:makeOWLThing("Flowering",?gs) \\ \longrightarrow \ GrowthStage("Flowering") \end{gathered} \tag{4.2}$$

The above rule 4.2 uses the SWRL existential built-in (swrlx:makeOWLThing), which can create an individual instance of the Growth Stage class anytime the hasGrowthStage(Flowering) property of the Bambara groundnut class is attained — meaning that the days after sowing of Bambaragroundnut DAS $\geqslant$ 30 and DAS $\leqslant$ 50. This also highlights the advantage of rules such that the assertion can be added pending to the firing conditions of the rule.

As shown in the rules interface depicted in Fig. 4.9, simple assertions are also possible as an alternative to class/property definitions in OWL. Example is provided in rule 4.3, where we assert a *featureOf (inverse: hasFeature)* relationship between members of *BambaraGroundnut* class and those of *Properties* class.

$$BambaraGroundnut(?bg) \ \wedge \ Properties(?p) \longrightarrow \ hasFeatures(?bg,?p) \tag{4.3}$$



BambaraGroundnut(?b), CultivationRegion(?z), Sandy(?x), hasSoilType(?z, ?x) -> hasBetterGrowth(?b, true), hasEasyHarvest(?b, true)
BambaraGroundnut(?b), CultivationRegion(?z), LightLoam(?y), hasSoilType(?z, ?y) -> hasBetterGrowth(?b, true)
BambaraGroundnut(?x), Stem(?z), isFeatureOf(?z, ?x) -> hasStemType(?x, "Short-lateral stems which bears Leaves")
BambaraGroundnut(?y), Leaf(?z), isFeatureOf(?z, ?y) -> hasLeafType(?y, "Trifoliate")
Pods(?y), Root(?x), Seed(?z), containsPart(?x, ?y), containsPart(?x, ?z), hasPart(?y, ?z) -> FoodPart(?x)
BambaraGroundnut(?x) -> hasBestSoilType(?x, "Sandy"), hasOptimumPhValue(?x, "5.0 - 6.5"), hasOptimumRainfall(?x, "Moderate"), hasOptimumTemp(?x, "20 - 28 oC")
BambaraGroundnut(?y), ModerateAnnualRainfall(?x), CurrentStage(?Flowering), hasRainfallRequirement(?y, ?x) -> High-Yield-and-Succesful-Growth(?y, true)
BambaraGroundnut(?x), BambaraGroundnutProperties(?y) -> hasProperty(?x, ?y)
BambaraGroundnut(?x), Root(?z), isFeatureOf(?z, ?x) -> hasRootType(?x, "Well developed tap-root")
BambaraGroundnut(?x), DAS(?z), GrowthStage(?y), hasGrowthStage(?x, ?y), hasAverageDaysAfterSowing(?y, ?b), hasCurrentDaysAfterSowing(?z, ?a), lessThanOrEqual(?a, ?b) -> CurrentStage(?y)

Figure 4.9: Fragment of SWRL Rules showing Assertions for UC-ONTO
Note: The Protégé rules tab uses comma instead of (∧) to represent conjunction.

The rule highlights the flexibility of the SWRL formalism and the fact that it subsumes the OWL language in terms of expressive powers — in the sense that most OWL assertions can be achieved using SWRL but not vice-versa.
We then continue to assert other rules that are not easily expressed in OWL to achieve more comprehensive domain modeling — thereby extending the expressiveness of the underutilized crops ontology. For example, rule 4.4, which asserts the specific feature to a Bambara groundnut's 'leaf' instance, is written thus:

$$BambaraGroundnut(?bg) \wedge Leaf(?l) \wedge hasFeature(?bg, ?l) \longrightarrow hasLeafType(?l, "Trifoliate") \quad (4.4)$$

The above SWRL expression can be interpreted thus; if it ascertained that 'BambaraGroundnut' class has a 'Leaf' feature, then it will be asserted that the leaf-type is "Trifoliate". Since features such as leaf are not exclusive to Bambara Groundnut, then unless the leaf individual is related to Bambara Groundnut, the leaf type 'Trifoliate', will not be immediately asserted. Such rules that are based on certain conditions being true or otherwise are hard to be expressed with OWL concept definitions. These and many more rule assertions were achieved using the SWRL formalism ensuring unbounded domain modeling with ease. We discuss the validation process of the added SWRL rules below with more results of the ontology development and example DL queries to confirm the decidability of the rule assertions.

4.3 Case Study 2: Ontology Utilization — The Onto-CropBase Semantic Search Engine

In this section, implementation of the ontology-based semantic search engine (The Onto-CropBase) is discussed. The Onto-CropBase is an ontology-driven search tool which serve as a web-based access point for the underutilized-crops ontology model. The tool utilizes the UC-ONTO — as the global ontology, extended with linked data in RDF — the local ontologies, integrated to serve as its knowledge base. As earlier presented in the design methods in Section 3.4, major discussion points are thus the functionalities offered by the tool in exploring the UC-ONTO knowledge model with the implementation details.

In its simplest form, the Onto-CropBase tool consists of a web interface, depicted in Fig. 4.10, which in the background utilizes the Java Server Pages (JSP) and *servlets* components of the web application. These components contain java codes that invoked the Jena and Pellet reasoner APIs integrated with the data model (ontologies), to accept user queries, fire the SPARQL query engine to probe the linked data models, and present search results to the user. Using the Apache Server, the Web application components are packaged and deployed as Web Archive (WAR) file, which is searchable using a specified URL.

4.3.1 Onto-CropBase Functionalities

Since the Onto-CropBase tool aims to provide an information retrieval interface for exploring an ontological knowledgebase integrated with relevant linked-data, the following are provided as its major functionalities: *(i)* provision for a keyword-based semantic search engine, *(ii)* query answering and presentation of query results, *(iii)* navigating the ontology and search results, and *(iv)* a map interface, showing relevant crop location information. The keyword-based search engine was developed as the first and single most important component of the Onto-

CropBase tool, allowing federated searches with a single user query over the knowledge bases. As location data is critical to crop-based knowledge systems especially that of the underutilized crops, a map interface is provided to display the crops location data.

Information provided by the tool is designed to be presented in a simple and straight forward fashion with minimum ambiguity. We briefly highlight here, some of the available functionalities:

4.3.1.1 Concept Search

Exploring the knowledgebase in the Onto-CropBase tool typically starts with the search space — see the home page in Fig. 4.10. The search query is matched against relevant concept definitions in the global ontology. Where a concept match is found, the RDF triples in the corresponding local ontologies are probed through their linked URIs and the associated triples are returned with a map view of the origin or farming area of the specific underutilized crop(s). However, the map ontology is searched separately according to the location data associated to the crop (where applicable), This is asserted through Google Maps' JavaScript API[4].We use the API version 3 for the Onto-CropBase map rendering.

4.3.1.2 Query Answering

User queries containing keywords entered in the search engine are compiled into SPARQL queries, as shown in the **Listing 4.1** below. The queries were designed to provide search results based on *classes, entities,* and textitmap area. The mediator, Jena-API, first loads the ontologies and the ARQ query engine uses the search keywords to generate a query plan. The results of executing the query plan generate an RDF data satisfying the query pattern, called the *Result Set*. The result sets are then passed to the *filtering object*, which returns only matching triple objects from each dataset as the final answer to the query. Ordering of the answer triples is achieved using a *binding hierarchy* generated by the query engine. This binding hierarchy reflects the query patterns and the final query output are presented based on the initial values to be resolved.

[4]https://developers.google.com/maps/documentation/javascript/

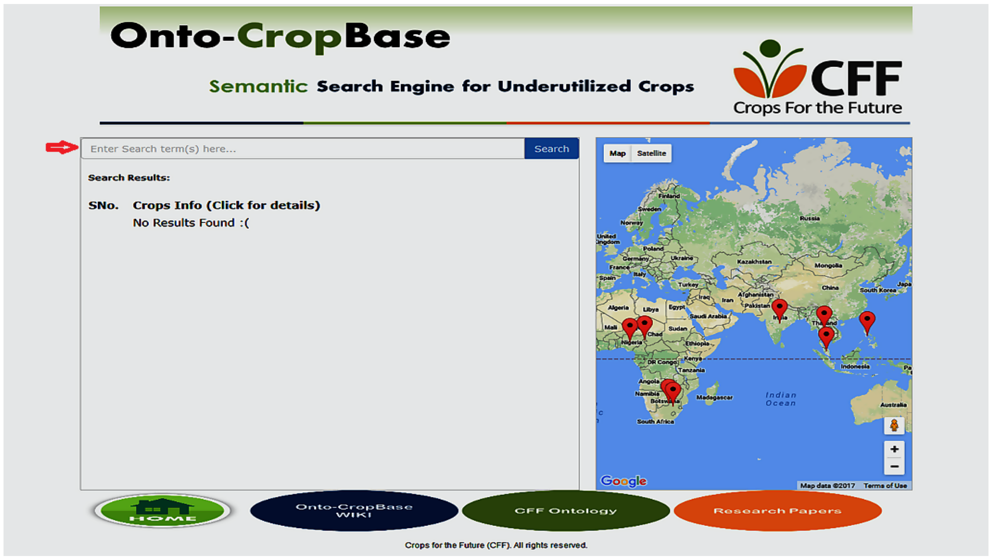

Onto-CropBase
Semantic Search Engine for Underutilized Crops
CFF Crops For the Future
Enter Search term(s) here...
Search
Search Results:
SNo. Crops Info (Click for details)
No Results Found :(
HOME
Onto-CropBase WIKI
CFF Ontology
Research Papers
Crops for the Future (CFF). All rights reserved.

Figure 4.10: The Onto-CropBase Home Page

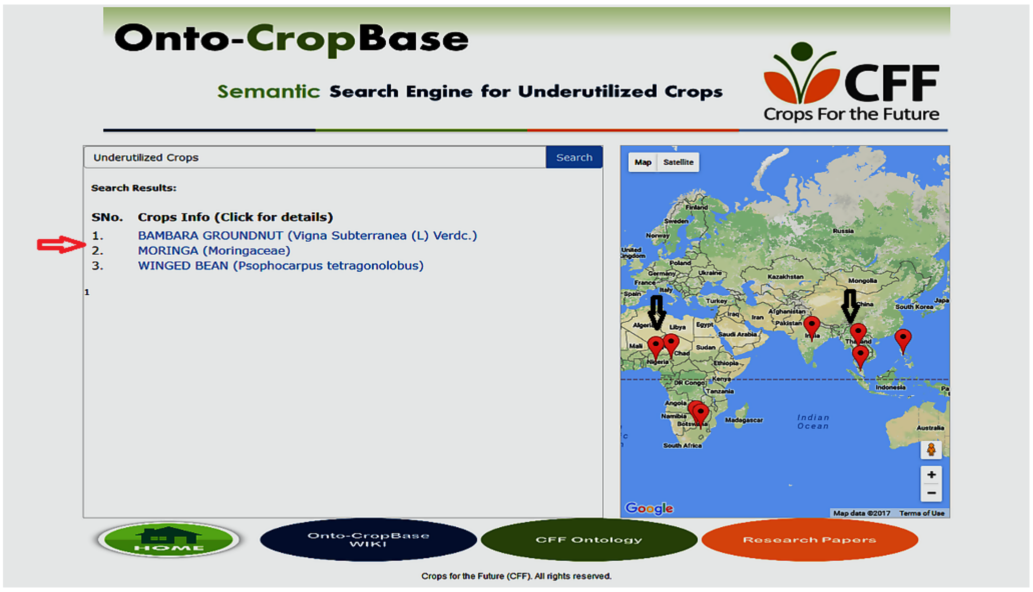
```
...
"PREFIX uconto: <http://www.nottingham.edu.my/ontologies
/2014/Ontology-uco#>" +
"PREFIX ucnutrition: <http://.../ontologies/2015/nut#>" +
"PREFIX ucnames: <http://.../ontologies/2015/Naming#>" +
"PREFIX agronomy: <http://.../ontologies/2015/agrono#>" +
SELECT distinct ?subject ?object " +
WHERE {" + " ?subClass rdfs:subClassOf uconto:" +
className + " . " + " " + " } " ;
```

Listing 4.1: A SPARQL example query to display all classes

An example SPARQL query to request all classes in the global OWL ontology is shown in **Listing 4.1** above. However, in order to query all the instance assertions matching the above subject from any of the local ontologies (e.g. the *agronomy* local ontology), the example SPARQL query can be written thus:

```
...
SELECT DISTINCT ?subject ?property ?object" + "
WHERE {" + " ?subject a agronomy:" + "queryString
"+" . " + " ?subject ?property ?object . " +"}
ORDER BY ?subject" ;
```

The results of the above queries are presented as set of navigational links grouped under the titles of their source ontologies — as shown in Fig. 4.11. Clicking on a particular dataset title shows the detailed information as *subject* and *object* pairs — as shown in Fig. 4.12.



Search Results:

SNo.	Crops Info (Click for details)
1.	BAMBARA GROUNDNUT (Vigna Subterranea (L) Verdc.)
2.	MORINGA (Moringaceae)
3.	WINGED BEAN (Psophocarpus tetragonolobus)

1

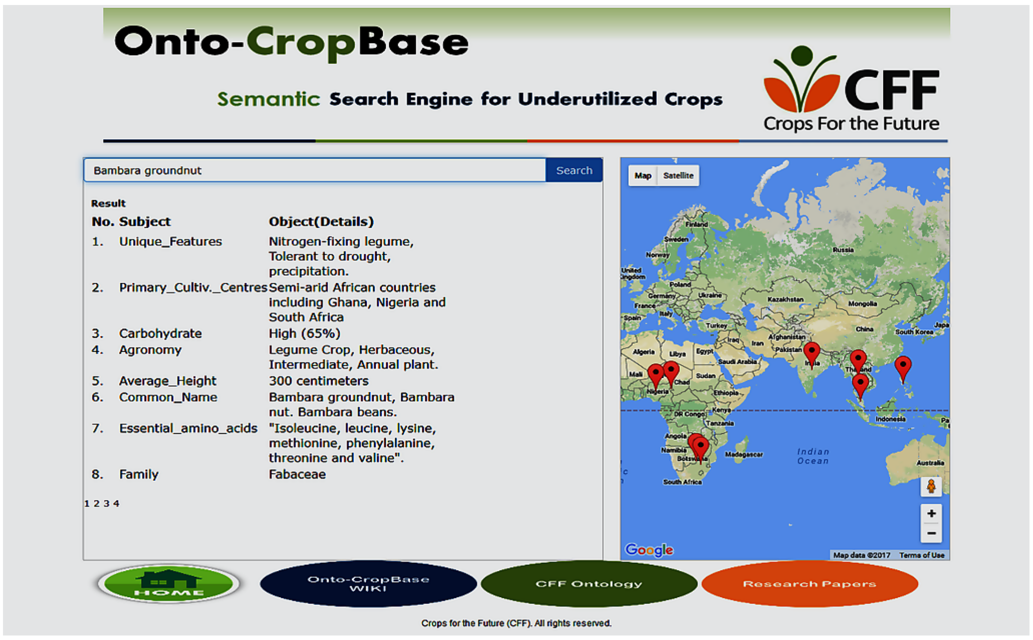
Figure 4.11: Onto-CropBase showing concept search *(Underutilized Crops)* and results as link titles.

4.3.1.3 Map Interface

Consistent with the location data imported from the FAO's geopolitical ontology [190], the Onto-CropBase tool is also designed to show the underutilized crops location data (where applicable and available) using the *Crop Origin* or *Cultivation Region* assertions. As explained earlier, Google Map is used as a base map for the location-based data and the information is extracted from the named location's elements — the 'east-west bound longitude' and 'north-south bound latitude'. The map location data can help users to get a clear picture of crops' origin, cultivation regions, and locations where similar crops can be cultivated, among other information.

4.3.1.4 Paging

As shown in the figures. 4.12 and 4.13, the Onto-cropBase search results are designed to be presented in a series of numbered rows showing the RDF result sets in the form of subject-object pair and with a maximum of 10 rows per page. Where the resulting information exceeds ten rows, a new page is automatically created



Result

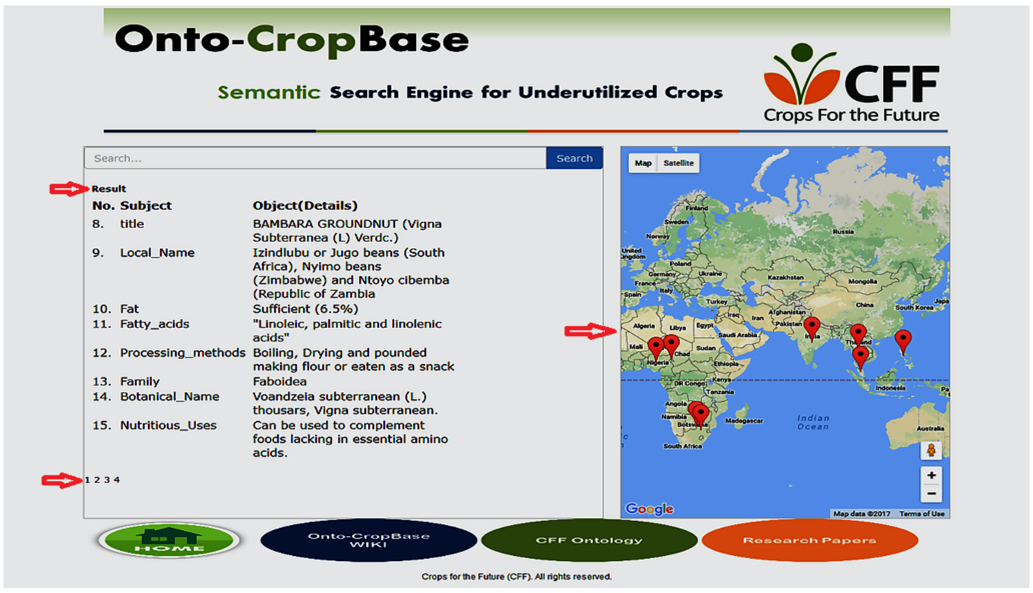
No.	Subject	Object(Details)
1.	Unique_Features	Nitrogen-fixing legume, Tolerant to drought, precipitation.
2.	Primary_Cultiv._Centres	Semi-arid African countries including Ghana, Nigeria and South Africa
3.	Carbohydrate	High (65%)
4.	Agronomy	Legume Crop, Herbaceous, Intermediate, Annual plant.
5.	Average_Height	300 centimeters
6.	Common_Name	Bambara groundnut, Bambara nut. Bambara beans.
7.	Essential_amino_acids	"Isoleucine, leucine, lysine, methionine, phenylalanine, threonine and valine".
8.	Family	Fabaceae

1 2 3 4

Figure 4.12: The Onto-CropBase tool showing title result *(Bambara groundnut)* and detailed facts.

with *page counts* similar to those found in the commercial search engines. Similarly, user can navigate these pages by clicking on the desired page number at the bottom of the search results. This allows the map interface to stay in focus throughout the navigation process giving users continuous access to the location data.

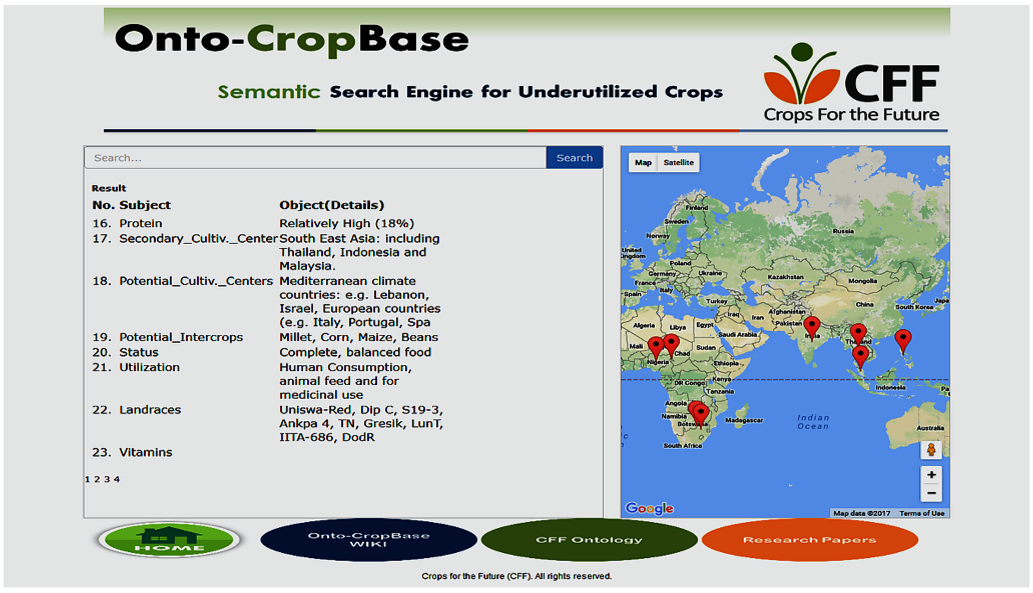
Further discussion of the Onto-CropBase tool and its evaluation is presented in Section 5.3. The evaluation highlights the performance measurement of the semantic search functionality. This is followed by the domain experts' evaluation of the tool and its comparison with relevant tools in Sections 5.3.1 and 5.3.2 respectively.



Result

No.	Subject	Object(Details)
8.	title	BAMBARA GROUNDNUT (Vigna Subterranea (L) Verdc.)
9.	Local_Name	Izindlubu or Jugo beans (South Africa), Nyimo beans (Zimbabwe) and Ntoyo cibemba (Republic of Zambia
10.	Fat	Sufficient (6.5%)
11.	Fatty_acids	"Linoleic, palmitic and linolenic acids"
12.	Processing_methods	Boiling, Drying and pounded making flour or eaten as a snack
13.	Family	Faboidea
14.	Botanical_Name	Voandzeia subterranean (L.) thousars, Vigna subterranean.
15.	Nutritious_Uses	Can be used to complement foods lacking in essential amino acids.

1 2 3 4



Result

No.	Subject	Object(Details)
16.	Protein	Relatively High (18%)
17.	Secondary_Cultiv._Center	South East Asia: including Thailand, Indonesia and Malaysia.
18.	Potential_Cultiv._Centers	Mediterranean climate countries: e.g. Lebanon, Israel, European countries (e.g. Italy, Portugal, Spa
19.	Potential_Intercrops	Millet, Corn, Maize, Beans
20.	Status	Complete, balanced food
21.	Utilization	Human Consumption, animal feed and for medicinal use
22.	Landraces	Uniswa-Red, Dip C, S19-3, Ankpa 4, TN, Gresik, LunT, IITA-686, DodR
23.	Vitamins	

1 2 3 4

Figure 4.13: The Onto-CropBase tool showing more result pages: facts No. 8-15 (top) and facts No. 16-23(bottom).

4.4 Case Study 3: Ontology Language Extension — The Fuzzy-Temporal Extension of Semantic Web Rule Language (FT-SWRL)

The FT-SWRL Model While fuzzy temporal knowledge modeling has been around since the early days of AI, the semantic web domain has seen fewer advancements in the temporal uncertainty modeling. Relevant research efforts have focused mostly on the uncertainty management or the representation of the temporal data as a domain. In FT-SWRL extension, we go beyond the simple structured time data in ontologies to provide additional syntax and semantics that enable the representation of vague temporal facts in the semantic web. The new semantic web rule language extension can handle the modeling of uncertainties that exists in the time domain, defined in the SWRL-FT ontology and with the possibility of reasoning and inference through the SWRL fuzzy temporal built-ins. Both presented in this section and followed with example FT-SWRL rules.

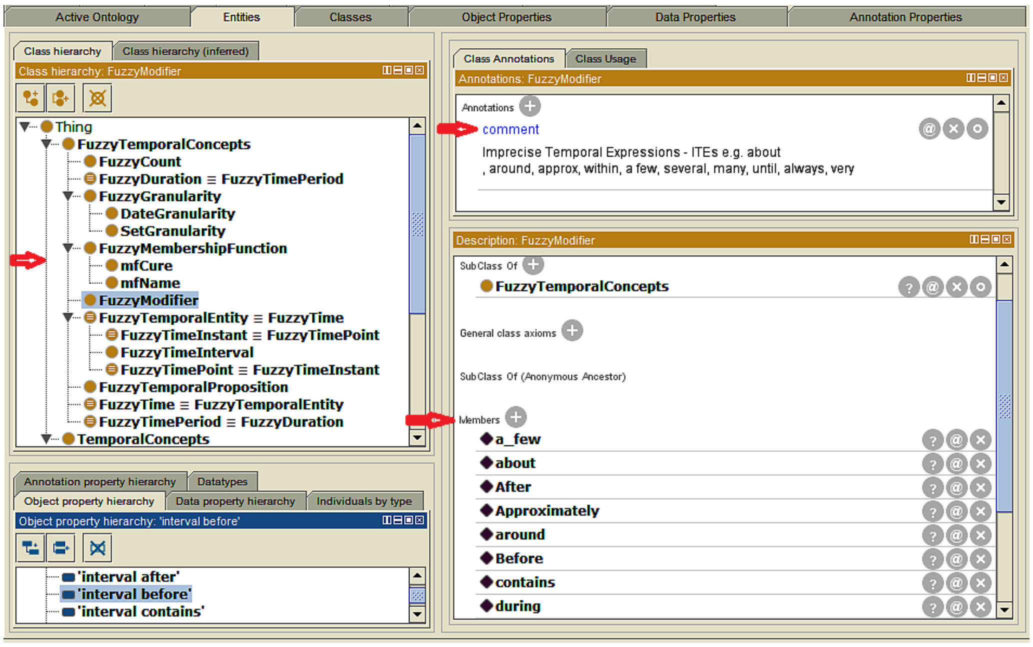
In this section, we present a fuzzy temporal extension to the semantic web rule language (FT-SWRL), which combines fuzzy theories based on the valid-time temporal model, to provide a standard approach for modeling imprecise temporal domain knowledge. FT-SWRL[5] introduces a fuzzy temporal model for the semantic web, which consists of two important components: (i) a SWRL fuzzy temporal ontology (SWRL-FTO), which formally specify the linguistic terminologies and variables of the FT-SWRL model and (ii) a set of fuzzy temporal built-ins for defining their semantics. The fuzzification process of the fuzzy temporal built-ins is presented in Section 4.4.3, with example FT-SWRL rules demonstrating their possible usage in modeling imprecise temporal expressions.

4.4.1 The SWRL Fuzzy Temporal Ontology

As shown in Fig. 4.14 below, in order to preserve the modular feature of the original temporal model, the SWRL Fuzzy-temporal ontology (SWRL-FTO) begins with the *FuzzyTemporalProposition* class, which is the class for all fuzzy-timed

[5]Manuscript titled: *FT-SWRL — A Fuzzy Temporal Extension of the Semantic Web Rule Language.* is submitted to the International Journal of Approximate Reasoning, Elsevier, March, 2017.

events i.e. events associated with imprecise temporal expressions. This is defined as a sibling of the *temporal:ExtendedProposition* class — designed to represent entities or propositions that extend over time and with the benefit of separating the temporal ontology from the main ontology for consistency. Similarly, the *FuzzyTemporalProposition* class will allow introducing consistent fuzzy model without interfering with either the main or temporal ontology. Hence it serves as the range of all the fuzzy temporal built-in expressions defined in the ontology. The *FuzzyTemporalPropositionclass* has three object properties: *hasFuzzyTime* — with a range over the FuzzyTime class, the *hasFuzzyDuration* — with a range over the FuzzyDuration class and the *hasModifier* property — with a range over the FuzzyModifier class. See Fig. 4.15 for the objects and datatype property assertions of the FT-SWRL ontology.



Imprecise Temporal Expressions - ITEs e.g. about
, around, approx, within, a few, several, many, until, always, very

Figure 4.14: FT-SWRL Ontology fragment showing Entities Implementation in Protege ontology editor

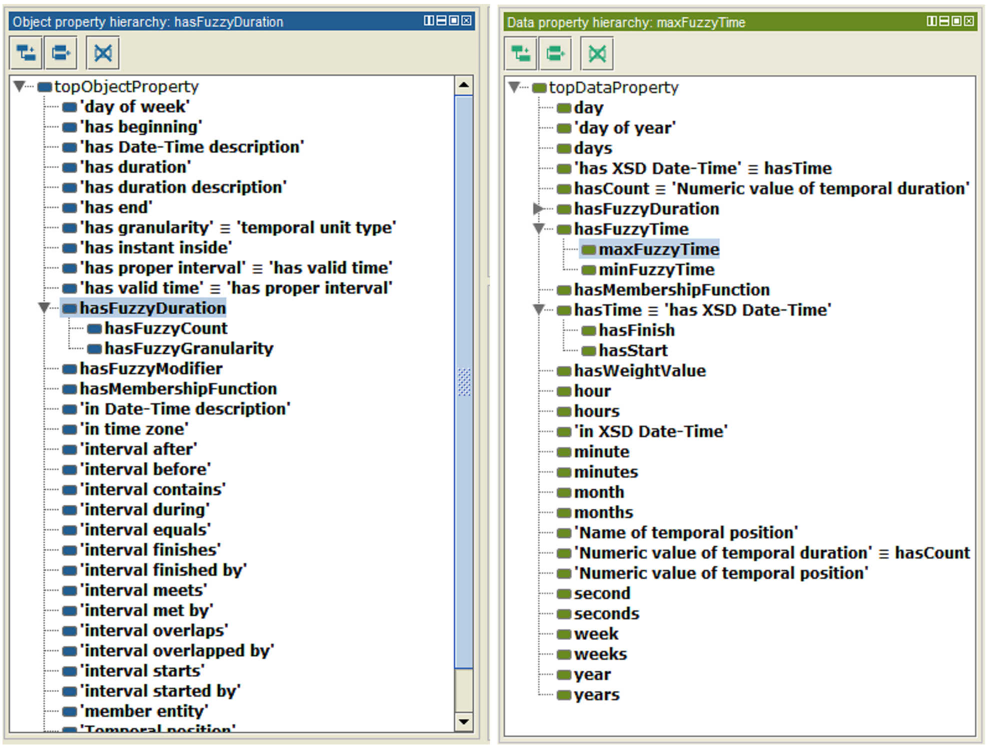
The *FuzzyTime* class represents the fuzzy time values of the fuzzy temporal propositions based on the evaluated fuzzy modifiers or ITEs. A FuzzyTime can be either a *FuzzyTimeInstant* or a *FuzzyTimePeriod* — with a range over the *xsd:dateTime* class. The two describe instantaneous as well as period-based events. Where the time of occurrence of an instantaneous or single timed event is imprecise, then we use the FuzzyTimeInstant (sameAs: FuzzyTimePoint) and FuzzyTimePeriod is used where the event happens over two imprecisely-timed points (period). *FuzzyDuration* class helps to represent such ITEs containing durations not defined as dateTime data, e.g. 'this weekend', 'within 3 weeks', 'after several hours' etc. As such, it has two subclasses as *FuzzyCounts* and *FuzzyGranularity*.

FuzzyTemporalPropositions as sets of temporal facts are categorized as *SetGranularities* (e.g. weekly, perHour, daily, etc.) — a subclass of the FuzzyGranularity class. Whereas, those that basically compare the object propositions with a current date (e.g. past 3 weeks, since last year, 2 weeks ago, etc.) are sub-classed as *DateGranularities*.

TheFuzzyModifier class represents the fuzziness of specific fuzzy temporal information or ITEs contained in the fuzzy temporal propositions. It has two object properties: *hasWeightedValue and hasMembershipFunction* which ranges over the *WeightValues* class and *MembershipFunction* class respectively. Corresponding values of the membership function and weight values are assigned to each ITE as functional datatype properties. It represents fuzzy functions for such ITEs as *about, around, approx, within, a few, several, many, until, always, very, etc.* — and are defined as a set of SWRL built-ins that can be used for temporal operations on the entities defined by the fuzzy temporal ontology. In the following sub-section, we briefly highlight these built-ins and the intuitions behind them.

4.4.2 SWRL-FT Built-ins: Semantics Definition

The SWRL fuzzy temporal built-ins are defined to allow temporal operations on imprecise temporal information during domain knowledge modeling. By defining selected ITEs as part of the SWRL built-in sets, FT-SWRL extends SWRL formalism, and equally the OWL language, with constructs to implement fuzzy temporal semantics within ontologies. This will allow the combination OWL/SWRL to handle fuzzy temporal knowledge for the first-time, without relying on external frameworks for reasoning over imprecise temporal data. Using the SWRL-FT



```
Object property hierarchy: hasFuzzyDuration
topObjectProperty
    'day of week'
    'has beginning'
    'has Date-Time description'
    'has duration'
    'has duration description'
    'has end'
    'has granularity' ≡ 'temporal unit type'
    'has instant inside'
    'has proper interval' ≡ 'has valid time'
    'has valid time' ≡ 'has proper interval'
    hasFuzzyDuration
        hasFuzzyCount
        hasFuzzyGranularity
    hasFuzzyModifier
    hasMembershipFunction
    'in Date-Time description'
    'in time zone'
    'interval after'
    'interval before'
    'interval contains'
    'interval during'
    'interval equals'
    'interval finishes'
    'interval finished by'
    'interval meets'
    'interval met by'
    'interval overlaps'
    'interval overlapped by'
    'interval starts'
    'interval started by'
    'member entity'
    'Temporal position'
```

```
Data property hierarchy: maxFuzzyTime
topDataProperty
    day
    'day of year'
    days
    'has XSD Date-Time' ≡ hasTime
    hasCount ≡ 'Numeric value of temporal duration'
    hasFuzzyDuration
    hasFuzzyTime
        maxFuzzyTime
        minFuzzyTime
    hasMembershipFunction
    hasTime ≡ 'has XSD Date-Time'
        hasFinish
        hasStart
    hasWeightValue
    hour
    hours
    'in XSD Date-Time'
    minute
    minutes
    month
    months
    'Name of temporal position'
    'Numeric value of temporal duration' ≡ hasCount
    'Numeric value of temporal position'
    second
    seconds
    week
    weeks
    year
    years
```

Figure 4.15: FTSWRL Ontology Objects Properties Implementation in Protege ontology editor

built-ins, imprecise temporal data can be encoded following the SWRL-FT ontology model and processed based on the valid-time temporal model for efficient knowledge representation and retrieval. Following the fuzzy temporal entities classification in Section 3.5.3, the following terminologies were defined as the first set of the SWRL fuzzy-temporal built-ins:

4.4.2.1 Fuzzy Duration Built-ins

The Fuzzy Duration built-ins were defined to operate on imprecise temporal durations. In this context, *FuzzyDuration* is considered as a temporal expression containing fuzzy Count at a specified base granularity. Unlike the *fuzzytemporal:FuzzyTimePeriod* which can be specified by two fuzzy times instants (*fuzzytemporal:FuzzyTimeInstants*), the fuzzy duration involves expressions such as ”few weeks”, ”several hours” etc. where the first part (few, several) are the fuzzy counts and the latter (weeks, hours) are the base granulaarity of the receiving proposition. As such, the FuzzyDuration built-in method requires a *FuzzyCount* and *FuzzyGranularity* as its arguments.

Other operators associated with the FuzzyDuration include the *fuzzyDurationLessThan, fuzzyDurationEqualsTo*, and *fuzzyDurationGreaterThan* built-ins: As a sub-built-in of the *temporal:Duration* predicate, the FuzzyDuration built-ins inherently includes these operators for comparable inference among consistent FuzzyDuration instants having bounded arguments. Moreover, inverses of these built-ins may well be considered for completeness.

4.4.2.2 Fuzzy Count Built-ins

These built-ins are designed to implement the imprecise counts on temporal data. Example cases include: 'several, many, long-time, this, next, last, cycles, times, twice, etc'. In their basic form, usage of these built-ins requires that they take the *FuzzyGranularity* as argument and after applying the relevant fuzzy operations defined by their semantics, returns a multiplier or comparison count of the granularity.

4.4.2.3 Fuzzy Granularity Built-ins

These built-ins are designed to implement the imprecise granularities for an xsd:dateTime class. Example cases include: 'weeks, weekend, fortnight, quarter, noon, etc'.They extend the original SWRL Date, Time and Duration built-ins[6].

4.4.2.4 Fuzzy Set Granularity Built-ins

These built-ins are designed to implement the imprecise set granularities for the xsd:dateTime class. They extend the *FuzzyGranularity* Built-ins to denote set-wise granularities for recurring events. Example cases include: 'Yearly, Monthly, Weekly, daily, hourly, perMinute, perSeconds, perHour, perWeek, perYear, etc'.

[6]http://www.daml.org/2004/04/swrl/builtins.html#8.5

4.4.2.5 Fuzzy Date Granularity Built-ins

These built-ins are designed to implement the imprecise date granularities for annotating the xsd:dateTime class. They extend the *FuzzyGranularity* Built-ins for comparison operations between the current time and the transaction time (temporal proposition object). Example cases include: 'the past, present, currently, recently, nowadays, ago, since, lately, earlier, etc'. Hence it requires two optional arguments; the event time and the current time — in xsd:dateTime instant or duration.

4.4.2.6 Imprecise Temporal Approximation Built-ins

As their name implies, these built-ins are designed to implement the vague temporal approximations. Example case includes: 'about, around, approx, within, a few, until, before, very, after, etc'). The built-ins take argument representing the *FuzzyTime* of the temporal fact to apply the relevant fuzzy temporal operations on them. The operation also requires two more arguments representing the *Count* and *FuzzyGranularity* as follows: *fuzzytemporal:about*(?FuzzyTime, ?Count, ?fuzzyGranularity).

Where the Count or Granularity arguments are missing, the default count = 1 and a base granularity of the temporal fact will be used. Nevertheless, the built-ins can be further expanded with more predicates as far as the tractability and semantics of the language can allow. Moreover, inverses of these built-ins (where applicable) can be considered as future extensions.

4.4.3 Reasoning Paradigm for FT-SWRL Ontology Model

The original SWRL temporal extension basically defines temporal interval operations as built-ins and neither contain inference rules for time expressions nor translation rules from natural language expression to times. However, the FT-SWRL proposal can lead the way in providing a consistent model for defining fuzzy temporal inference rules for (some of) the commonly encountered imprecise temporal expressions (ITEs). To this end, we propose the fuzzification of the interval-based temporal logic in order to achieve a complete OWL-based reasoning for the fuzzy temporal built-ins. This is particularly important as it can enable the OWL/SWRL combination to enforce temporal semantics as well as handling vague temporal knowledge. Moreover, with the SWRL Query language (SQWRL) able to handle such temporal reasoning, querying temporal information from OWL ontologies will be highly improved.

For efficient representation and reasoning about the fuzzy temporal information encoded in FT-SWRL rules, we define the fuzzy times of the ITEs using carefully selected membership functions superimposed on their interval-based temporal definitions. The fuzzy membership functions were selected based on their correspondences to the imprecise temporal expressions using the weighted value (w) as the gauge of the temporal information as it approached the true value (T). However, we give some formularized restrictions to these weighted values within which the statements are found to be a close-enough representation of the temporal information. We focused on the frequently used ITEs found in the crops domain knowledge narratives — as earlier presented in 3.5.1.

4.4.3.1 SWRL-FT Built-ins Fuzzification

Following the linguistic terminologies and variables definition in the SWRL-FT ontology and Built-ins, we describe the *Fuzzification* [182, 148] of the SWRL-FT built-ins that will serve as translation rules during fuzzy temporal reasoning. In what follows, we chose suitable membership functions for some selected imprecise temporal expressions and demonstrate how we can generate their corresponding fuzzy values based on the membership functions:

Fuzzy temporal term	**’about’ (T)**, where: T = fuzzy duration interval.
sameAs	Around (T), approximately (T), nearly (T).
Super Class	FuzzyModifier (annotation: Approximation-ITEs)
Properties	hasWeightDegree, hasFuzzyTime, hasModifierFunction
Required arguments	WeightDegree, FuzzyTime
Fuzzy MF	Gaussian (Gaussmf)

Table 4.3: Fuzzy temporal term *about*

As shown in Fig. 4.16: Membership Function for 'about (T)' fuzzy temporal expression, we use the Gaussian membership function (gaussmf) to define the fuzziness as a set over the about temporal expression as the universal set of discourse. Based on the semantic definition of the 'temporal approximation' keywords such as about, around, etc, the use of such imprecise times is usually where the narrator refers to time units that are close to the exact time (of an event or process) and when such assumed times are up to a complete granularity. For example, the statement "Bambara beans germination time is *around/about/approximately* 1 to 2 weeks". Here, the granularity (of weeks) is used to show that an event — the germination of Bambara beans, may happen either in the first or the second week. This can basically be represented by the Gaussian MF (gaussmf), with the temporal value T = 7 days or 1st week as the peak-value (weight degree w = 1). The peak time can then be approached from either direction with increasing certainty (as 'w' tends to 1) until the actual time (truthTime T) of the event is reached. As such, the required information needed to model the approximation keyword will simply be 'the weighted degree of truth' of the information source.

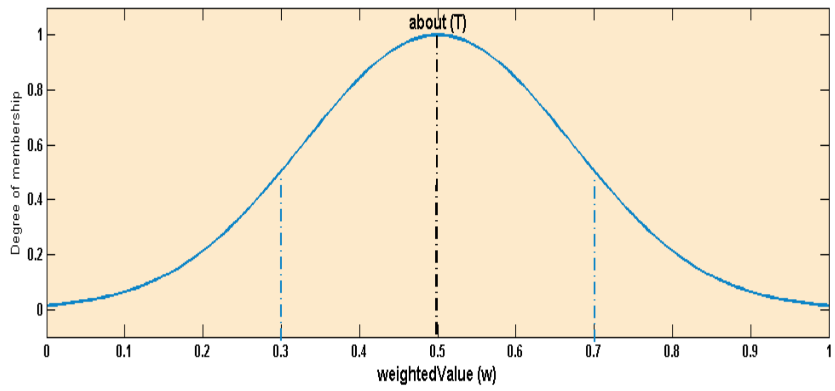


Figure 4.16

The fuzzy time (f_T) for each ITE can then be calculated based on the assigned membership distribution function and weighted value. For the approximation ITEs, we calculate the minimum fuzzy time ($\min f_T$) and maximum fuzzy times ($\max f_T$) as the border-points for the resulting fuzzy temporal membership function as follows:

$$\min f_T = [T - (1 - w) * T/2] \tag{4.5}$$

$$\max f_T = [T + (1 - w) * T/2] \tag{4.6}$$

$$\min f_T < f_T < \max f_T \tag{4.7}$$

Where:

- T is the valid time unit in the temporal expression.
- w is the weighted truth degree of the expression (or information source).
- $(1 - w) * T/2$ is the distribution function for determining the fuzzy time based on the 'w' values.
- f_T is the fuzzy time based on the weightedValue(w).
- $\min f_T$ is the lower-bound fuzzy time for the *about (T)* expression.
- $\max f_T$ is the upper-bound fuzzy time for the *about (T)* expression.
- Note: the intuition in the distribution function is that the higher the degree of certainty (w), the closer the fuzzy times (f_{T--}) / f_{T++})) becomes to the actual valid time (T) on both sides.

With the above equations (4.5 - 4.7), we simplified the fuzzification of the ITE by calculating the possible minimum and maximum valid times for the expression. We use T/2 as a simplified distribution of the fuzzy variable (w) for the 'about (T)' expression, which implies that the fuzzy time can take values from $T - T/2$ for the possible times before T, to the $T + T/2$ possible timestamps after T. This is found to be consistent with our explanation that the about/around/approximately ITEs are commonly used to describe imprecise times (or events) that are within 1 or 2 granularities to the expected or precise time.

Example: Consider the expression; "Flowering time of Bambara nut is around 30 days from the date after sowing" Therefore, the parameters are: T = 30 days, suggested values from the 'about MF' for w = (0.3 0.7). Now assuming w = 0.4, then:

$$\min f_T = [30 - (1 - 0.4) * 30/2] = 21\ days] \quad \Rightarrow very\ early \tag{4.8}$$

and

$$\max f_T = [30 + (1 - 0.4) * 30/2] = 39\ days] \quad \Rightarrow very\ late \tag{4.9}$$

The resulting parameters can be easily interpreted thus: *"The flowering is early if it occurs before 30 days and after 21 days. It is late if it occurs after 30 days but before the 39th day* — thereby enabling the assertions of fuzzy terms *late* and *early* in to the knowledge base. Moreover, as the keywords, 'before' and 'after' are already defined as part of Allen's interval algebra [157], therefore reasoning operations with other relevant data (e.g. other flowering times) can easily generate a consistent temporal network that can infer additional knowledge. Moreover, this approach, as explained earlier, can be easily applied on existing temporal ontologies by introducing the temporal fuzzification (through the ITE built-ins from FT-SWRL rules) to generate the available fuzzy times (f_T). Such modeling scenario help to confirm the earlier assertion that FT-SWRL will not only allow managing fuzzy temporal information in OWL ontologies but also help to improve the utilization of existing temporal operators.

Using similar approach, we fuzzify other relevant ITEs such as 'few (T)', 'within (T)', 'before (T)', and 'after (T)' as shown below. These ITEs were selected as the first set of SWRL fuzzy temporal built-ins as they are the most frequent expressions (based on surveys presented in [191]) found in domain knowledge descriptions and natural language processors.

The 'within (T)' Built-in: From the previous example, if the statement reads: "Bambara beans germinate *within* 2 weeks from the date of sowing" it can be seen that the 'within' keyword is usually employed to express the maximum possible times that an event happens. Here we may use the '2 weeks' as the peak value. We use the Trapezoidal membership function (trapmf see Fig. 4.17) to represent the progression of the fuzzy time as the weight-value (w) increases until the flat top — where the valid-time (T) may be reached (i.e. $w = 1$). However, the sharp drop

Fuzzy temporal term	**’Within’ (T)**, where: T = fuzzy duration interval or granularity (e.g. done within a week).
sameAs	in less than (T), in under (T), at most (T), in no more than (T), etc.
Super Class	FuzzyModifier (annotation: Time closure operator)
Properties	hasWeightDegree, hasFuzzyTime, hasModifierFunction
Required arguments	WeightDegree, FuzzyTime
Fuzzy MF	Trapezoidal (trapmf)

Table 4.4: Fuzzy temporal term *within*

of the trapezoidal space function at the right-hand side corresponds to the small possible increment above the valid time (T_{++}). This follows the semantics of the ’within’ operator where a small addition to a transaction time will still be valid e.g. ’15 to 17 days’ may still be referred as within 2 weeks in a fuzzified knowledge base (FKB). Note, however, a triangular membership function (trimf) can also be used, for simplicity, to represent the fuzzy space of the ’within’ operation or where the valid time is a fuzzy instant time.

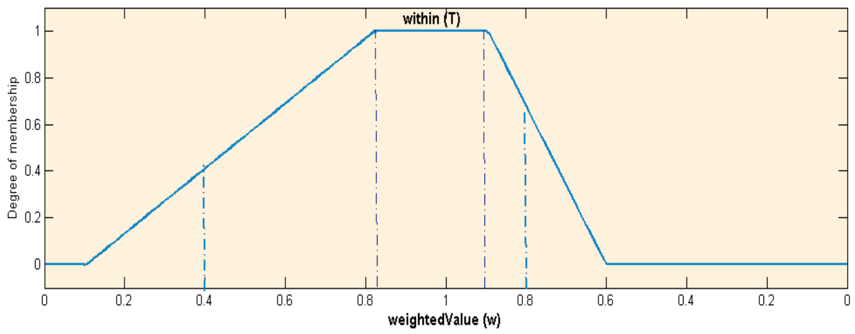


Figure 4.17: Membership Function for ’within (T)’ Fuzzy Temporal Expression

For the ’within’ built-in, we calculate the minimum fuzzy time ($\min f_T$), the left-hand side of the trapezoid and the maximum fuzzy times ($\max f_T$) at the right, as the border points for the resulting fuzzy temporal value as follows:

$$\min f_T = [T - (1 - w) * T] \tag{4.10}$$

$$\max f_T = [T + (1 - w) * T] \tag{4.11}$$

$$\min f_T < f_T < \max f_T \tag{4.12}$$

Where: $(1-w)*T$ is the distribution function for determining the fuzzy times based on the 'w' values. Note: the intuition in the distribution function is that the higher the degree of certainty (w), the closer the fuzzy times (f_{T--}) / f_{T++})) becomes to the actual valid time (T) on both sides.

Fuzzy temporal term	**'Few (T)',**, where: T = granularity (e.g. completed in 'a few' days).
sameAs	a few (T), a little (T), more or less (T), etc.
Super Class	FuzzyModifier
Properties	hasWeightDegree, hasFuzzyTime, hasModifierFunction
Required arguments	FuzzyTime, WeightDegree
Fuzzy MF	Bell membership function (gbellmf)

Table 4.5: Fuzzy temporal term *few*

The 'few (T)' Built-in: From the previous example, if the statement reads:"The Bambara beans will germinate in *few weeks* if moderate rainfall continues", shows the use of the 'few' operator to express a flexible time increment without any specified amount or granularity. However, it usually represents small changes in time, which can sometimes be negligible. As discussed in [191], the use of this ITE is common in natural language narratives when the imprecise time referred to is very close to the actual time. We use the Gaussian bell membership function (gbellmf see Fig. 4.18) — which has gentle curves near the peak value, to represent the small progression of the fuzzy time as the weight value inchess towards the flat top (w = 1).

For the *few (T)* built-in, we calculate the minimum fuzzy time ($\min f_T$), the left-hand side of the bell and the maximum fuzzy times ($\max f_T$) at the right, as border points for the resulting fuzzy temporal value as follows:

$$\min f_T = [T - (1 - w) * 0.75T] \tag{4.13}$$

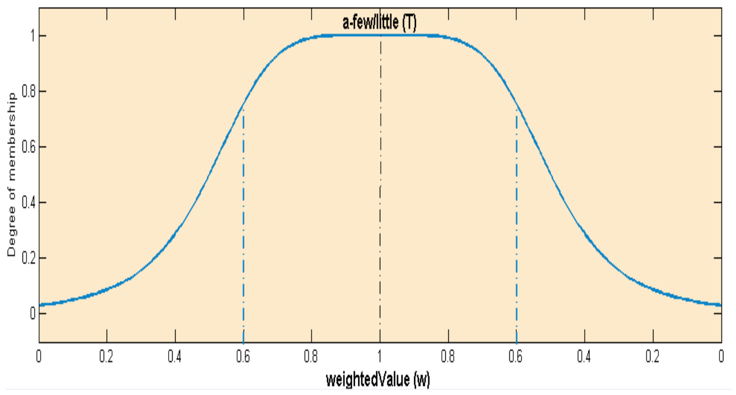


Figure 4.18: Membership Function for 'a-few (T)' Fuzzy Temporal Expression

$$\max f_T = [T + (1 - w) * 0.75T] \quad (4.14)$$

Where: $(1 - w) * 0.75T$ the distribution function for determining the fuzzy times based on the w values. Similarly, as the degree of certainty (w) tends to 1, the fuzzy times (f_{T--}) / f_{T++})) tends to the actual valid time (T) from both sides.

Fuzzy temporal term	**'before (T)',**, where: T = date, time or granularity (e.g. happens before 3pm/7 days).
sameAs	until, earlier than, previously, prior to, at most, etc.
Super Class	FuzzyModifier
Properties	hasWeightDegree, hasFuzzyTime, hasModifierFunction
Required arguments	FuzzyTime, WeightDegree (w = 0.6 1)
Fuzzy MF	S-membership function (smf)

Table 4.6: Fuzzy temporal term *before*

The 'before (T)' Built-in: Consider the statement: "Bambara beans usually germinate *before* 30 from the date of sowing. This shows the use of 'before' ITE to express that germination do or will take place before 30 days (the peak period) after planting. However, it is not clear how close or far away the germination will be from the specified time. We use the S-membership function (smf — see Fig. 4.19) to represent the progression of the fuzzy time with the increasing degree of truth from the bottom to the top of the S-function — where the fuzzy time may be equal to the valid time (T) i.e. w = 1. The continuous flat after the curve shows that we are not interested in times after the actual valid time (T_{++}). Hence the fuzziness is only on the left-hand side of the valid time to express (with some certainty) how close or far away the calculated fuzzy time is from the valid time. Which follows the semantics of the 'before' keyword — used to safely express possible times prior to a known valid time. For example, 29, 20 or even 2 days may still be referred as 'before a month' in a fuzzy knowledge base based on the degree of certainty (w).

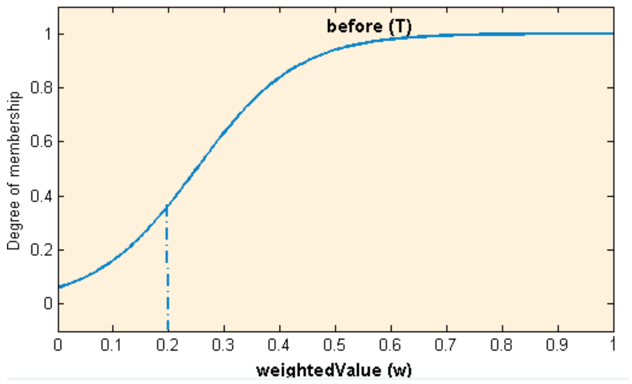


Figure 4.19: Membership Function for 'before (T)' Fuzzy Temporal Expression

Similarly, we calculate the ($\min f_T$) and ($\max f_T$) as border points for the resulting fuzzy temporal value as follows:

$$\min f_T = [T - (1 - w) * T/2] \tag{4.15}$$

$$\max f_T = T \tag{4.16}$$

Note that as the S-function is designed here as the left-half of the Gaussian function, the minimum fuzzy times and calculated parameters are the same as the 'about' built-in. The exception being that 'before' built-in has a maximum fuzzy time of T as the degree of certainty (w) tends to 1. Hence, the consistency is preserved as $f_T \leqslant T$.

The 'after (T)' Built-in: Consider the statement: "Bambara bean plant begins flowering *after* 30 days from the date of sowing" shows the use of the 'after' as

Fuzzy temporal term	**'after (T)',**, where: T = date, time or granularity (e.g. arrive shortly 'after' 13:00 hours).
sameAs	Later than (T), afterwards (T), subsequent to (T), etc.
Super Class	FuzzyModifier
Properties	hasWeightDegree, hasFuzzyTime, hasModifierFunction
Required arguments	FuzzyTime, WeightDegree (w = 0.6 1)
Fuzzy MF	Z-membership function (zmf)

Table 4.7: Fuzzy temporal term *after*

ITE to express that flowering happens after a month (the peak period). However, it cannot be ascertained how close or far away the flowering may start from the '30 days' after planting. In this case, we use the z-membership function (zmf see Fig. 4.20) to represent the regression of the fuzzy time with the decreasing degree of truth from the top of the curve — where the suggested valid time is known (w = 1). The continuous flat line before the curve shows that we are not interested in the time before the actual valid time. Hence the fuzziness is only on the right-hand side of the valid time (T_{++}). This also follows the natural semantics of the 'after' temporal expression, where it used to safely express possible transaction times that follows a known valid time. For example, 31, 32 or even 1000 days may still be referred to as 'after a month' in a fuzzy knowledge base. However, the built-in uses the weight value (w) as determining factor to safely express the possible times.

Hence, for the 'after' built-in operator, which is basically the opposite of 'before', we represent the $(\min f_T)$ and $(\max f_T)$ as border points for the resulting fuzzy temporal value as follows:

$$\min f_T = T \tag{4.17}$$

$$\max f_T = [T - (1 - w) * T/2] \tag{4.18}$$

Similarly, the maximum fuzzy time and other parameters are the same as the right-hand-side of the 'about' built-in with a minimum fuzzy valid-time of $f_T \geqslant T$ as the degree of certainty (w) tends to 1.

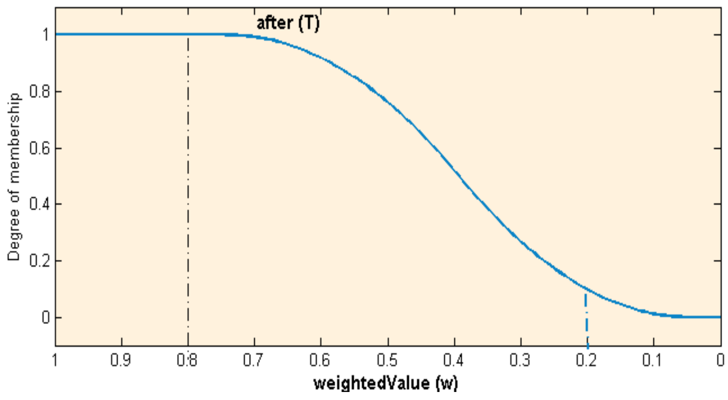


Figure 4.20: Membership Function for ’after (T)’ Fuzzy Temporal Expression

4.5 Summary

In this chapter, the case studies describing the experimentation of ontology engineering, ontology utilization and ontological language extensions were presented. The first case describes the implementation of the underutilized crops ontology (UC-ONTO) describing the results of the knowledge acquisition and conceptualization process, the UC-ONTO standardization through merging and alignment, as well as its extension with SWRL rules for added expressiveness. The chapter further presents the ONto-CropBase tool as a second case study describing the implementation of the UC-ONTO based semantic search engine. The tool’s functionalities realized were presented highlighting the design intuitions behind the search interface, search results presentation as well as the tool’s navigation. Lastly, the language extension case study presents the implementation results of the FT-SWRL model describing the pioneer fuzzy temporal ontology and built-ins. Fuzzification of the fuzzy temporal built-ins to provide a semantic definition and possible reasoning strategy is further explored and presented as the reasoning paradigms for the FT-SWRL model.

In the following chapter, an evaluation of the case studies is discussed leading to three main discussion sections: the UC-ONTO evaluation, the Onto-CropBase evaluation and lastly the FT-SWRL model evaluation.

Chapter 5

Analysis of Results and Discussion

5.1 Introduction

This chapter presents an analysis and discussion of the case study results presented in the preceding chapter. The chapter is structured as follows: Section 5.2 discusses the UC-ONTO evaluation highlighting the ontology metrics, depth and structure, followed by the evaluation of the UC-ONTO using competency questions. A functional evaluation of the ontology extension with SWRL rules is presented in Section 5.2.4 and the decidability of the ontology is also evaluated through reasoning and queries in Section 5.2.5.

The Onto-CropBase tool's evaluation and discussion is presented in Section 5.3, which discusses the performance measurement of semantic searching followed by an analysis of domain experts' evaluation of the tool in Section 5.3.1. Comparison of the Onto-CropBase with other relevant tools is discussed in Section 5.3.2 and the scalability of the tool is discussed in Section 5.3.3. FT-SWRL model evaluation is discussed in Section 5.4 with the ontology's modularization, depth and structure discussed in Section 5.4.1 followed by the ontology metrics evaluation in Section 5.4.2. The usability and rule format of the FT-SWRL model built-ins is further discussed in Section 5.4.3 and lastly, conclude in Section 5.5 with the chapter summary.

5.2 The Underutilized Crops Ontology (UC-ONTO) Evaluation

The previous chapter presented the development process leading to the current version of the underutilized crops ontology (UC-ONTO) [192] using the Protégé 4.2 ontology editor. The ontology currently consists of core SWRL built-ins, OWL-time ontology, and FAO geopolitical ontology as direct imports. This is because these ontologies being domain-independent and available in the OWL format can stand-alone without posing compatibility problems and inconsistencies. As such, in line with the recommendation of ontology reuse, the ontologies are imported into UC-ONTO and utilized for added meaning to the underutilized crops data.

5.2.1 Modularization, Depth and Structure

From the inferred conceptual hierarchy of the UC-ONTO, depicted in Fig. 5.1, it can be seen that the ontology is modularized into specialized ontologies such as, *Naming Ontology, Agronomy, Nutrition, Farming Ontology,* etc. These are aligned together to form a larger ontology model for the underutilized crops domain. Individual instances added to the UC-ONTO through the local ontologies in RDFS, includes *Bambara groundnut, Taro, Moringa, Tef,* and *Wing Bean*.

While details on the development methodology and the aspects of the UC-ONTO (such as agronomic, physiological traits) are excluded for brevity, the ontology is available for download as .owl file in the link[1] and for collaborative development, an interactive version of the ontology is also available in the Web-Protégé — see Fig. 5.2 and accessible through the link[2]. At a glance, the composition of the UC-ONTO is as follows: In the ontology, all crops related concepts were grouped together under the $DomainConcepts$ as an ancestral class for all underutilized crops concepts and their direct relations. Whereas, non-crop info such as $Entity, Temporal, Soil, Temperature, TimeZone,$ etc. and all other related concepts — offered by imported ontologies, were composed as siblings

[1]https://drive.google.com/file/d/0B1FHDywt7JQlWWNhMzRjWmRuNVk/view

[2]https://webprotege.stanford.edu/#Edit:projectId=1475900a-1753-4de5-b411-57cf03d8086c

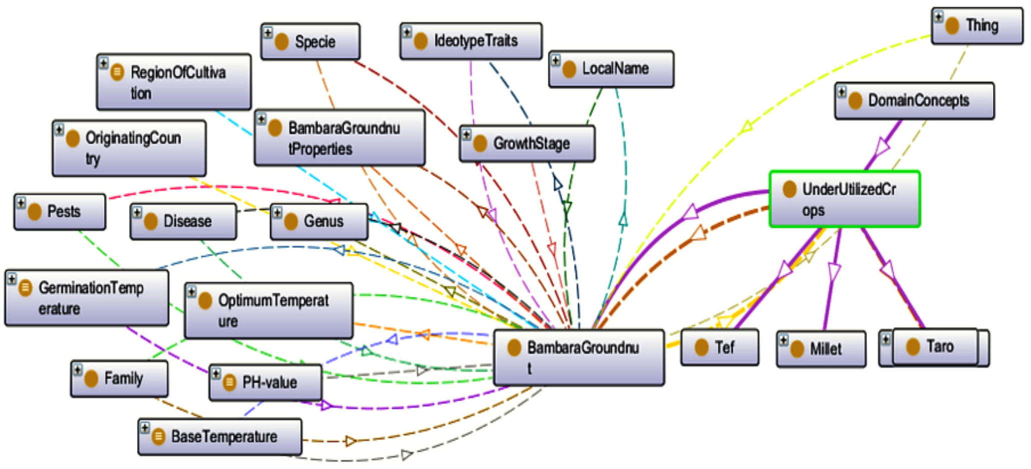


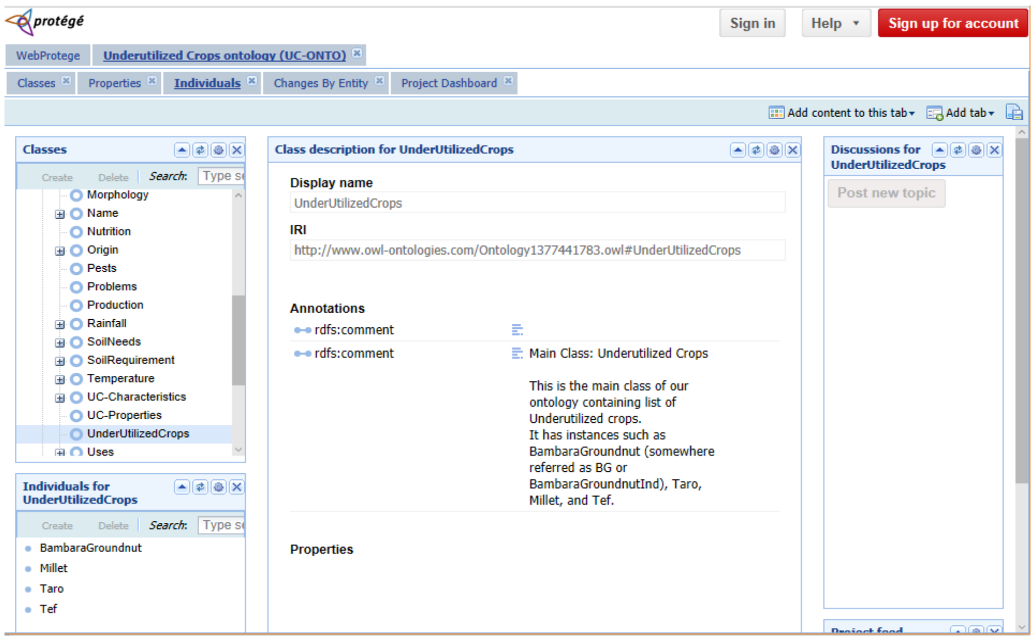
Figure 5.1: A conceptual hierarchy from the Protégé ontology editor showing fragment of UC-ONTO.

of the Domain Concept class. Crops terminologies are further grouped under the $UnderutilizedCrops$ class which contains four main sub-classes and instances: $Taro$, Tef, $Millet$ and $BambaraGroundnut$ class, also added as sub classes (an expressive feature allowed in OWL 2). These instances dominates most of the object and data-type property assertions available in the current version of the UC-ONTO as shown in Fig. 5.1 and 5.2 respectively.

5.2.2 UC-ONTO Metrics — Measuring the Ontology Size and Expressiveness

As discussed in Section 2.7, domain ontologies can also be evaluated based on their depth and contents. The ontology metrics tab in Protégé gives a first-hand information on the amount of concepts, individuals, data values and the relationships both asserted and inferred — also known as 'Axioms'. Other important information that can be provided by the ontology metrics tab is the 'DL expressivity' of an ontology, which depicts the depth of expressiveness of an ontology based on the complexity of asserted conceptualization. We describe the expressiveness of OWL 2 language in Section 2.2.3.2.

As shown in Fig. 5.3, the current version of the Underutilized Crops ontology has: over 53,000 axioms, with over 1500 different crop related concepts (classes,



Display name
UnderUtilizedCrops

IRI
http://www.owl-ontologies.com/Ontology1377441783.owl#UnderUtilizedCrops

rdfs:comment
rdfs:comment Main Class: Underutilized Crops

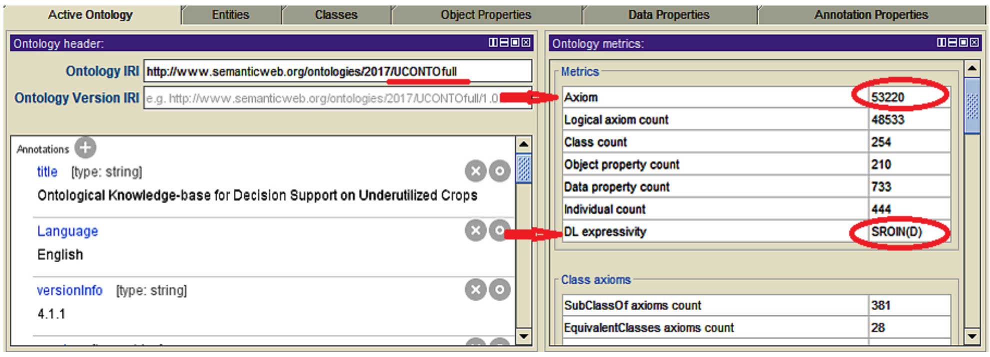
This is the main class of our ontology containing list of Underutilized crops.
It has instances such as BambaraGroundnut (somewhere referred as BG or BambaraGroundnutInd), Taro, Millet, and Tef.

Figure 5.2: The Web-Protégé collaborative ontology development environment showing UC-ONTO.

properties and individuals). The fact that there is a high ratio between the concept to axioms count (1:50 approx) shows that the ontology is well versed in logical assertions that a simple connection of vocabulary — as usually is the case in taxonomies and dictionaries. This logical maturity of ontologies is also highlighted by Protégé by providing a separate count of the 'Logical axioms count', which is seen to be exactly 48,533 as shown in Fig. 5.3.

Regarding the expressiveness metric, UC-ONTO's expressivity is shown to conform to the $\mathcal{SROIN}(\mathcal{D})$ algorithm — see Fig. 5.3. This shows that the ontology is highly expressive as it reached the highest expressiveness allowed by the OWL 2 language, as described in the literature review in Section 2.2.3.2. The expressiveness equivalence of the $\mathcal{SROIN}(\mathcal{D})$ algorithm implies that:

(i) The UC-ONTO knowledge model involves an Attributive Language with Complement $\mathcal{ALC}$ extended with the Role $\mathcal{R}$ hierarchy and roles (properties) transitivity. In other words, the knowledge model involves a Terminology box (T-Box) as well as an individuals Assertion box (A-Box) — represented by $\mathcal{S}$.



Ontology IRI: http://www.semanticweb.org/ontologies/2017/UCONTOfull

Ontology Version IRI: e.g. http://www.semanticweb.org/ontologies/2017/UCONTOfull/1.0

title [type: string]
Ontological Knowledge-base for Decision Support on Underutilized Crops

Language
English

versionInfo [type: string]
4.1.1

Metrics

Axiom	53220
Logical axiom count	48533
Class count	254
Object property count	210
Data property count	733
Individual count	444
DL expressivity	SROIN(D)

Class axioms

SubClassOf axioms count	381
EquivalentClasses axioms count	28

Figure 5.3: UC-ONTO Metrics showing Expressivity, Axioms, classes and property count.

(ii) Added expressiveness implied are the enumerated classes $\mathcal{O}$, inverse role assertions $\mathcal{I}$, and cardinality restrictions $\mathcal{N}$ — discussed in details Section 2.2.3.2.

(iii) The letter $\mathcal{R}$ refers to the presence of role inclusions, local reflexivity Self, and the universal role U, as well as the additional role characteristics of transitivity, symmetry, asymmetry, role disjointness, reflexivity, and irreflexivity.

One noticeable difference is the expressiveness $\mathcal{SROIN}(\mathcal{D})$ of UC-ONTO (OWL 2) instead of $\mathcal{SHOIN}(\mathcal{D})$ of OWL-DL expressivity described in Section 2.2.3.2. This shows the fact that the current version of OWL 2 is evolving beyond the expressive powers of its predecessor, the OWL-DL in that it allows higher expressiveness with regards to the property assertions as much as OWL-DL allows for classes.

5.2.3 UC-ONTO Evaluation using Competency Questions

Competency Questions (CQs) posed as queries to ontological knowledge bases provides a good measure of the ontology quality. Moreover CQs can be used to check whether the functional and non-functional requirements of an ontology have been realized based on the accuracy of query results, the comprehensiveness of domain coverage and conceptualization among other things. Based on the UC-

ONTO's competency questions designed and enumerated in Section 3.2.1, some of the queries proposed to evaluate the UC-ONTO are presented as follows:

Query No.	**For the Underutilized Crops Ontology (UC-ONTO), find:**
Q1.	List of common Underutilized Crops (UCs).
Q2.	The Characteristics/Features of Underutilized crops.
Q3.	The agronomy and properties of Bambara groundnut.
Q4.	Its optimum growing conditions.
Q5.	The Soil/Rainfall/Temperature requirements for growth of Bambara groundnut.
Q6.	The regions of cultivation of Bamabara groundnut.
Q7.	Country of Origin and Landraces of Bambara groundnut for a selected country.
Q8.	Nutrient components of Bambara groundnut.
Q9.	The food parts of the UC.
Q10.	The number of days after sowing for Bambara groundnut to begin flowering.
Q11.	The rainfall characterization for Bambara groundnut.
Q12.	Root type of Bambara groundnut plant.
Q13.	List the commercial products of Bambara groundnut.
Q14.	The common Pests and Diseases of Bambara groundnut.
Q15.	Other local names for Bambara groundnut.

Table 5.1: Competency Queries for the UC-ONTO Evaluation

From the list of queries above, the UC-ONTO can thus be evaluated based on contents and structure such as hierarchy of classes, properties, individuals, disjoint classes, intersection and union of classes among others. In what follows, we present some test results from running DL-queries over the underutilized crops ontology as shown in Table 5.1. For each query, the part of the model explored have been highlighted to further understand the query results. Moreover, the tests were categorized into three cases, (i) Complete — if the test results have completely answered or fulfilled the modeling requirements. (ii) Partial — where the results partially answered the modeling requirement, thereby highlighting the information to be added into the ontology. (iii) Failed — where the test results have completely failed to answer the competency query and therefor the ontology need to be modified or extended.

Query	Model Probed	Competency Status
Q1.	Class enumeration (List of Individual instances)	Complete

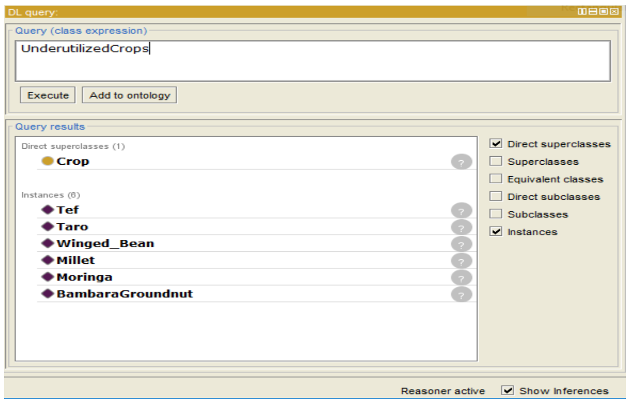

DL query:
Query (class expression)
UnderutilizedCrops
Execute | Add to ontology
Query results
Direct superclasses (1)
● Crop
Instances (6)
◆ Tef
◆ Taro
◆ Winged_Bean
◆ Millet
◆ Moringa
◆ BambaraGroundnut
[✓ Direct superclasses]
Superclasses
Equivalent classes
Direct subclasses
Subclasses
[✓ Instances]
Reasoner active [✓ Show Inferences]

Figure 5.4: DL Query showing results of Competency Query 1.

Query	Model Probed	Competency Status
Q2.	Class enumeration (List of Individual instances)	Complete

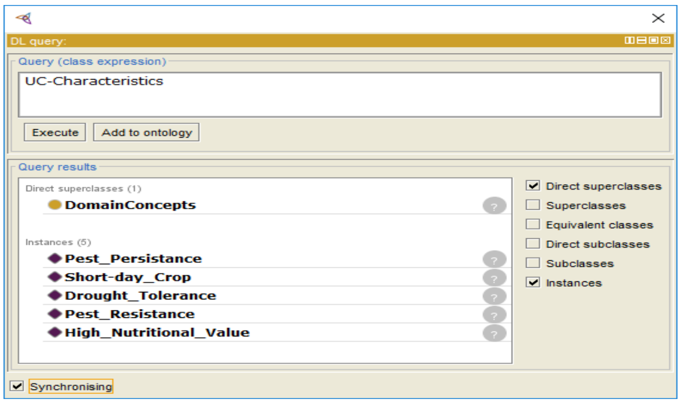
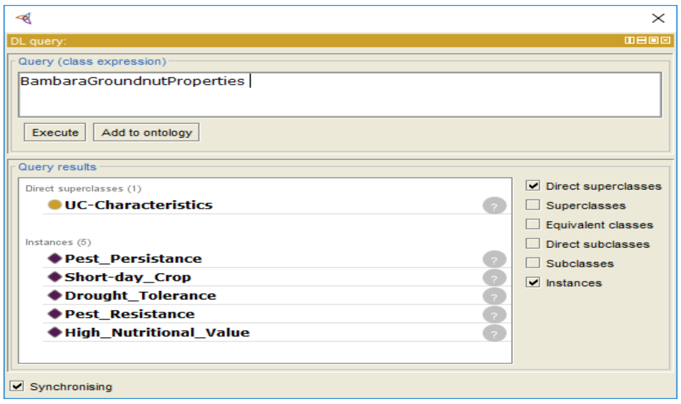

DL query:
Query (class expression)
UC-Characteristics
Execute | Add to ontology
Query results
Direct superclasses (1)
● DomainConcepts
Instances (5)
◆ Pest_Persistance
◆ Short-day_Crop
◆ Drought_Tolerance
◆ Pest_Resistance
◆ High_Nutritional_Value
[✓ Direct superclasses]
Superclasses
Equivalent classes
Direct subclasses
Subclasses
[✓ Instances]
[✓ Synchronising]

Figure 5.5: DL Query showing results of Competency Query 2.

Query	Model Probed	Competency Status
Q3.	Class enumeration (List of Individual instances)	Complete


DL query:

Query (class expression)

BambaraGroundnutProperties

Execute | Add to ontology

Query results

Direct superclasses (1)
- UC-Characteristics

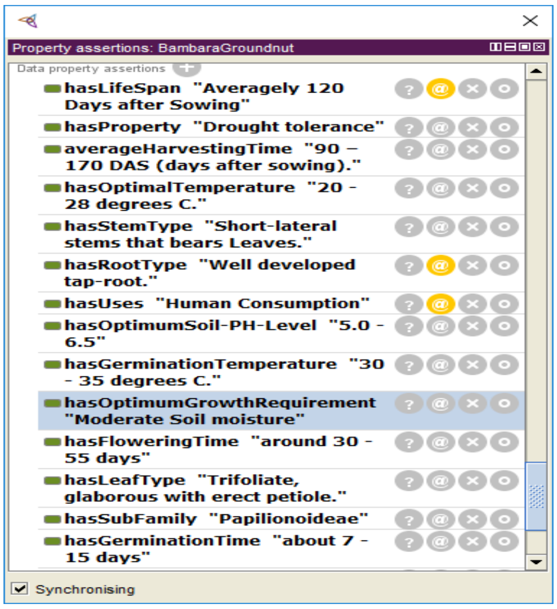
Instances (5)
- Pest_Persistance
- Short-day_Crop
- Drought_Tolerance
- Pest_Resistance
- High_Nutritional_Value

[✓] Direct superclasses
[ ] Superclasses
[ ] Equivalent classes
[ ] Direct subclasses
[ ] Subclasses
[✓] Instances

[✓] Synchronising

Figure 5.6: DL Query showing results of Competency Query 3.

Query	Model Probed	Competency Status
Q4.	Data type Property assertions	Complete

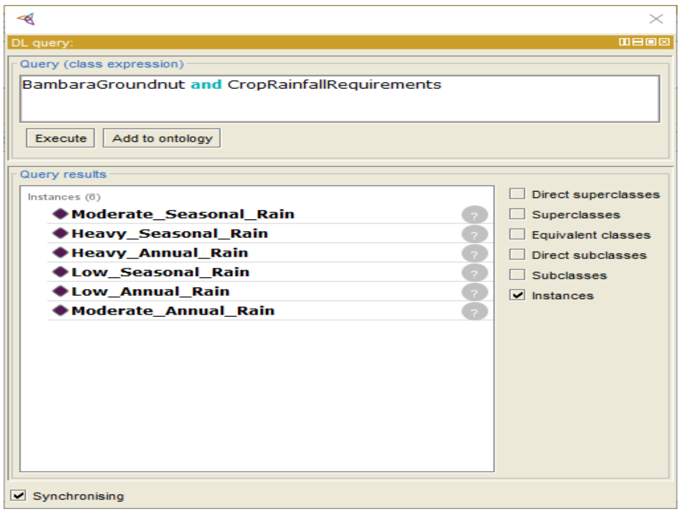

Property assertions: BambaraGroundnut

Data property assertions
- hasLifeSpan "Averagely 120 Days after Sowing"
- hasProperty "Drought tolerance"
- averageHarvestingTime "90 – 170 DAS (days after sowing)."
- hasOptimalTemperature "20 - 28 degrees C."
- hasStemType "Short-lateral stems that bears Leaves."
- hasRootType "Well developed tap-root."
- hasUses "Human Consumption"
- hasOptimumSoil-PH-Level "5.0 - 6.5"
- hasGerminationTemperature "30 - 35 degrees C."
- hasOptimumGrowthRequirement "Moderate Soil moisture"
- hasFloweringTime "around 30 - 55 days"
- hasLeafType "Trifoliate, glaborous with erect petiole."
- hasSubFamily "Papilionoideae"
- hasGerminationTime "about 7 - 15 days"

[✓] Synchronising

Figure 5.7: DL Query showing results of Competency Query 4.

Query	Model Probed	Competency Status
Q5.	Intersection of Classes (Instances list)	Partial

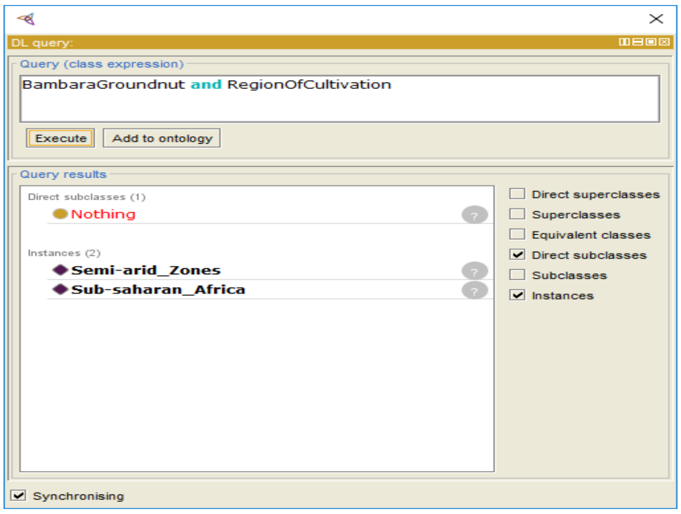


BambaraGroundnut and CropRainfallRequirements

Instances (6)
Moderate_Seasonal_Rain
Heavy_Seasonal_Rain
Heavy_Annual_Rain
Low_Seasonal_Rain
Low_Annual_Rain
Moderate_Annual_Rain

Figure 5.8: DL Query showing results of Competency Query 5.

Query	Model Probed	Competency Status
Q6.	Joint class enumeration (Individual instances)	Complete



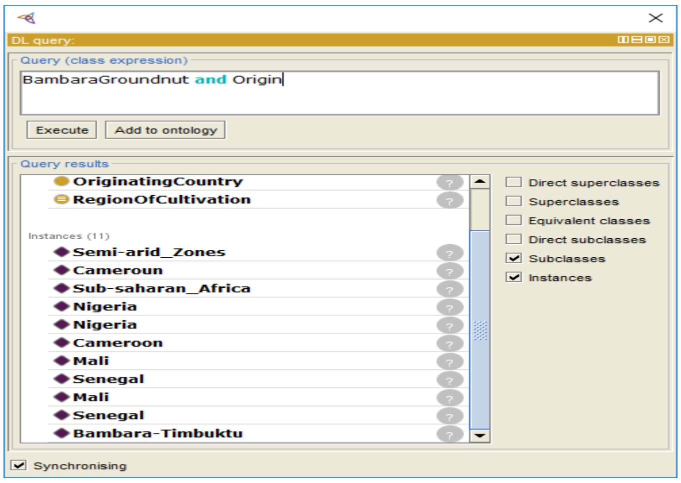
BambaraGroundnut and RegionOfCultivation

Direct subclasses (1)
Nothing

Instances (2)
Semi-arid_Zones
Sub-saharan_Africa

Figure 5.9: DL Query showing results of Competency Query 6.

Query	Model Probed	Competency Status
Q7.	Joint class enumeration (Individual instances)	Complete

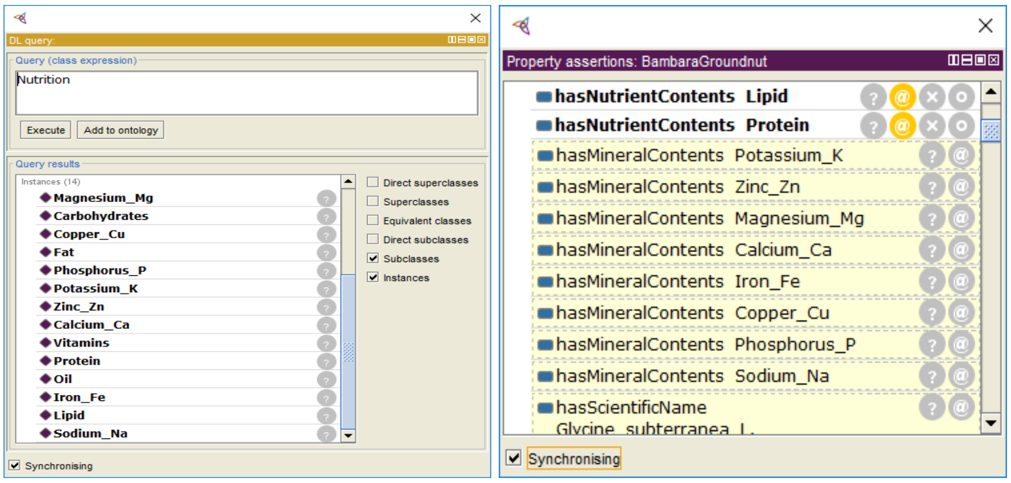

DL query:
Query (class expression)
BambaraGroundnut and Origin
Execute | Add to ontology
Query results
OriginatingCountry
RegionOfCultivation
Instances (11)
Semi-arid_Zones
Cameroun
Sub-saharan_Africa
Nigeria
Nigeria
Cameroon
Mali
Senegal
Mali
Senegal
Bambara-Timbuktu

☐ Direct superclasses
☐ Superclasses
☐ Equivalent classes
☐ Direct subclasses
☑ Subclasses
☑ Instances

☑ Synchronising

Figure 5.10: DL Query showing results of Competency Query 6.

Query	Model Probed	Competency Status
Q8.	Class enumeration, Properties and Datatype Inferences (logical assertions)	Complete


DL query:
Query (class expression)
Nutrition
Execute | Add to ontology
Query results
Instances (14)
Magnesium_Mg
Carbohydrates
Copper_Cu
Fat
Phosphorus_P
Potassium_K
Zinc_Zn
Calcium_Ca
Vitamins
Protein
Oil
Iron_Fe
Lipid
Sodium_Na

☐ Direct superclasses
☐ Superclasses
☐ Equivalent classes
☐ Direct subclasses
☑ Subclasses
☑ Instances

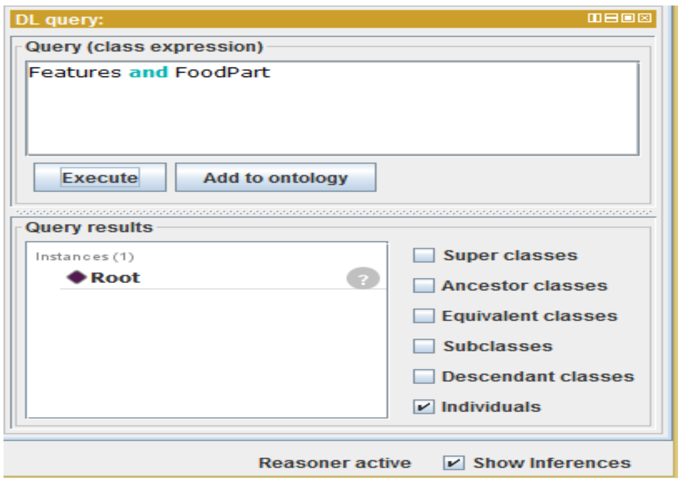
☑ Synchronising

Property assertions: BambaraGroundnut
hasNutrientContents Lipid
hasNutrientContents Protein
hasMineralContents Potassium_K
hasMineralContents Zinc_Zn
hasMineralContents Magnesium_Mg
hasMineralContents Calcium_Ca
hasMineralContents Iron_Fe
hasMineralContents Copper_Cu
hasMineralContents Phosphorus_P
hasMineralContents Sodium_Na
hasScientificName
Glycine_subterranea_L.

☑ Synchronising

Figure 5.11: DL Query (left) and Inferred Property assertions (right) showing results of Competency Query 8.

Query	Model Probed	Competency Status
Q9.	Class intersections (Individual instances)	Complete


DL query:

Query (class expression)

```
Features and FoodPart
```

Query results

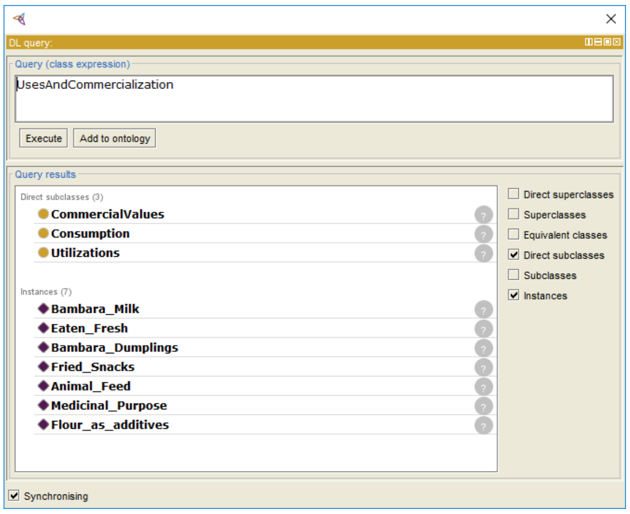
Instances (1)
- Root

- [ ] Super classes
- [ ] Ancestor classes
- [ ] Equivalent classes
- [ ] Subclasses
- [ ] Descendant classes
- [✓] Individuals

Reasoner active [✓] Show Inferences

Figure 5.12: DL Query showing results of Competency Query 9.

Query	Model Probed	Competency Status
Q13.	Class Enumeration (Individual Instances)	Complete


DL query:

Query (class expression)

```
UsesAndCommercialization
```

Query results

Direct subclasses (3)
- CommercialValues
- Consumption
- Utilizations

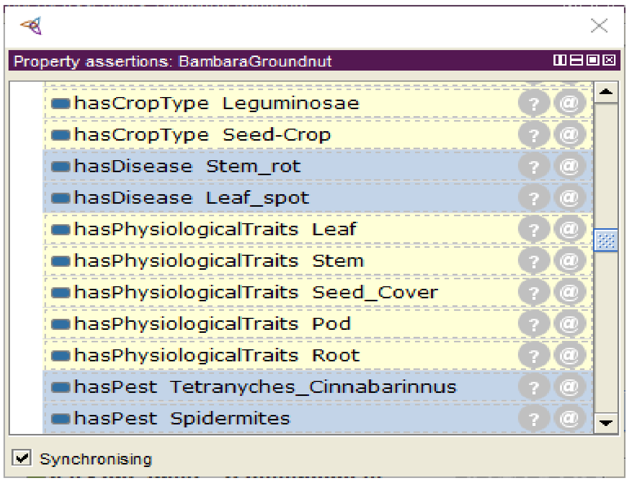
Instances (7)
- Bambara_Milk
- Eaten_Fresh
- Bambara_Dumplings
- Fried_Snacks
- Animal_Feed
- Medicinal_Purpose
- Flour_as_additives

- [ ] Direct superclasses
- [ ] Superclasses
- [ ] Equivalent classes
- [✓] Direct subclasses
- [ ] Subclasses
- [✓] Instances

[✓] Synchronising

Figure 5.13: DL Query showing results of Competency Query 13.

Query	Model Probed	Competency Status
Q14.	Property Assertions (Logical Inferences)	Complete

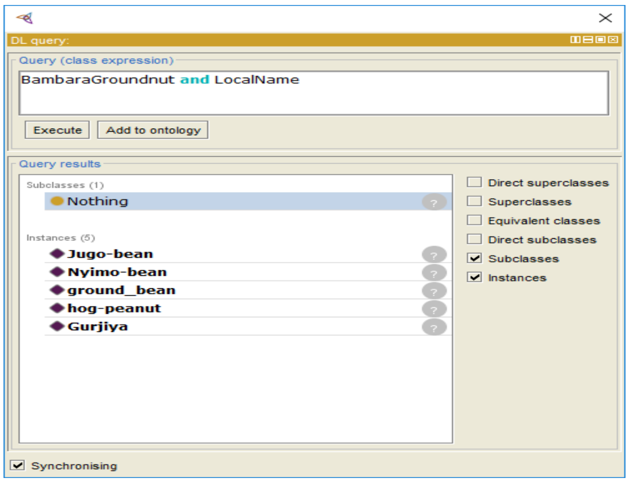

Property assertions: BambaraGroundnut

```
hasCropType  Leguminosae
hasCropType  Seed-Crop
hasDisease  Stem_rot
hasDisease  Leaf_spot
hasPhysiologicalTraits  Leaf
hasPhysiologicalTraits  Stem
hasPhysiologicalTraits  Seed_Cover
hasPhysiologicalTraits  Pod
hasPhysiologicalTraits  Root
hasPest  Tetranyches_Cinnabarinnus
hasPest  Spidermites
```

☑ Synchronising

Figure 5.14: Inferred Property assertions showing results of Competency Query 14 (Pests and Diseases of Bambara groundnut).

Query	Model Probed	Competency Status
Q15.	Joint Class enumeration (Individual Instances)	Complete


DL query:

Query (class expression)

```
BambaraGroundnut and LocalName
```

Execute | Add to ontology

Query results

Subclasses (1)
- Nothing

Instances (5)
- Jugo-bean
- Nyimo-bean
- ground_bean
- hog-peanut
- Gurjiya

☐ Direct superclasses
☐ Superclasses
☐ Equivalent classes
☐ Direct subclasses
☑ Subclasses
☑ Instances

☑ Synchronising

Figure 5.15: DL Query showing results of Competency Query 15.

The competency status for Query 5 is considered to be partial as only the rainfall requirements are presented. The complete information on Bambara groundnut growth requirements were asserted as combination of class instances, object and datatype properties as shown in the results of Queries 4, 8 and 14.

Results for Competency Queries 10 — 12 were not found in the first instance of the evaluation and therefore had to be added. We use the SWRL rules assertions to extend the existing knowledge model to incorporate the required conditions for the queries — as shown in the SWRL validation results in the next sub-section. More precisely, Query 10 has been answered by the assertion of SWRL Rule No. 6. Followed by Query 11 which was answered by the assertion of Rules 8 and 10 respectively. Whereas, Query 12 was answered through the assertion of Rule No. 11. These queries and various other assertions were validated to show the usability of SWRL rules in extending the UC-ONTO knowledge model as highlighted in the SWRL validation table.

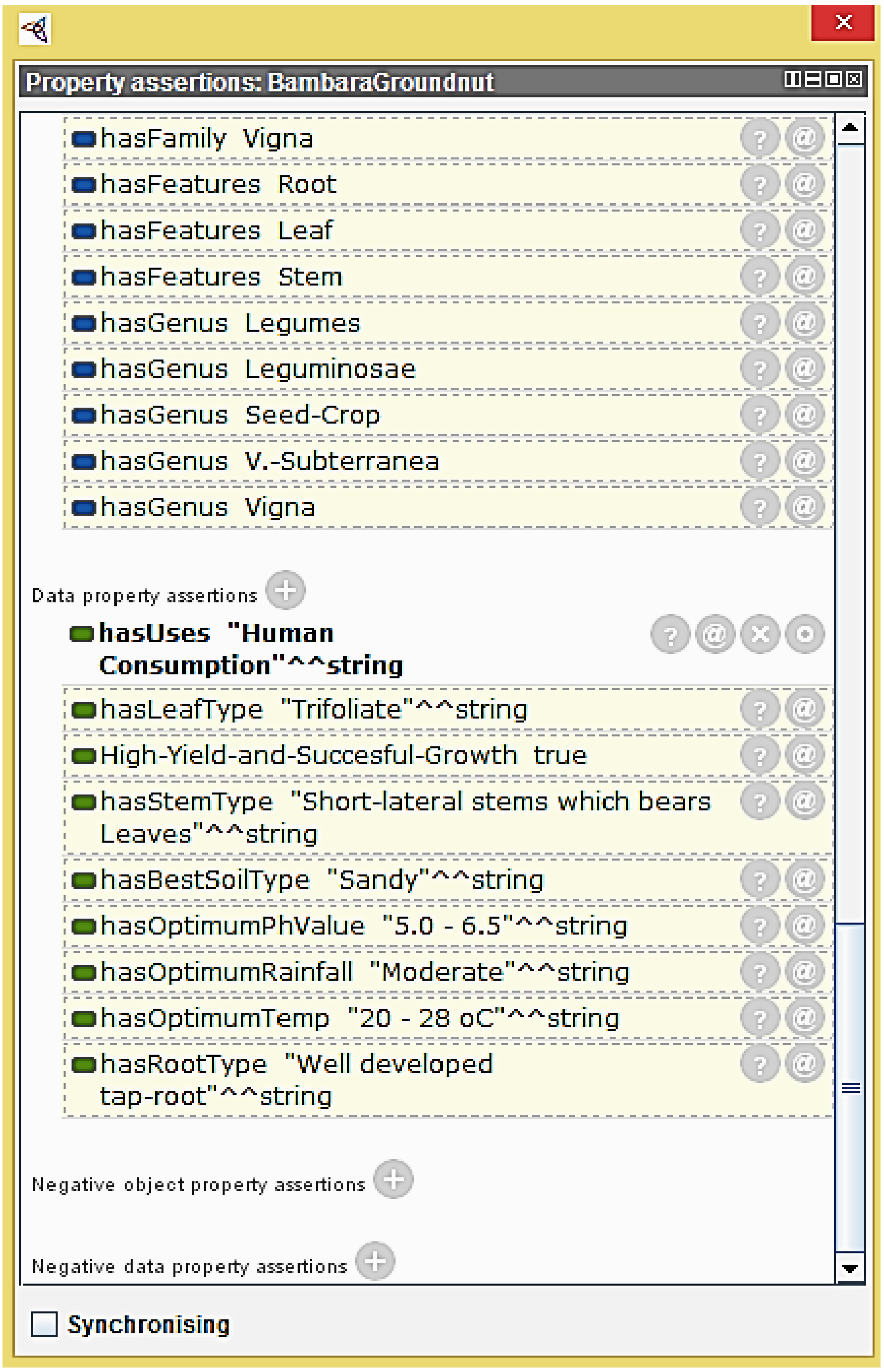
5.2.4 SWRL Rules Extension — Functional Evaluation

The UC-ONTO extension with SWRL rules is also described with example rules written using the SWRL tab to specify complex domain facts, This gives more flexibility through declarative class and property assertions in the UC-ONTO. Fig. 5.16 shows the reasoner inferences on the SWRL rules (righ) highlighted as 'Property assertions of the Bambara groundnut class.

Inferences on OWL assertions is also possible by invoking any of the DL reasoners available in the Protégé development environments. As mentioned earlier, common reasoners include the Hermit, Fact++, Pellet reasoner etc. The Pellet [193] is found to be more suitable for inference on SWRL rules. We briefly discuss the Reasoning and query processing in the next sub-section.

SWRL Validation Similarly, the user-defined SWRL rule assertions were validated by writing DL queries to check their resulting inference or otherwise. Some of the Rule assertions from Fig. 5.16 are discussed below highlighting their syntaxes, purpose and the results of their execution validated:

Rule	SWRL Syntax
R1.	`BambaraGroundnut(?x), Properties(?y) → hasProperty(?x, ?y)`
	• This rule asserts that every member of BambaraGroundnut (BG) class hasProperty in Properties class. **Validation Result**: see Fig. R3: Data property tab showing Reasoner inferences below.
R2.	`BambaraGroundnut(?b), CultivationRegion(?z), Sandy(?x), hasSoilType(?z, ?x) → hasBetterGrowth(?b, true), hasEasyHarvest(?b, true)`
	• This rule will assert the Boolean data-types properties *hasBetterGrowth* and *hasEasyHarvest* of BG class to true if the Cultivation region of BG has a Sandy soil type. **Validation Result**: see Fig. R3: Data property tab showing Reasoner inferences below.
R3.	`BambaraGroundnut(?b), CultivationRegion(?z), LightLoam(?y), hasSoilType(?z, ?y) →hasBetterGrowth(?b, true)`
	• Here, if the Cultivation region has Light-loam as soil type, the rule will assert that BG has Better growth. **Validation Result**: see Fig. R3: Data property tab showing Reasoner inferences below.



Property assertions: BambaraGroundnut

hasFamily Vigna
hasFeatures Root
hasFeatures Leaf
hasFeatures Stem
hasGenus Legumes
hasGenus Leguminosae
hasGenus Seed-Crop
hasGenus V.-Subterranea
hasGenus Vigna

Data property assertions

hasUses "Human Consumption"^^string
hasLeafType "Trifoliate"^^string
High-Yield-and-Succesful-Growth true
hasStemType "Short-lateral stems which bears Leaves"^^string
hasBestSoilType "Sandy"^^string
hasOptimumPhValue "5.0 - 6.5"^^string
hasOptimumRainfall "Moderate"^^string
hasOptimumTemp "20 - 28 oC"^^string
hasRootType "Well developed tap-root"^^string

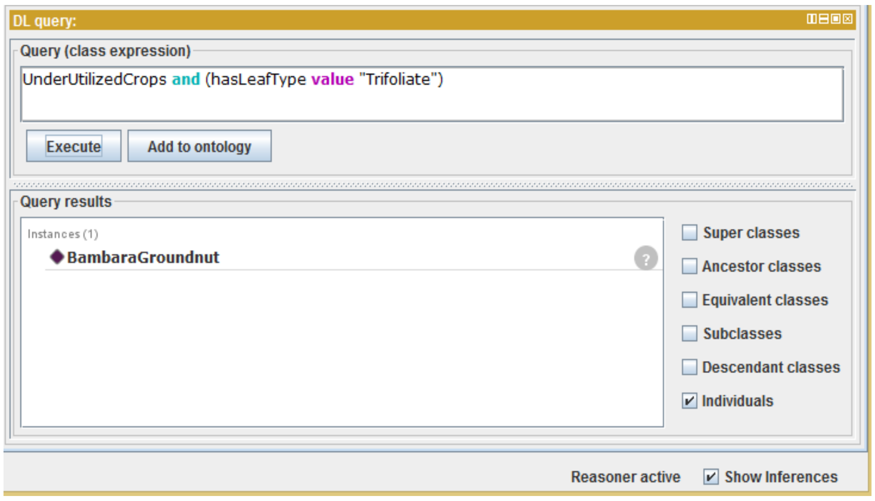
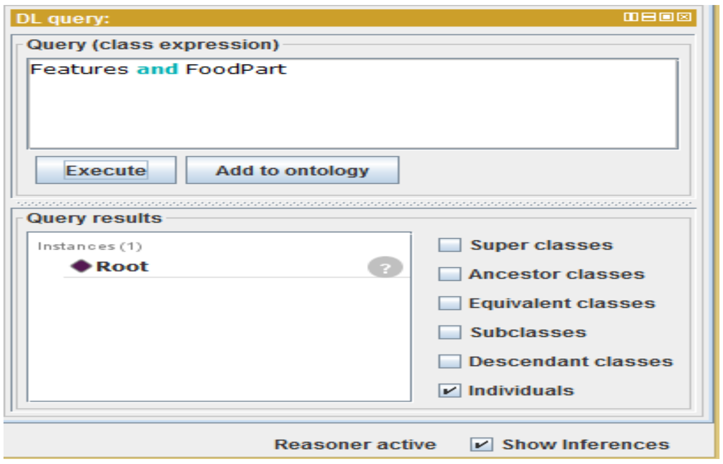
Negative object property assertions

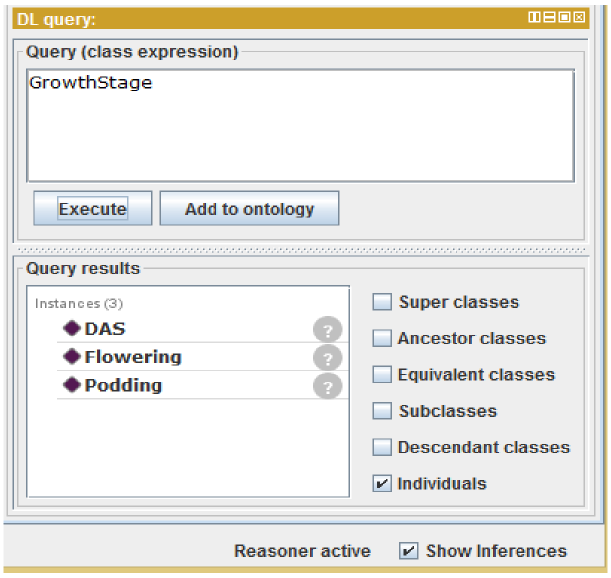
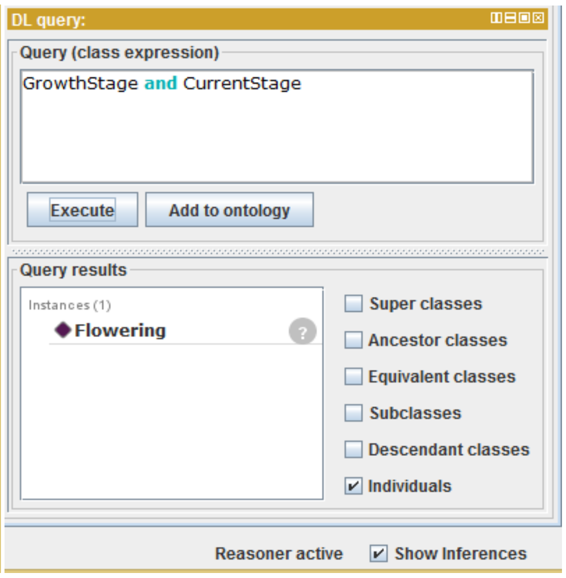
Negative data property assertions

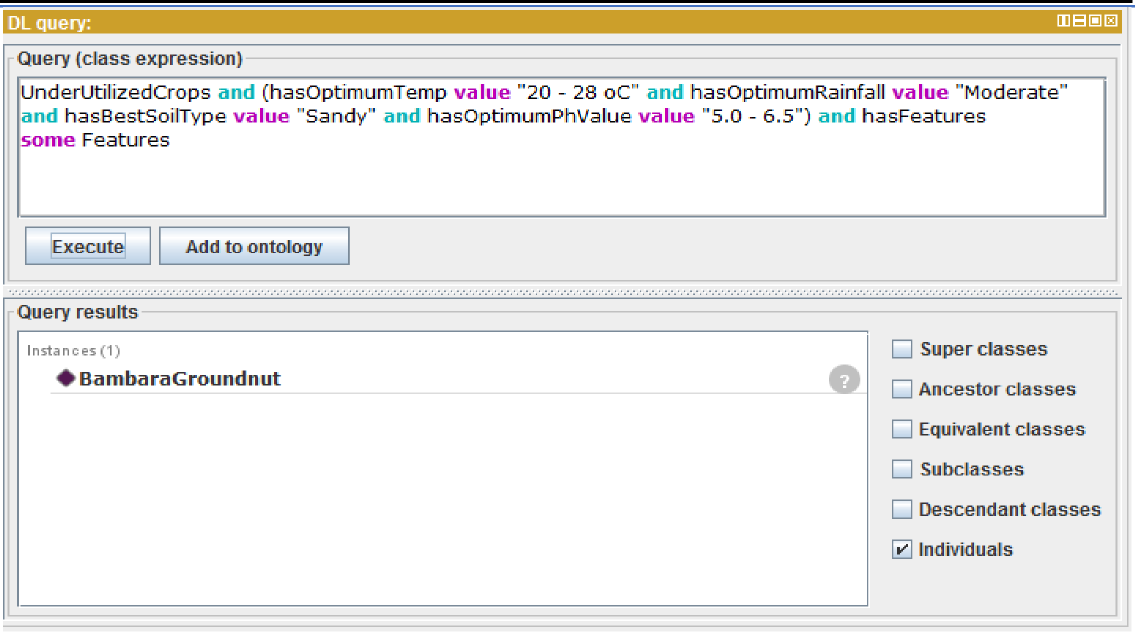
Synchronising

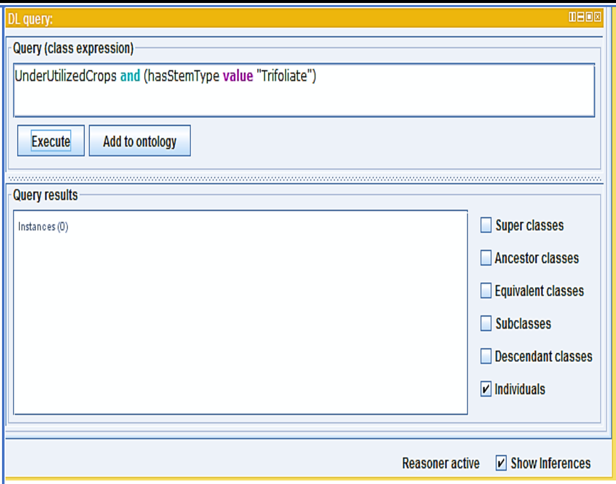
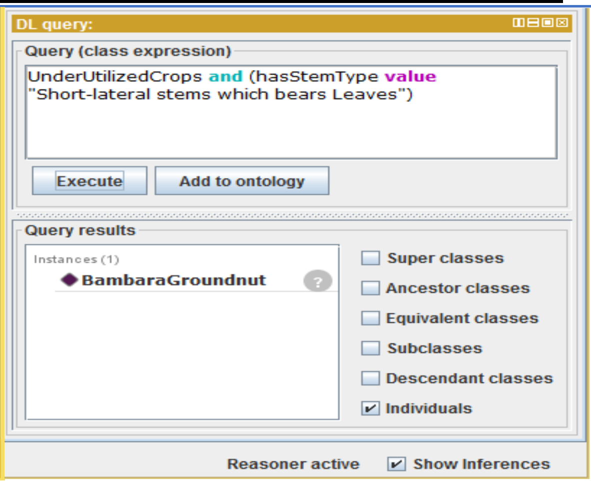
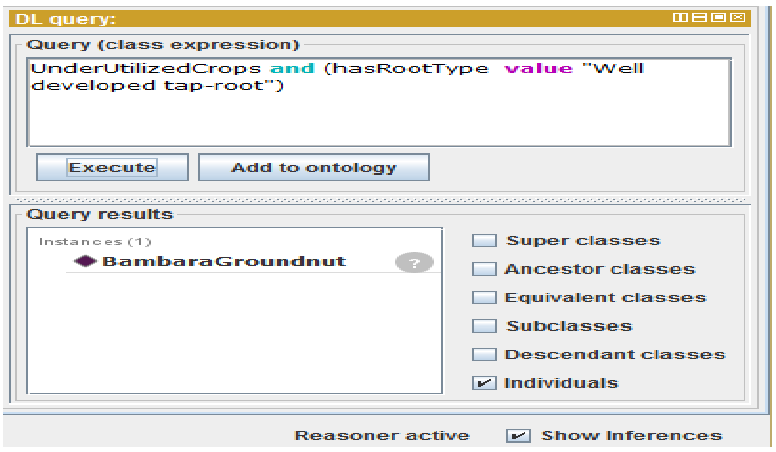
Fig. R3: Reasoner inference on SWRL rules (Data properties asserted and Inferred)

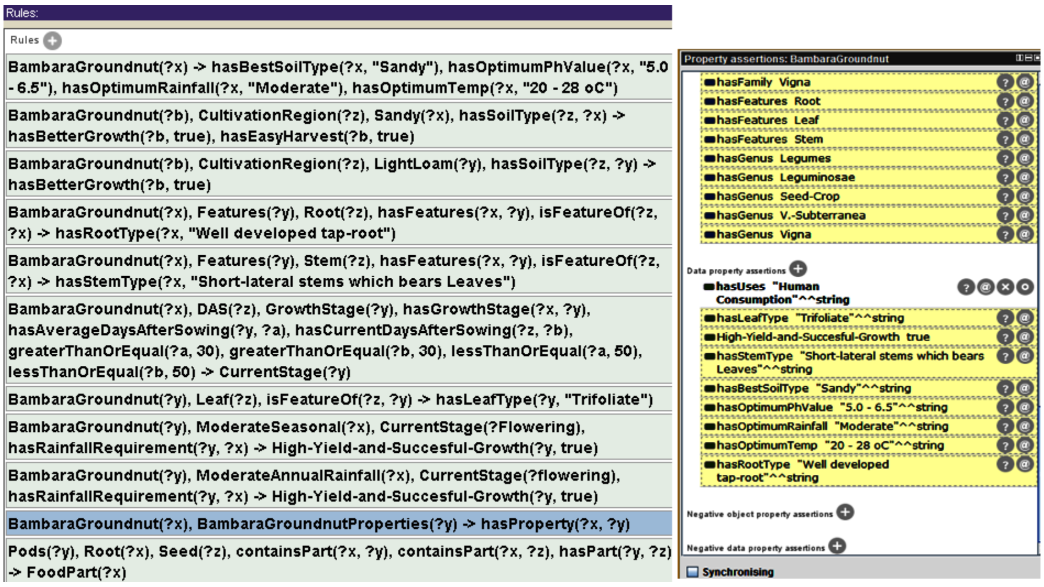
R4.	`BambaraGroundnut(?y), Features(?x), Leaf(?z), hasFeatures(?y, ?x), isFeatureOf(?z, ?y) →hasLeafType(?y, "Trifoliate")`

	• Here, if it is true that BG has a feature and the feature is a leaf, then it will assert that the leaf type is 'trifoliate'. Since features such as leaf are not exclusive to BG, then unless the leaf individual is related to BG individual, the type 'trifoliate' will not be asserted. **Note:** Property assertions based on conditions, such as this, are impossible to be expressed in OWL 2 alone.  DL query: Query (class expression) UnderUtilizedCrops and (hasLeafType value "Trifoliate") Query results Instances (1) BambaraGroundnut
R5.	`Pods(?y), Root(?x), Seed(?z), containsPart(?x, ?y), containsPart(?x, ?z), hasPart(?y, ?z) → FoodPart(?x)`
	• This rule tests for 'Food parts' of BG from the set of individuals in the ontology, If a Root contains a Pod and also contains Seed and Seed and Pod are symmetrically related (with *hasPart* relation) then Root should be asserted in to the 'FoodPart' class of BG.  DL query: Query (class expression) Features and FoodPart Query results Instances (1) Root

R6.	`BambaraGroundnut(?x), DAS(?z), GrowthStage(?y), hasGrowthStage(?x, ?y), hasAverageDaysAfterSowing(?y, ?b), hasCurrentDaysAfterSowing(?z, ?a), equal(?a, ?b) → CurrentStage(?y)`
	• This rule uses the SWRL built-in '*swrlb: equal'* to compare the number of days BG is planted and the number of days asserted for the different growth stages (e.g. Flowering *AverageDaysAfterSowing* = 50) if it founds a match, then it will be asserted as the current stage of BG.  DL query: Query (class expression) GrowthStage Execute · Add to ontology Query results Instances (3) ◆DAS ◆Flowering ◆Podding ☐ Super classes ☐ Ancestor classes ☐ Equivalent classes ☐ Subclasses ☐ Descendant classes ☑ Individuals Reasoner active ☑ Show Inferences  DL query: Query (class expression) GrowthStage and CurrentStage Execute · Add to ontology Query results Instances (1) ◆Flowering ☐ Super classes ☐ Ancestor classes ☐ Equivalent classes ☐ Subclasses ☐ Descendant classes ☑ Individuals Reasoner active ☑ Show Inferences (Querying all members of *GrowthStage* class) (Querying those that are also in *CurrenStage*)
R7.	`BambaraGroundnut(?x) → hasBestSoilType(?x, "Sandy"), hasOptimumPhValue(?x, "5.0 - 6.5"), hasOptimumRainfall(?x, "Moderate"), hasOptimumTemp(?x, "20 – 28°C")`
	• This rule is an optimum condition test for Bambara Groundnut (BG). It can be implemented as a result of a query ''what are the best conditions for BG farming''

	 DL query: Query (class expression) UnderUtilizedCrops and (hasOptimumTemp value "20 - 28 oC" and hasOptimumRainfall value "Moderate" and hasBestSoilType value "Sandy" and hasOptimumPhValue value "5.0 - 6.5") and hasFeatures some Features Execute · Add to ontology Query results Instances (1) ◆BambaraGroundnut ☐ Super classes ☐ Ancestor classes ☐ Equivalent classes ☐ Subclasses ☐ Descendant classes ☑ Individuals
R8.	`BambaraGroundnut(?y), ModerateAnnualRainfall(?x), CurrentStage(?Flowering), hasRainfallRequirement(?y, ?x) → High-Yield-and-Successful-Growth(?y, true)`
	• This rule asserts to true that BG has High-Yield-and-Succesful-Growth if its current stage is flowering and rainfall is moderate annually.
R9.	`BambaraGroundnut(?x), Features(?y), Stem(?z), hasFeatures(?x, ?y), isFeatureOf(?z, ?x) →hasStemType(?x, "Short-lateral stems which bears Leaves")`
	• A version of rule 4 that asserts that if Stem is a feature of BG then it has a “Short-lateral stem which bears leaves” as the Stem type.

	 Query (class expression): UnderUtilizedCrops and (hasStemType value "Trifoliate") Query results: Instances (0) DL query: Execute Add to ontology Super classes Ancestor classes Equivalent classes Subclasses Descendant classes Individuals Reasoner active Show Inferences Wrong value (No results)  Query (class expression): UnderUtilizedCrops and (hasStemType value "Short-lateral stems which bears Leaves") Query results: Instances (1) BambaraGroundnut DL query: Execute Add to ontology Super classes Ancestor classes Equivalent classes Subclasses Descendant classes Individuals Reasoner active Show Inferences Correct value(returns result)
R10	`BambaraGroundnut(?y), ModerateSeasonal(?x), CurrentStage(?Flowering), hasRainfallRequirement(?y, ?x) → High-Yield-and-Successful-Growth(?y, true)`
	• Similar to rule 8, however, this rule asserts to true that BG has High-Yield-and-Successful-Growth if its current stage is flowering and rainfall is moderate seasonally. (Has to be written separate since SWRL rules cannot have disjunctions. **Validation Result**: see Fig. R3: Data property tab showing Reasoner inferences above.
R11	`BambaraGroundnut(?x), Features(?y), Root(?z), hasFeatures(?x, ?y), isFeatureOf(?z, ?x) →hasRootType(?x, "Well developed tap-root")`
	• A version of rule 4 that asserts that if Root is a feature of BG then it *hasRootType* "Short Well developed tap-root".  Query (class expression): UnderUtilizedCrops and (hasRootType value "Well developed tap-root") Query results: Instances (1) BambaraGroundnut DL query: Execute Add to ontology Super classes Ancestor classes Equivalent classes Subclasses Descendant classes Individuals Reasoner active Show Inferences



```
Rules:
Rules
BambaraGroundnut(?x) -> hasBestSoilType(?x, "Sandy"), hasOptimumPhValue(?x, "5.0
- 6.5"), hasOptimumRainfall(?x, "Moderate"), hasOptimumTemp(?x, "20 - 28 oC")
BambaraGroundnut(?b), CultivationRegion(?z), Sandy(?x), hasSoilType(?z, ?x) ->
hasBetterGrowth(?b, true), hasEasyHarvest(?b, true)
BambaraGroundnut(?b), CultivationRegion(?z), LightLoam(?y), hasSoilType(?z, ?y) ->
hasBetterGrowth(?b, true)
BambaraGroundnut(?x), Features(?y), Root(?z), hasFeatures(?x, ?y), isFeatureOf(?z,
?x) -> hasRootType(?x, "Well developed tap-root")
BambaraGroundnut(?x), Features(?y), Stem(?z), hasFeatures(?x, ?y), isFeatureOf(?z,
?x) -> hasStemType(?x, "Short-lateral stems which bears Leaves")
BambaraGroundnut(?x), DAS(?z), GrowthStage(?y), hasGrowthStage(?x, ?y),
hasAverageDaysAfterSowing(?y, ?a), hasCurrentDaysAfterSowing(?z, ?b),
greaterThanOrEqual(?a, 30), greaterThanOrEqual(?b, 30), lessThanOrEqual(?a, 50),
lessThanOrEqual(?b, 50) -> CurrentStage(?y)
BambaraGroundnut(?y), Leaf(?z), isFeatureOf(?z, ?y) -> hasLeafType(?y, "Trifoliate")
BambaraGroundnut(?y), ModerateSeasonal(?x), CurrentStage(?Flowering),
hasRainfallRequirement(?y, ?x) -> High-Yield-and-Succesful-Growth(?y, true)
BambaraGroundnut(?y), ModerateAnnualRainfall(?x), CurrentStage(?flowering),
hasRainfallRequirement(?y, ?x) -> High-Yield-and-Succesful-Growth(?y, true)
BambaraGroundnut(?x), BambaraGroundnutProperties(?y) -> hasProperty(?x, ?y)
Pods(?y), Root(?x), Seed(?z), containsPart(?x, ?y), containsPart(?x, ?z), hasPart(?y, ?z)
-> FoodPart(?x)
```

```
Property assertions: BambaraGroundnut
hasFamily  Vigna
hasFeatures  Root
hasFeatures  Leaf
hasFeatures  Stem
hasGenus  Legumes
hasGenus  Leguminosae
hasGenus  Seed-Crop
hasGenus  V.-Subterranea
hasGenus  Vigna

Data property assertions
hasUses  "Human
Consumption"^^string
hasLeafType  "Trifoliate"^^string
High-Yield-and-Succesful-Growth  true
hasStemType  "Short-lateral stems which bears
Leaves"^^string
hasBestSoilType  "Sandy"^^string
hasOptimumPhValue  "5.0 - 6.5"^^string
hasOptimumRainfall  "Moderate"^^string
hasOptimumTemp  "20 - 28 oC"^^string
hasRootType  "Well developed
tap-root"^^string

Negative object property assertions
Negative data property assertions
Synchronising
```

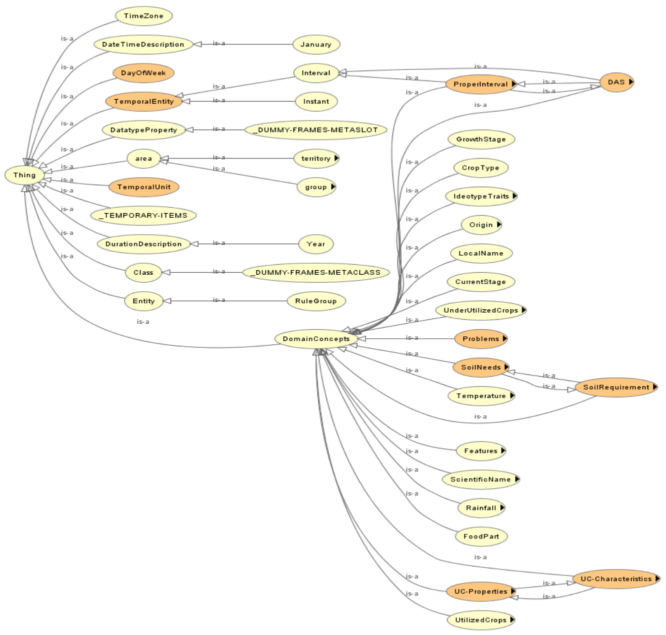
Figure 5.16: Some SWRL Rules of the UC-ONTO and their inference

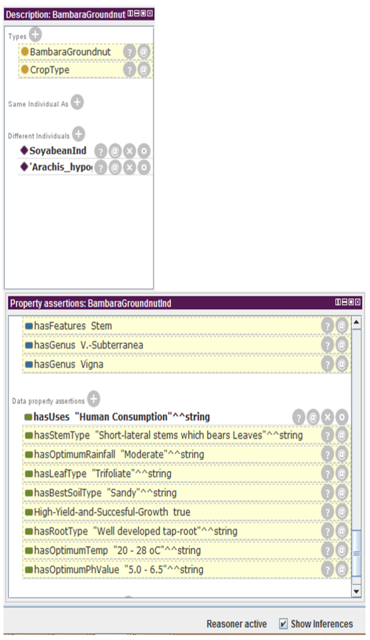
5.2.5 Decidability, Reasoning and query processing

Interactive ontology development environments such as Protégé allows for measuring ontology performance both at the design stage as well as at run-time. This is achieve via a reasoner to compute the ontology classification and ensure consistency. As such, during query writing, a reasoner needs to be active and the ontology classified before writing any DL Queries.

The conceptualization, hierarchy and user-defined SWRL rules were also validated by writing DL queries to check their inference or otherwise by the reasoner. For example, the query result of rule 4.1, which determines the current 'growth stage' of a $BambaraGroundnut$ plant is depicted in Fig. 5.17 (right). The remaining inferences are results of the rules listed in the SWRL rule tab as shown in Fig. 5.16 — precisely those of rules $1, 2, 3, 8$ and 10, on the rule tab. The rules basically assert data-type properties to the $BambaraGroundnut$ individual based on the inference provided by the Pellet reasoner. Fig. 5.17 below gives a graphic overview of the concepts, roles and rules specified in the UC-ONTO (left) with the non-visualized assertions highlighted as inferences of Bambara groundnut properties.

Example DL-queries written at the run-time is depicted in Fig. 5.18. DL-




Description: BambaraGroundnut

Types
BambaraGroundnut
CropType

Same Individual As

Different Individuals
SoyabeanInd
'Arachis_hypo

Property assertions: BambaraGroundnutInd

hasFeatures Stem
hasGenus V.-Subterranea
hasGenus Vigna

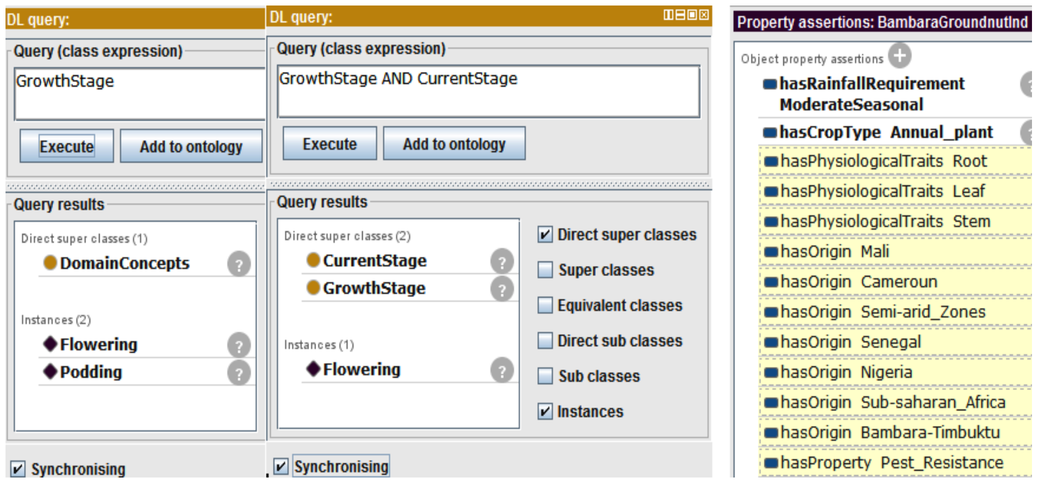
Data property assertions
hasUses "Human Consumption"^^string
hasStemType "Short-lateral stems which bears Leaves"^^string
hasOptimumRainfall "Moderate"^^string
hasLeafType "Trifoliate"^^string
hasBestSoilType "Sandy"^^string
High-Yield-and-Succesful-Growth true
hasRootType "Well developed tap-root"^^string
hasOptimumTemp "20 - 28 oC"^^string
hasOptimumPhValue "5.0 - 6.5"^^string

Reasoner active ✓ Show Inferences

Figure 5.17: Graphic view of concepts in UC-ONTO (left) and Inferences on $BambaraGroundnut$ individual (right)

queries allows writing Select queries in the Protege editor by simply writing class names — hence the name DL-query. Conjunctions and disjunctions of classes is also possible through the logical operators: 'AND', 'OR' — as shown in the second DL-query, which evaluates the intersection of the 'GrowthStage' and 'CurrentStage' classes of the Bambara groundnut concept. Unlike the first DL query, which shows two instances of the 'GrowthStage' class, the intersection of the two classes is shown to contain only one instance, the 'Flowering Stage'. This shows that the Current Growth Stage of the crop, as asserted by rule 4.1, is Flowering stage. Similarly, non-visualized assertions are highlighted as inferences of Bambara groundnut properties on the right of Fig. 5.18.

As explained in the Preliminaries Section in 2.2.1, Decidability refers to the ability of a reasoner to classify an ontology and achieve inference over a finite time. That is, given any ontology, a set of rules, and a sentence, the Reasoner can check that the sentence is entailed by the ontology and rules. As such, where an inference is achieved within a finite time, an ontology can be said to be decidable. From the query results and reasoner inferences shown, it can be concluded there-


DL query:
Query (class expression)
GrowthStage
Execute | Add to ontology
Query results
Direct super classes (1)
DomainConcepts
Instances (2)
Flowering
Podding
☑ Synchronising

DL query:
Query (class expression)
GrowthStage AND CurrentStage
Execute | Add to ontology
Query results
Direct super classes (2)
CurrentStage
GrowthStage
Instances (1)
Flowering
☑ Direct super classes
☐ Super classes
☐ Equivalent classes
☐ Direct sub classes
☐ Sub classes
☑ Instances
☑ Synchronising

Property assertions: BambaraGroundnutInd
Object property assertions
hasRainfallRequirement ModerateSeasonal
hasCropType Annual_plant
hasPhysiologicalTraits Root
hasPhysiologicalTraits Leaf
hasPhysiologicalTraits Stem
hasOrigin Mali
hasOrigin Cameroun
hasOrigin Semi-arid_Zones
hasOrigin Senegal
hasOrigin Nigeria
hasOrigin Sub-saharan_Africa
hasOrigin Bambara-Timbuktu
hasProperty Pest_Resistance

Figure 5.18: DL-query Evaluations showing resulting Inferences

fore that the UC-ONTO is decidable. More importantly, the results for each query is returned within a reasonable amount of time.

This is achieved through careful modeling and conceptualization, where the domain concepts and their relationships are carefully added with proper restrictions through the domain and ranges. Furthermore, the SWRL rules can be kept decidable through the DL-Safety restriction, where only concepts already defined in the main ontology were utilized in wirting the SWRL rules. In order to maintain the decidability of the ontology, the DL reasoner was consistently active throughout the stages of the ontology development. Hence, any inconsistency introduced can be normalized immediately without accumulating beyond tracking. Nevertheless, there are cases, where the development have to backtrack and restore to an earlier consistent version of the ontology — especially in the early stages. A good advice to ensuring a consistent ontology is not only to keep the reasoners active for every development session but also to use the available interchangeably to utilize their various strengths in achieving consistency.

5.3 The Onto-CropBase Evaluation

In order to measure how well an information retrieval system meets the requirement needs of its end-users, a series of formal and informal evaluations are necessary. We briefly discussed some of these evaluation strategies as compared to the Onto-CropBase tool's functionalities. Moreover, being an interactive gateway to relatively new sets of crops data, where the stakeholders may include both experts and novice users, there is a need to ensure adequate usability of the Onto-CropBase tool with end-user satisfaction. In this section, we discuss the evaluation framework for our tool and present the preliminary evaluation results for the prototype.

As highlighted in [194], usability of semantic search engines such as the Onto-CropBase, can be evaluated based on the search query input types – such as natural language, view-based or facets, as well as type of end-users involved in the evaluation process, i.e. novice or expert users. The onto-cropbase being ontology-based and with natural language hierarchical presentation of search result offers more user friendly queries. This is due to the guidance provided by the users as the hierarchical data allows users to understand the structure of the data they are exploring. Moreover, it allows users to input queries using keywords or sentences as guided in the controlled natural language vocabulary – the ontology.

Performance Measurements for Semantic Search Engines Various methodologies exist for evaluating the efficiency of large Information Retrieval Systems (IRS) and commonly used evaluation strategies includes the Cranfield Paradigm [195], The TREC (Text Retrieval Conference) series [196]. While the Cranfield experiments advocates the use of a benchmark called test collection, with a set of search tasks and relevant judgements to measure the search results based on the retrieved lists as compared to the original document collection. The TREC series among others uses a set of standardized approaches such as interactive or user-based retrieval, precision and recall, tasks-specific evaluation, as well as domain-specific search evaluations, among numerous tracks. The domain-specific evaluation – Dynamic domain track is a subject of interest for the future evaluation of the Onto-cropbase. However, at this stage, a more generic semantic search engines evaluation methodology as suggested by Elbedweihy et al. in [197] can be applied in which the focus is on evaluating the following criterions: 'Query expressiveness', 'Usability' — including 'user satisfaction' and 'efficiency', 'scalability', 'quality of documentation', and 'search performance' — including speed of execution, precision and recall.

In the case of the Onto-CropBase tool, these suggestions were applied in the course of two stages namely, (i) the Comparison evaluation, where the tool was evaluated based on its available features and components as compared to existing system and (ii) the Performance evaluation by domain experts to test the usability of the tool. In the first case, the Onto-CropBase was evaluated conventionally, by considering its pre-designed requirements and further comparing it with similar tools that are considered successful in the related field, as discussed in the following scenarios. In the second case however, being a prototype version, the performance testing of the Onto-CropBase tool follows a flexible usability evaluation approach, where the evaluation success is measured entirely based on the end user's perception and ability to effectively utilize the tool – something difficult to be accurately measured.

5.3.1 Domain Experts Evaluation

When employed as evaluators in the usability evaluation of domain-specific knowledge systems, domain experts can provide excellent insights into the effectiveness of a knowledge model [198]. As such, domain experts walk-through's were employed (in a natural setting) as part of the Onto-CropBase tool's evaluation process. Unlike in a uncontrolled usability testing condition, the 'natural setting' allows users (testers) to freely interact with the system as they would in real-life.

As earlier mentioned, the Onto-CropBase tool has been primarily designed to be used by users seeking first-hand information on underutilized crops, rather than for software agents. To evaluate the suitability of the Onto-CropBase tool and the quality of information provided by the ontological knowledge system, two approaches were considered. The first approach employ the services of domain experts — including those actively involved in the domain knowledge modeling while the second test condition involves novice users. In either case however, some level of technical expertise is recognized, as all participants are familiar with traditional search engines. An average usability evaluation session of 15 minutes is designated per user with the first 5 minutes spent introducing the tool and the possible query patterns it accepts (keywords, concept names or short sentences containing relevant terms in the ontology). The next session allow users to carry out tasks without any help from trainers and in the end the users were asked to express their concerns or satisfaction verbally or by filling out the usability test form as shown in the Appendix — Section A.2.

5.3.1.1 Tasks and Queries

Information searching in the Onto-CropBase tool basically involves natural language concept searching, where users can simply type sentences containing search keywords and the query engine searches the linked data (the RDF ontologies) for matching concepts and their related entries. As shown in the search results captured in Fig. 4.12, test queries ranges from domain-specific keywords, to short natural language sentences containing relevant terminologies. Moreover, where users were allowed to take imaginative steps to explore the Onto-CropBase tool. However, the following basic tasks were selected both for the novice and expert users:

(i) *Find all underutilized crops available in the UC-ONTO* — A simple task to explore the different ontological datasets using one keyword 'underutilized'.

(ii) *Find the details of the Bambara groundnut information* — This task is aimed at testing the users' understanding of the results presentation flow from the search engine. As search results are presented as list of subjects with clickable links, the user is expected to simply click a subject to reveal its detailed information.

(iii) *Find and compare the Optimum germination temperature of Moringa and Bambaragroundnut* — While this task involves two different datasets and may be tricky even to technical users, a simple query consisting of the keyword 'optimum' followed by the name of the crops will suffice to get the relevant search results. Thanks to the 'concept search' feature of the Onto-CropBase semantic search tool.

Such basic tasks were performed during the evaluation process to test usability of the tool as well as its effectiveness in answering user queries. However, the following list describes the basic assumptions upheld during the usability evaluation of the Onto-CropBase prototype:

Assumptions In line with the general procedure of software evaluation paradigm and in order to establish an effective scope that will simplify the evaluation process, some of the following facts were presumed regarding the onto-CropBase tool:

- Users: both novice and experts are assumed to have basic understanding of information searching using computers.
- Only OWL ontologies (UC-ONTO) and local RDF datasets stored in our local host server were used as the test data with no external RDFS or remote Sparql end-points.
- Location data is subject to the accuracy of latitude and longitude supplied in the knowledgebase.
- Due to the size and restrictive nature of the vocabulary involved, the automated evaluation such as the speed of query execution, precision and recall are assumed to be satisfactory. As such no detailed metrics are important in this stage.

5.3.1.2 Data Capturing

In order to collect user responses, a usability test form is provided as an option. In other cases, especially with the domain experts, a verbal interview is recorded based on the task and questions provided in the usability test form — see Section A.2 of the Appendix. As shown in the form, a simple demographic data is collected to identify the user expertise, position, department and/or specialization. This helps to identify the users as either underutilized-crops domain experts or as domain novices. The next section of the form explains the tasks to be undertaken during the testing process as described earlier in Section 5.3.1.1. This is followed by the acceptability measurement where a simple scoring scale is used to measure how well the users agree or disagree to the following statements regarding the components and functionalities of the Onto-CropBase tool:

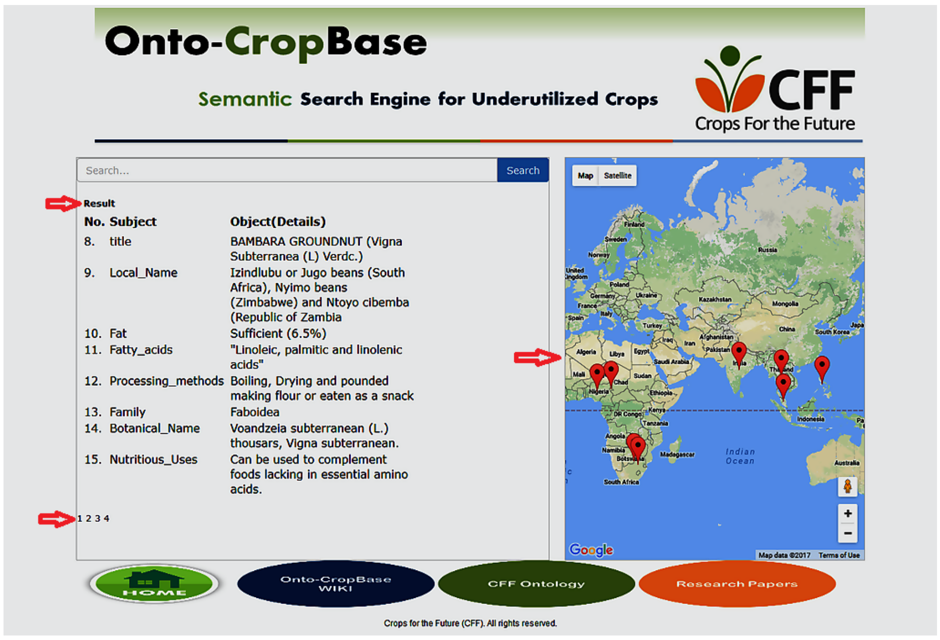
- Domain coverage and Datasets are sufficient.
- Information presented is relevant.
- Information presented is useful and relatively accurate.
- Searching (Query) style is clear and easy to use.
- User Interface (labels, fonts, instructions) is clear and Informative.
- Search results presentation is understandable.
- Navigating the search tool and results is easy.

Lastly, the domain experts were asked to identify specific features or components as liked-features, the disliked features and additional features, if any, they would like to see in the next version of the system.

Even though there were no time limits during the experts' evaluation process, the domain experts identified some usability problems and offer few design suggestions. However, much insights were provided into the structure of the knowledge presented by the search engine as well as comments on the validity of the information delivered. Nevertheless, the domain experts' suggestions, such as those recorded in Table 5.2, were considered highly critical and therefore given much priority in the subsequent design and development. More detailed evaluation results are discussed briefly in the following sub-headings:

5.3.1.3 Onto-CropBase Usability — Labels, Functionality, Navigation and Visual Outputs

From the summary of responses, all five domain experts and seven novice users involved agreed on the question that the information provided by the tool is 'relevant' and that the tool is 'simple' to use. In essence, the usability of the Onto-CropBase tool was found to be acceptable even for non-technical experts. Similarly, almost all the end users exposed to the Onto-CropBase tool were able to freely navigate its functionalities without much intervention.



Result

No.	Subject	Object(Details)
8.	title	BAMBARA GROUNDNUT (Vigna Subterranea (L) Verdc.)
9.	Local_Name	Izindlubu or Jugo beans (South Africa), Nyimo beans (Zimbabwe) and Ntoyo cibemba (Republic of Zambia
10.	Fat	Sufficient (6.5%)
11.	Fatty_acids	"Linoleic, palmitic and linolenic acids"
12.	Processing_methods	Boiling, Drying and pounded making flour or eaten as a snack
13.	Family	Faboidea
14.	Botanical_Name	Voandzeia subterranean (L.) thousars, Vigna subterranean.
15.	Nutritious_Uses	Can be used to complement foods lacking in essential amino acids.

1 2 3 4

Crops for the Future (CFF). All rights reserved.

Figure 5.19: The Onto-CropBase interface showing the search results navigation and Map interface

Moreover, the user feedbacks highlighted some of the 'most-liked' features of the tool to be: *simple search interface, familiarity of search language, the map feature, grouping and paging of search results* among others (See Fig. 5.19. Whereas some features were not favored especially by the expert users as they complaint against the *restrictive nature of the search domain, lack of look-up queries, and storage of query results, lack of ordering os search results* and *lack of image to show concept hierarchies*. These sentiments are further summarized in Table 5.2 and categorized according to the relevant component of the Onto-CropBase for clarity and further actions.

5.3.2 Comparison with Other Semantic Search Engines

As part of the preliminary validation and to ensure that desired functionalities are provided by the Onto-CropBase tool, we compare its features with those of existing ontology-based knowledge systems. For effective validation, we focus on relevant existing systems from the field of life-sciences, such as the Crop Ontology curation tool [52], and Semantic Faceted Search Engine (SemFacet) [133].

Onto-CropBase Component	**Approved Features**	**Dislikes**
Domain coverage and Datasets	Precision of facts	Restrictive nature of the search domain
Input and Query Style	Natural language search	No look-up for frequent searches
User Interface and Input	Simplicity of Search Interface	—
Query Results and Presentation Style	The Map interface	No image presentations
	Page grouping of search results	Unordered Query results

Table 5.2: User Perceptions of the Onto-CropBase Features

The comparison is presented in Table 5.3. It shows that our Onto-CropBase tool's basic functionalities, such as the concept-based search and navigational presentation of search results, are relevant and effective approaches as employed in those systems. In addition, the map-view feature offered by the Onto-CropBase tool, its simplicity, as well as the highly precise search results are commendable for the semantic search system. This is partly, due to the domain-specific ontologies employed and the fact that the global ontology (UC-ONTO), which guides the conceptual definitions is also purposely and carefully developed for the Onto-CropBase tool.

Features	**Onto-CropBase**	**Crops Ontology** [52]	**SemFacet** [133]
Knowledge Domain	Crops (Specific)	Crops (Generalized)	Life-Sciences
Search Type	Keyword + Concept	Keyword + hierarchy navigation	Keyword + Faceted navigation
Ontology Format	OWL2 + RDF	OBO	OWL2 + RDF
Result Visualization	Text + Map	Text + Image	Text + Image
Data Curation	None	Keyword + hierarchy	None

Table 5.3: Comparison of features for Onto-CropBase, Crop Ontology and Semfacet.

However, we observed the need to provide a visualization of hierarchical relations between concepts to enhance the information presentation— a need already mentioned by some of our end users. Another useful functionality not currently covered by our tool is the implementation of an ’ontology-based faceted search mechanism’. This is important in ontology-based searches as it helps to portray the inner structure of the ontology as a guide for efficient probe of the knowledgebase. Nonetheless, we considered these functions to be of lower priority as compared to the domain-specific integration of the ontological knowledge models and the current features of the Onto-CropBase search engine. Though they may well be included in the future releases of the tool.

5.3.3 Scalability of the Onto-CropBase Tool

In terms of scalability, the Onto-CropBase being a linked-data based knowledge system is by nature scalable. This is due to the fact that the SPARQL federated queries, which involve all available components of the knowledgebase represented by their ’prefixes’, are designed to be explored for semantically related concepts. As such only a search term that have information in all component ontologies will result in data generated from the entire ontologies. While the searching always goes through all available knowledge sources thereby making the average search time to be the same for all queries, the search results however depends on the universality of the search concept in the knowledge base. That is, where a search term is contained in many linked ontologies, a larger search result is presented. Whereas, if the search term only involves few linked data then a smaller search result is presented. As such, typical search results (as shown in Figures 4.11, 4.12, etc) shows the inherent scalability function of the Onto-CropBase search engine, whereby large result-sets were presented in cases where the ’search concept’ involves querying larger ontologies, and vice-versa.

5.4 The Fuzzy-Temporal SWRL Model Evaluation

5.4.1 FT-SWRL Ontology Modularization, Depth and Structure

From the conceptual hierarchy of the FT-SWRL ontology, depicted in Fig. 5.20 below, it can be seen that the ontology model is highly modularized to enable efficient representation of time domain as well as representing imprecision in time expressions. The FT-SWRL model defines fuzzy temporal concepts and how they relate to each other and the temporal facts that can utilize them from a domain ontology. It defines specialized ontologies such as, *Temporal Concepts, Temporal propositions, Resources, Fuzzy Temporal Concepts, Fuzzy Counts, Fuzzy Granularity,* etc. These were structured together to form a larger ontology fuzzy-temporal model for representing temporal uncertainties in domain ontologies such as the UC-ONTO. Individual instances added to the FT-SWRL ontology to represent indivisible fuzzy-temporal objects (as shown earlier in Fig. 4.14 includes *about, approximately, few, around, before, etc.* among others. Natural language comments and annotations were provided for the concepts to serve as first-hand documentation for ease of use of the FT-SWRL ontology model as exemplified in Fig. 5.22.

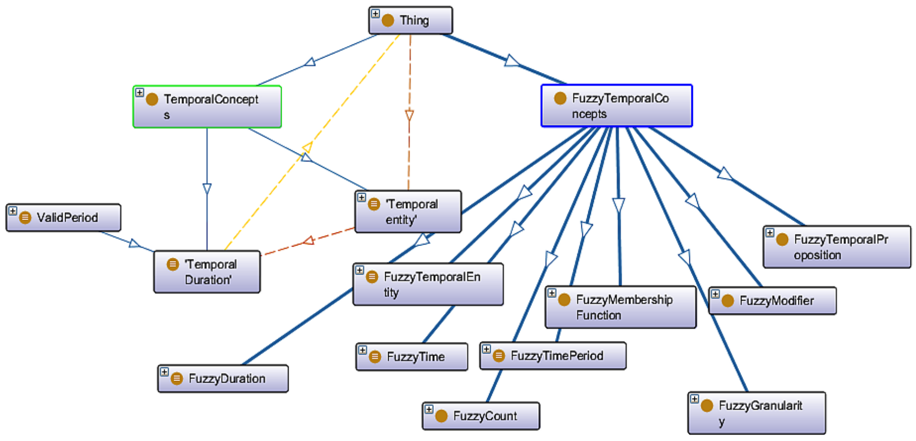


Figure 5.20: A conceptual hierarchy visualization of the FT-SWRL Ontology.

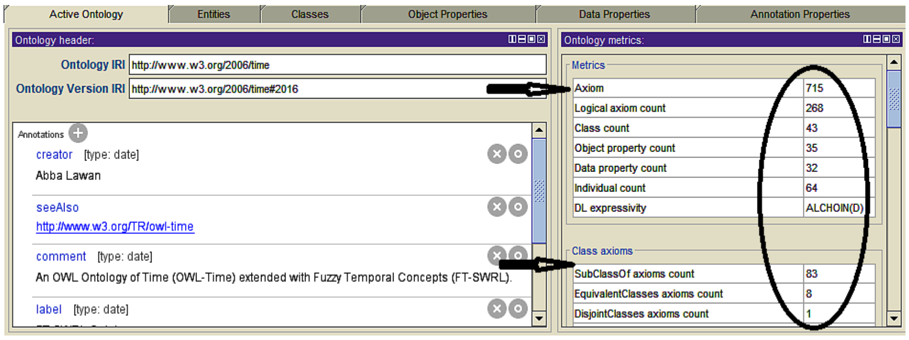
AS the FT-SWRL model is built to incorporate a time ontology model, it goes deep and wide in representing the time domain as well as the fuzzy-temporal domain. Two main classes in the ontology are the Temporal Concepts and the Fuzzy Temporal Concepts. As imprecise temporal expressions can not be explicitly represented without first representing the temporal domain, the time ontology [151] structure was adopted to serve as the starting point for the temporal concept definitions. This is further extended to incorporate the Date-time description as well as Valid time description leading to the formulation of classes such as the Temporal Duration class, which defines the time expressions that incorporates granularities (e.g. 2 weeks) and Temporal Entity — which have Time Instants (e.g. 2:00 pm) and Time Intervals such as 13:00 -14:00.

Similarly, the Fuzzy Temporal Concepts class contains such sub-classes as Fuzzy Temporal Entity, which defines Fuzzy Time Interval and Fuzzy Time Intervals. Other classes include the Fuzzy Modifier class, which holds the imprecise temporal expressions (E.g. a few, about, around, etc) that are often employed to modify a simple definite temporal statement into an imprecise one. Furthermore, the Fuzzy Count and Fuzzy Granularity classes were defined to contain those imprecise granularities such as always, cycles, every-time, etc. Another important class is the Fuzzy Membership Function which contains the mathematical model representation of members of the Fuzzy Modifier class, including their Membership Function Name (mfName) and Membership Function Curve (mfCurve).

5.4.2 FT-SWRL Ontology Metrics

As described in the UC-ONTO evaluation in Section 5.2.2, a look at the ontology metrics can help to ascertain the expressiveness, number of concepts, individuals, data values and the relationships contained in an ontology. As shown in Fig. 5.21, the FT-SWRL ontology have over 700 axioms with 43 different classes, 67 different object and property counts as well as 64 individual assertions. Also similar to the UC-ONTO there is a high ratio between the concepts and axioms count, which shows that the ontology is logically sound beyond simple list of terminologies.

The expressiveness metric of the FT-SWRL ontology borders around the $\mathcal{ALCH}$ $\mathcal{OIN}(\mathcal{D})$ algorithm. This shows that the ontology is moderately expressive. The



Ontology IRI: http://www.w3.org/2006/time
Ontology Version IRI: http://www.w3.org/2006/time#2016

creator [type: date]
Abba Lawan

seeAlso
http://www.w3.org/TR/owl-time

comment [type: date]
An OWL Ontology of Time (OWL-Time) extended with Fuzzy Temporal Concepts (FT-SWRL).

label [type: date]

Axiom	715
Logical axiom count	268
Class count	43
Object property count	35
Data property count	32
Individual count	64
DL expressivity	ALCHOIN(D)

SubClassOf axioms count	83
EquivalentClasses axioms count	8
DisjointClasses axioms count	1

Figure 5.21: FT-SWRL Ontology Metrics showing Expressivity, Axioms, classes and property count.

expressiveness equivalence of the $\mathcal{ALCHOIN}(\mathcal{D})$ algorithm implies that the FT-SWRL Ontology model consists of an 'Attributive Language with Complement $\mathcal{ALC}$' with the Role and Hierarchies $\mathcal{H}$. This is extended with expressiveness of enumerated classes $\mathcal{O}$, inverse role assertions $\mathcal{I}$, cardinality restrictions $\mathcal{N}$, and data type assertions $\mathcal{D}$. The OWL expressiveness algorithm is discussed in details in Section 2.2.3.2.

5.4.3 FT-SWRL Model Built-ins Usability and FT-SWRL Rules Format

As described in [186], user-defined SWRL built-ins can be used directly in SWRL rules. However, in order to use them and their extensions, such as the SWRL-FT built-ins, they need to first be imported into the main ontology by importing their definition — in this case, the SWRL-FT ontology. Final implementation of the SWRL-FT ontology requires defining the fuzzy temporal built-ins as instances of the original *swrl:Builtin* class. This is followed by their corresponding java implementations through the *SWRLBuiltInBbridge* [3]. The SWRLBuiltInBridge is a component of the SWRLTab (available in the protégé ontology editor) that allows the manipulation of SWRL built-ins using Java. Relevant built-ins are usually grouped together in a single OWL file — which can then be imported into any domain ontology for utilization. Considering the scope and goal of this thesis, which is to define an extended ontology model for representing a Fuzzy-

[3]http://protege.cim3.net/cgi-bin/wiki.pl?SWRLBuiltInBridge#nid88T

Temporal expressions, the final implementation stage is left for further exploration and evaluation in the future works as discussed in Section 6.2.

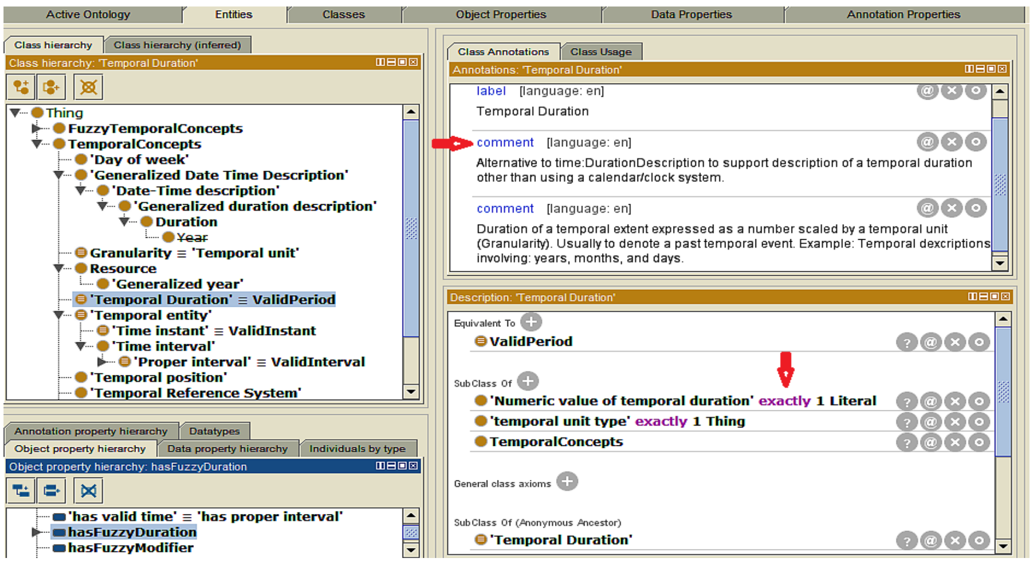
Nevertheless, a detailed discussion on the fuzzification of the fuzzy temporal built-ins is presented in Section 4.4.3. However, in what follows we give some example FT-SWRL model classes and properties in Table 5.4 to highlight possible usability of the built-ins library. This is followed by example FT-SWRL rules to explain the formats by modeling some of the imprecise temporal facts on Bambara groundnut crop as highlighted in the FT-SWRL model motivation in Section 3.5.1.

Fuzzy Temporal Class	**Sub-classes**	**Example Terms (ITEs)/ Phrases**
FuzzyTemporal-Proposition		Event time (e.g. hasGermination-Time), Date of Events (e.g has-DateOfSowing), etc.
FuzzyTime	FuzzyTimeInstant FuzzyTimePe-riod	Several hours, within three weeks, This Weekend, before 2:00pm, etc.
FuzzyCount		Cycles, Next, Previous, this, Several, within, etc.
FuzzyGranularity	SetGranularity DateGranularity	Weeks, Fortnight, Noon, Quarter, Weekend, etc. (granularities not defined in the Granularity class)
SetGranularity		Yearly, Monthly, Daily, Weekly, Hourly, Per-minute, per-second, etc.
DateGranularity		Ago, Earlier, Lately, Nowadays, Recently, Past, Since, Until, etc.
FuzzyModifiers		about, around, within, a few, approximately, before, after, during, etc.
MembershipFunction	mfName mfCurve	Bell, Gaussian, Sigmoid, Z-function, etc.

Fuzzy Temporal Class	Sub-classes	Example Terms (ITEs)/ Phrases

Table 5.4: Example of Fuzzy Temporal SWRL Entities

The Table above shows the FT-SWRL entities for handling imprecise temporal expressions (ITEs), which is a pioneer reference model for representing fuzzy temporal domain knowledge in OWL ontologies. As the model is aimed at modeling those ITEs that are common to all knowledge domains and due to the reusable nature of ontologies, as discussed in details in Section 2.3 of the literature, the FT-SWRL model is thus applicable as a reusable reference model to all and not just the Underutilized Crops domain.


Active Ontology
Entities
Classes
Object Properties
Data Properties
Annotation Properties
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: 'Temporal Duration'
Thing
FuzzyTemporalConcepts
TemporalConcepts
'Day of week'
'Generalized Date Time Description'
'Date-Time description'
'Generalized duration description'
Duration
Year
Granularity ≡ 'Temporal unit'
Resource
'Generalized year'
'Temporal Duration' ≡ ValidPeriod
'Temporal entity'
'Time instant' ≡ ValidInstant
'Time interval'
'Proper interval' ≡ ValidInterval
'Temporal position'
'Temporal Reference System'
Annotation property hierarchy
Datatypes
Object property hierarchy
Data property hierarchy
Individuals by type
Object property hierarchy: hasFuzzyDuration
'has valid time' ≡ 'has proper interval'
hasFuzzyDuration
hasFuzzyModifier
Class Annotations
Class Usage
Annotations: 'Temporal Duration'
Description: 'Temporal Duration'
Equivalent To
ValidPeriod
SubClass Of
General class axioms
SubClass Of (Anonymous Ancestor)

label [language: en]
Temporal Duration

comment [language: en]
Alternative to time:DurationDescription to support description of a temporal duration other than using a calendar/clock system.

comment [language: en]
Duration of a temporal extent expressed as a number scaled by a temporal unit (Granularity). Usually to denote a past temporal event. Example: Temporal descriptions involving: years, months, and days.

'Numeric value of temporal duration' exactly 1 Literal
'temporal unit type' exactly 1 Thing
TemporalConcepts

'Temporal Duration'

Figure 5.22: A conceptual hierarchy from the Protégé ontology editor showing fragment of FT-SWRL Ontology.

As highlighted in Fig. 5.22, natural language definition of the concepts were provided as annotations and comments to aid users in understanding the FT-SWRL model. More example use of the fuzzy temporal SWRL concepts is highlighted in the sample FT-SWRL rules below:

1. ”Bambara beans take around 1 to 2 weeks to germinate”

   BambaraBeans (?bb) ∧ *hasGerminationTime(?bb, ?gt)* ∧ *fuzzytemporal:**around**(?gt,* *'2', temporal:weeks) ⟶ GerminationPeriod(?bb, True).*

2. "Bambara beans germinate within 15 days from the date of sowing"
   *BambaraBeans (?bb) ∧ hasDateOfSowing(?bb, ?dos) ∧ fuzzytemporal: **within**(?dos, '15', temporal:days) ⟶ GerminationPeriod(?bb, True)*

3. "Bambara beans will germinate in few weeks if moderate rainfall continues"
   *BambaraBeans (?bb) ∧ hasModerateRainfall(?bb, ?mdR) ∧ hasGerminationTime(?bb, ?gt) ∧ fuzzytemporal:**few**(?gt, temporal:weeks) ⟶ GerminationPeriod(?bb, True)*

4. "Seeds stored for about12 months germinate well, but longer storage results in loss of viability"
   *Seed (?s) ∧ hasStorageTime(?s, ?st) ∧ fuzzytemporal:**before**(?st, 12, temporal:months) ⟶ GerminationPeriod(?bb, True).*

   *Seed (?s) ∧ hasStorageTime(?s, ?st) ∧ fuzzytemporal:**after**(?st, 12, temporal:months) ⟶ GerminationPeriod(?bb, False)*

5.5 Discussion and Summary

In the first major section (5.2) of this chapter, an ontological knowledge model for representing underutilized crops information using OWL and SWRL rules is evaluated. The framework of the model, presented in Section 3.2, describes ontology (UC-ONTO) engineering and standardization approaches as well as the integration of OWL ontologies with rules for added expressiveness. Regarding the techniques for ontology generation, previous chapters have explored the methodologies and experimentations that highlights the process of creating ontologies from scratch, from other heterogeneous data sources, through competency questions, as well as the benefits of ontology reuse — as shown in Section 4.2.1. The practice of generating domain-specific concepts through competency questions is highly emphasized with the detailed methodology and practice presented in the preceding chapter. Whereas the validation of ontologies using the competency questions is discussed in this chapter and shown to be a good source of competency queries for evaluating the contents and structure of ontologies as well as a good foundation for rule formation. Ontology standardization is as important as the knowledge acquisition process. Different standardization approaches were implemented including alignment, merging and natural language annotations of domain concepts.

Based on the reviewed literature, experimentation and validations of the rule assertions of UC-ONTO, SWRL Rules are shown to be necessary in OWL ontologies for expressing complex scenarios and modeling user queries to utilize the ontological knowledge base. This is chiefly due to the semantic compatibility between the two languages. Using DL-queries and Reasoner inferences, Section 5.2.4 of this chapter presented an evaluation that consists on querying the knowledge base to check that the query results are consistent with the added rules. This includes validation of the ontology and SWRL rules by writing appropriate DL queries. In conclusion, ontology development in the agricultural domain no doubt requires knowledge reuse of existing upper domain ontologies. Where the intention is to utilize the knowledge base for decision support, competency questions positively influences the knowledge elicitation, modeling process, as well as the validation of ontologies.

Sections 3.4 and 4.3 presented the methodology and case study of the Onto-CropBase semantic search engine respectively and Section 5.3 — of this chapter describes the evaluation process of the tool. The Onto-CropBase tool demonstrates an ontology utilization approach, which is semantic web application for browsing the underutilized-crops ontology (UC-ONTO) extended with relevant linked data in RDF. Basic tool's functionalities provided a web-based user interface with a search engine for querying the ontology-based knowledge model. At the center of the knowledgebase, is an OWL 2 ontology, serving as a *global ontology* containing shared concepts from integrated local ontologies or RDF datasets. The functionalities of Onto-CropBase as validated through domain experts and in comparison with relevant tools, provided for ease of use and help in asserting the effectiveness of the Onto-CropBase as an ontology-based search engine. With a simple search space, click-able links for exploring the results and page-wise presentation of the search details with informative captions and labels, gives the tool a natural feel-and-touch of a search engine. In essence, the preliminary evaluation of the tool, shows it can serve as first-hand information portal for basic facts on underutilized crops – one basic requirement of the thesis. Though, there are still open issues to be considered in the future as discussed in the 'limitations and future works' in Section 6.2 of the following chapter.

As the effectiveness of the Onto-CropBase search results depends entirely on the data source (UC-ONTO), which in turn depends on the expressive powers of the OWL/SWRL language (see Section 2.2.3.2), a language extension of the SWRL formalism was proposed in Section 3.5 with a case study example implemented in Section 4.4, and subsequently the ontology model evaluated in Section 5.4 of this chapter. The FT-SWRL model provides a knowledge representation formalism for managing temporal imprecisions in domain knowledge. With the current OWL/SWRL combination, temporal information can only be stored by associating each piece of knowledge with a fixed time stamp. However, when capturing domain expert's narratives involving imprecise temporal expressions, it becomes imperative to associate some sense of vagueness to the captured temporal facts for accurate representation of the domain knowledge. Due to the expressive limitations of the current OWL/SWRL formalism, time-dependent vague expressions of domain facts cannot be captured into the UC-ONTO. Numerous extensions were reviewed in Section 2.9 and yet imprecise temporal extension of SWRL is missing. To this end, the new language extension of SWRL, the FT-SWRL model was proposed and the ontology model implemented. The ontology defines the required language terminologies and built-ins to serves as reference standard for fuzzy temporal modeling. The validation process in Section 5.4 also discusses the fuzzification process (with examples) of some of the newly defined built-ins to support their semantic evaluation using carefully designed fuzzy membership functions and inference rules. Furthermore, example FT-SWRL rules were also presented to demonstrate the rules format. In conclusion, considering the depth, modularity and size of the proposed FT-SWRL model, it can thus serve as a formal specification for handling imprecise temporal expressions in OWL ontologies.

The following chapter presents the thesis conclusions, summarizing the major contributions, their limitations and suggestions for improvement as future works.

Chapter 6

Conclusion

The thesis presented a study on ontology-based knowledge systems and explored the approaches and practice of semantic knowledge modeling towards an ontology-driven CropBase knowledge system. The project involves three major studies in the areas of ontology modeling, ontology utilization, and ontological language extensions. In the area of ontology modeling, a framework for representing domain knowledge using OWL and SWRL rules was presented leading to the development of a pioneer SWRL-enabled Underutilized Crops Ontology (UC-ONTO) in OWL. In the area of ontology utilization, an ontology-based semantic search engine was proposed leading to the realization of the Onto-CropBase semantic search tool that utilizes the UC-ONTO as a knowledge base. While in the ontology language extension, the expressive limitations of the OWL/SWRL combination were extended with the proposal of the Fuzzy Temporal Semantic Web Rule Language (The FT-SWRL model). This leads to the development of the first fuzzy temporal SWRL ontology that defines the OWL entities and built-ins for representing imprecise temporal expressions in ontologies.

The motivations of the thesis and research context were introduced in Chapter 1, followed research questions, aims, and objectives among others. While in Chapter 2, the literature behind the semantic web technologies and the related methods in ontology-based knowledge modeling were explored in details. Ontology-based knowledge modeling task was categorized into three: ontology engineering from domain facts, ontology standardization for improvement, and extending ontologies with the expressiveness of logic programming rules for modeling complex domain knowledge. Chapter 3 discusses the research methodologies highlighting the ontology engineering methods for the UC-ONTO, the Onto-CropBase design and development approaches as well as the FT-SWRL model design approach. Chapter 4 then presented the case studies on the implementation of the UC-ONTO, the Onto-CropBase and the FT-SWRL model highlighting the experimentation details and implementation experiences. Chapter 5 presents the analysis and discusses the evaluation of the case study results.

In what follows, the remainder of this chapter summarizes the major contributions of the thesis and their technical limitations followed by suggestions for improvement. Lastly, the possible directions for further research were discussed.

6.1 Summary of Major Contributions

The initial contribution, mainly presented in Sections 3.2, 4.2.1 and 5.2, involves the development of the first OWL-based, SWRL-extended underutilized crops ontology (UC-ONTO) [184], which serve as a formal specification of facts representing the underutilized crops domain knowledge. Notable design choices involves the use of competency questions for domain knowledge gathering and conceptualization. With regards to the ontology development methodology, we recommended the tasks of *'ontology versioning* and *assembly'* to be considered during ontology development. The ontology versioning helps to manage the problem of tracking ontology changes, while ontology assembly describes how those changes can best be assembled into a coherent domain specification. Following investigations of the expressive capabilities of the OWL ontology language, it was concluded that the ontology development requires the integration with logic programming (LP) rules for added expressiveness. This thesis shows how the semantic web rule language (SWRL), as an examplar LP rule, can be employed to extend the OWL ontologies. Based on our observation however, the current usability limitations of SWRL rules and its expressiveness extensions may be due to the following reasons: (i) Lack of readily-available reasoners that can draw inference on the SWRL extensions within decidable portion of the language. (ii) Modeling real-world problems using SWRL is still at the research stage, with little large-scale developmental efforts. (iii) Furthermore, the availability of alternative rule and query languages such as SPARQL and RuleML that are able to achieve (though usually in a turnaround fashion), some of the current limitations of the SWRL formalism.

Another major contribution, presented in Sections 3.4, 4.3 and Section 5.3, deals with the issue of ontology utilization through ontology-based semantic searching. While successful ontology development efforts helps to strengthen the semantic knowledge modeling and presentation, succeeding challenges involves the retrieval and dissemination of the ontology knowledge to non-technical stakeholders for informed decision-making. The thesis addresses this challenge through the Onto-CropBase tool, a flexible semantic web application for exploring the underutilized-crops ontology (UC-ONTO) — integrated with other relevant crops datasets in RDF. The tool provides a web-based user interface with a search engine for querying ontology-based knowledge models. At the center of the knowledge-base, is the OWL ontology (UC-ONTO), serving as a *global ontology* containing shared concepts from the integrated local ontologies (the linked RDF datasets). Preliminary evaluations confirms that the Onto-CropBase can serve as a first-hand information portal for information on underutilized crops and their products. Furthermore, the tool was also found to be usable even to non-technical or domain experts.

From the ontology utilization approaches, the Onto-CropBase implementation and usability assessment, it is clear that ontology standardization is as important as the knowledge acquisition process itself. This leads to the investigation of various ontology standardization approaches as discussed in Section 2.5.4. The UC-ONTO standardization approaches recommends the domain experts' validation of fact assertions, alignment with upper level ontologies as well as an ontology standardization approach through Competency Questions posed to the ontological knowledgebase, to ascertain its quality of response as compared to the domain experts' opinion — see Section 5.3..

The limitations of the Web Ontology Language (OWL) and the Semantic Web Rule Language (SWRL) for added expressiveness leads to the study of the SWRL's expressiveness extensions presented in Section 2.9. This in turn leads to the next major contribution of the thesis, which involves the engineering of the pioneer fuzzy temporal extension of SWRL (the FT-SWRL model) for managing temporal uncertainties commonly found in domain knowledge. This is particularly useful in order to capture the true nature of domain processes and domain experts' narratives, which are usually imprecise and time-dependent. Therefore, Sections 3.5, 4.4, and 5.4 presented the FT-SWRL model design approach, the case study implementation, and the evaluation of the model respectively. The fuzzy temporal extension to the Semantic Web Rule Language (SWRL) combines fuzzy theories based on the valid-time temporal model to provide a reference model for managing imprecise temporal expressions in OWL ontologies. FT-SWRL extension introduces a fuzzy temporal model for the semantic web, which consists of the FT-SWRL ontology and FT-SWRL built-ins for the syntactic and semantic definition of the fuzzy temporal concepts respectively. Reasoning paradigm for the FT-SWRL constructs is discussed, which involves the fuzzification of selected temporal expressions using carefully selected membership and weight functions followed by example rule assertions to highlight the FT-SWRL rules format. Considering the conceptual definitions, depth, modularity and expressiveness of the FT-SWRL ontology, led to the conclusion that the pioneered FT-SWRL model can serve as a formal specification for handling imprecise temporal expressions on the semantic web.

6.2 Limitations and Suggestions for Improvement

As both the ontologies and their representation formalisms are ever-evolving, it is believed that the works presented in this thesis are not the end but beginnings of more research in ontology-based knowledge systems. Moreover, the contributions also raised new questions in some cases and leave room for improvements in others.

From the reviewed SWRL expressivenes extensions for example, there is a need to further evaluate the practicalities of the theoretical SWRL extensions reviewed and further validate their decidability and inferences. There is also the need for a special fuzzy reasoner for checking the completeness of these theoretical fuzzy SWRL extensions. Standardization of these SWRL extensions is another research gap identified in the review, as some of the extensions can only be utilized in specific application domains while others were found to be generic. With the continuous development efforts on the expressive powers and usability of SWRL formalism, it can undoubtedly evolve into a complete ontology rules language for the Semantic Web community, one that is both comprehensive in its expressive powers and computationally viable in achieving inference.

From the Onto-CropBase tools point of view: Regarding the ontology-based semantic search tool, future efforts can focus on enriching the backend model with more crops-related RDF datasets and SPARQL endpoints, to allow integration of remote datasets for comprehensive search results. This will require migrating the Mediator component of the semantic search tool from the current 'Apache Jena' version to the new 'Apache Fuseki' version — which its developers claimed can deliver local ontologies as RDF data over http. Hence, automating the data integration process and further expanding the knowledgebase of the onto-CropBase search engine. Moreover, the user interface can be enhanced with navigation-hierarchy or *facets* to highlight the structure of the knowledge base for more efficient user queries. Lookup and comparison tables can also be designed to compare some of the domain facts, such as comaparing the nutritional info of underutilized crops with that of major crops. Lastly, a more specialized *location-based information* on underutilized crops can be added to highlight soil types and climate requirements, among other features.

From the Fuzzy Temporal SWRL (FT-SWRL) Model point of view: More imprecise temporal expressions need to be incorporated into the FT-SWRL model for comprehensiveness. A complete reasoning system for the FT-SWRL rules also need to be considered in order to achieve the required fuzziness in domain knowledge modeling. This can be achieved through modifying the existing 'SWRL-API Temporal model' to support the newly-defined fuzzy-temporal operators. Moreover, by leveraging the implementation mechanism for SWRL extensions, such as described in [199], the FT-SWRL model can be efficiently documented and fully realized to allow modeling and reasoning over imprecise temporal domain knowledge. Furthermore, a fuzzy temporal semantic web query language (FT-SQWRL) extension may as well be considered in the future for exploring FT-SWRL extended ontologies.

6.3 Concluding Remarks

From the thesis contributions, summarized in 6.1, most of the proposed research questions have thus been elicited. In particular, the UC-ONTO design, development methodologies and case study, affirms that ontologies can be effectively utilized in the agricultural domain as a formal specification of the domain knowledge. Similarly, the ontology Standardization approaches and results show that ’ontology reuses’ and their alignment with upper-level ontologies are particularly necessary for efficient modeling of the domain knowledge. Furthermore, the UC-ONTO’s extension with the expressiveness of SWRL rules, concludes that comprehensive domain modeling using OWL ontologies ultimately requires the declarative assertiveness of logic programming rules.

Furthermore, the FT-SWRL model particularly the Fuzzy temporal ontology, which provides the hierarchical definition of fuzzy temporal terms and their relationships, helps to conclude that OWL and SWRL formalisms as ontology development languages can be enhanced with additional expressiveness constructs — in this case, through the fuzzification of interval-based temporal logic, to effectively represent imprecise temporal expressions inherent in domain knowledge description.

In essence, the thesis findings conclude that ontologies can be employed to standardize domain knowledge representation and further shows that Semantic Web applications (such as the Onto-CropBase tool) can utilize such domain ontologies with relevant linked-data to provide efficient and reusable information systems for informed decision-making on Underutilized Crops and their related products.

Appendix A.

A1. More Results in Pictures

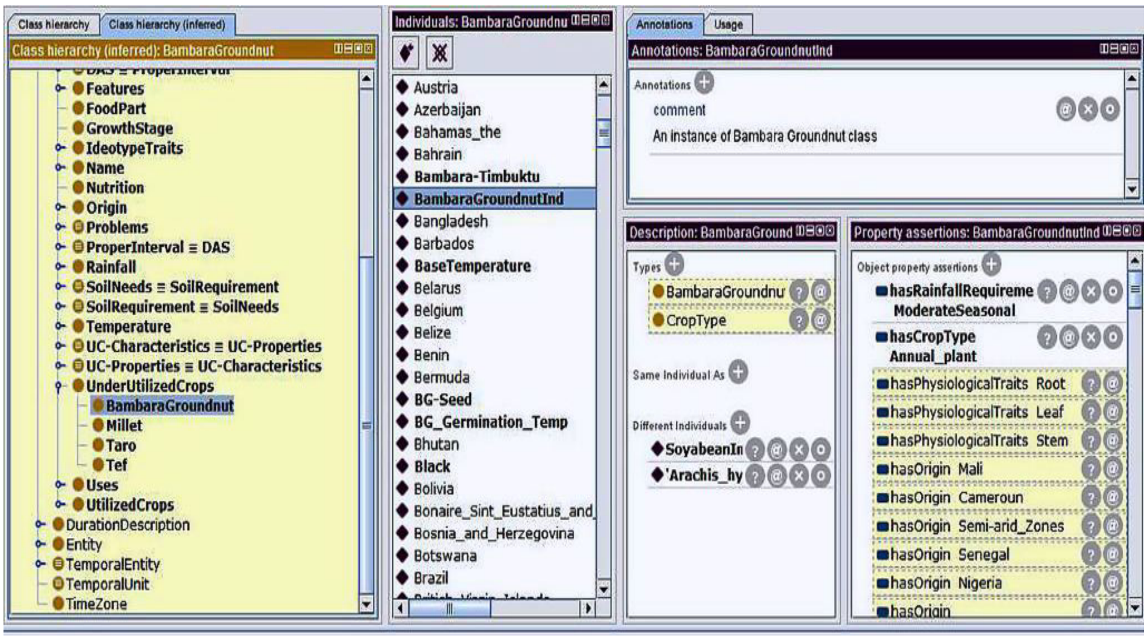
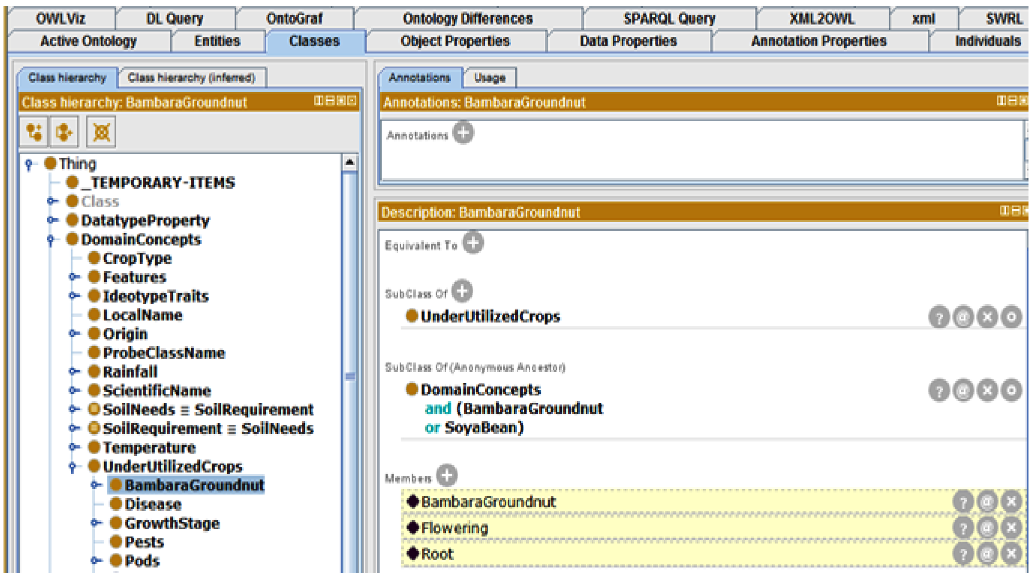


Figure 6.1: Bambaranut Instance (middle) with inferred class hierarchy (left) and Property assertions (bottom right)



Class hierarchy: BambaraGroundnut

- Thing
  - _TEMPORARY-ITEMS
  - Class
  - DatatypeProperty
  - DomainConcepts
    - CropType
    - Features
    - IdeotypeTraits
    - LocalName
    - Origin
    - ProbeClassName
    - Rainfall
    - ScientificName
    - SoilNeeds ≡ SoilRequirement
    - SoilRequirement ≡ SoilNeeds
    - Temperature
    - UnderUtilizedCrops
      - BambaraGroundnut
      - Disease
      - GrowthStage
      - Pests
      - Pods

Annotations: BambaraGroundnut

Annotations

Description: BambaraGroundnut

Equivalent To

SubClass Of

UnderUtilizedCrops

SubClass Of (Anonymous Ancestor)

DomainConcepts
and (BambaraGroundnut
or SoyaBean)

Members

BambaraGroundnut
Flowering
Root

Figure 6.2: Bambara groundnut assertion showing Anonymous ancestors and inferred class members

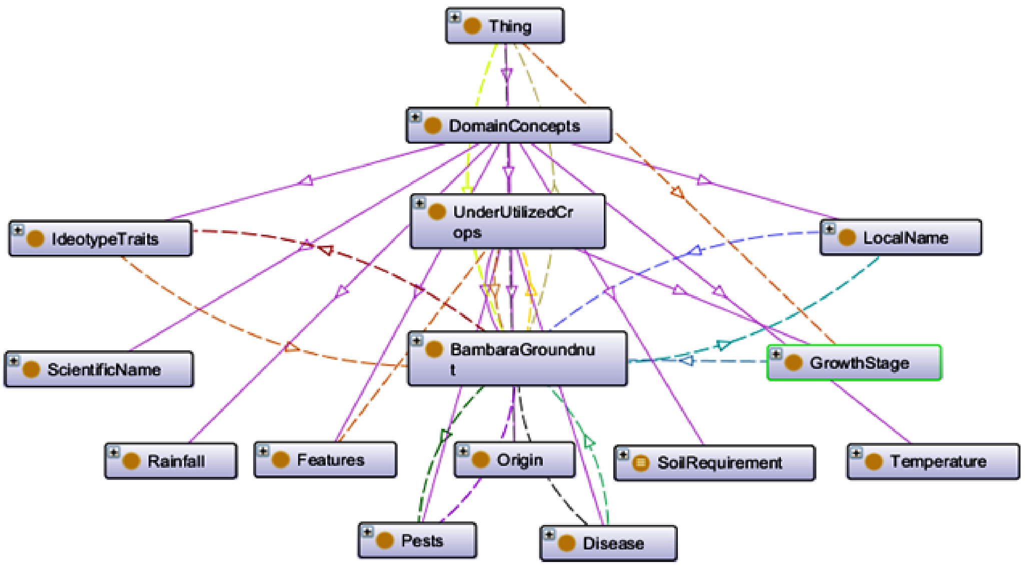
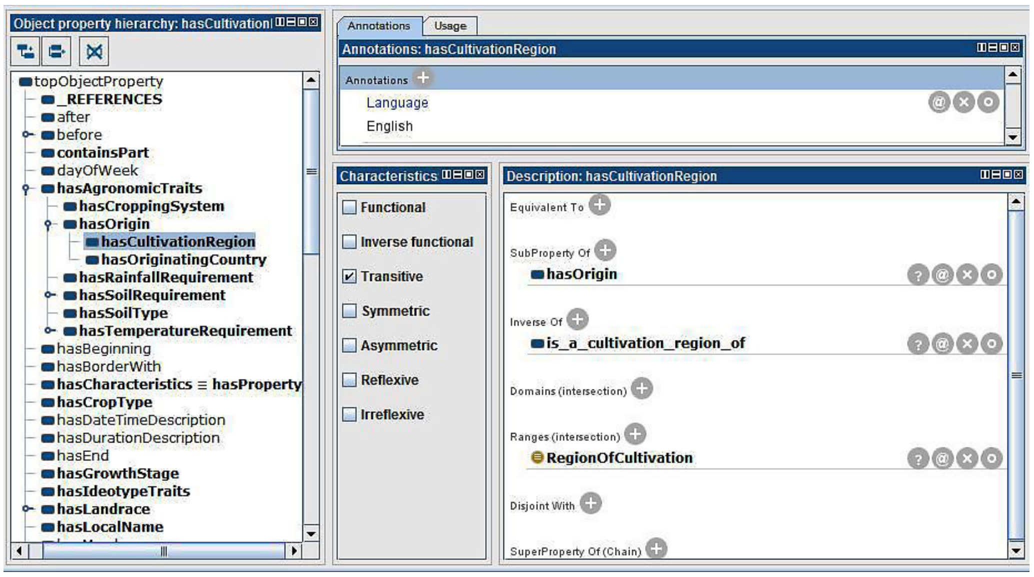


Figure 6.3: Ontograph showing fragment of UC-ONTO hierarchy


Object property hierarchy: hasCultivationRegion

topObjectProperty
_REFERENCES
after
before
containsPart
dayOfWeek
hasAgronomicTraits
hasCroppingSystem
hasOrigin
hasCultivationRegion
hasOriginatingCountry
hasRainfallRequirement
hasSoilRequirement
hasSoilType
hasTemperatureRequirement
hasBeginning
hasBorderWith
hasCharacteristics ≡ hasProperty
hasCropType
hasDateTimeDescription
hasDurationDescription
hasEnd
hasGrowthStage
hasIdeotypeTraits
hasLandrace
hasLocalName

Annotations: hasCultivationRegion

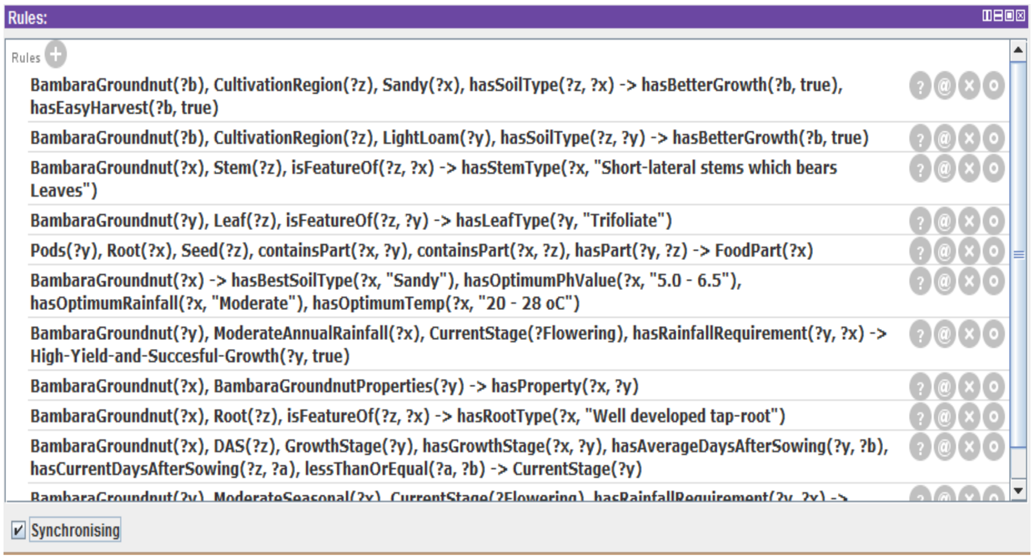
Annotations
Language
English

Characteristics
Functional
Inverse functional
Transitive ✓
Symmetric
Asymmetric
Reflexive
Irreflexive

Description: hasCultivationRegion

Equivalent To
SubProperty Of
hasOrigin
Inverse Of
is_a_cultivation_region_of
Domains (intersection)
Ranges (intersection)
RegionOfCultivation
Disjoint With
SuperProperty Of (Chain)

Figure 6.4: Object Property assertions showing Reflexivity with Domain and Range


Rules:

BambaraGroundnut(?b), CultivationRegion(?z), Sandy(?x), hasSoilType(?z, ?x) -> hasBetterGrowth(?b, true), hasEasyHarvest(?b, true)
BambaraGroundnut(?b), CultivationRegion(?z), LightLoam(?y), hasSoilType(?z, ?y) -> hasBetterGrowth(?b, true)
BambaraGroundnut(?x), Stem(?z), isFeatureOf(?z, ?x) -> hasStemType(?x, "Short-lateral stems which bears Leaves")
BambaraGroundnut(?y), Leaf(?z), isFeatureOf(?z, ?y) -> hasLeafType(?y, "Trifoliate")
Pods(?y), Root(?x), Seed(?z), containsPart(?x, ?y), containsPart(?x, ?z), hasPart(?y, ?z) -> FoodPart(?x)
BambaraGroundnut(?x) -> hasBestSoilType(?x, "Sandy"), hasOptimumPhValue(?x, "5.0 - 6.5"), hasOptimumRainfall(?x, "Moderate"), hasOptimumTemp(?x, "20 - 28 oC")
BambaraGroundnut(?y), ModerateAnnualRainfall(?x), CurrentStage(?Flowering), hasRainfallRequirement(?y, ?x) -> High-Yield-and-Succesful-Growth(?y, true)
BambaraGroundnut(?x), BambaraGroundnutProperties(?y) -> hasProperty(?x, ?y)
BambaraGroundnut(?x), Root(?z), isFeatureOf(?z, ?x) -> hasRootType(?x, "Well developed tap-root")
BambaraGroundnut(?x), DAS(?z), GrowthStage(?y), hasGrowthStage(?x, ?y), hasAverageDaysAfterSowing(?y, ?b), hasCurrentDaysAfterSowing(?z, ?a), lessThanOrEqual(?a, ?b) -> CurrentStage(?y)
BambaraGroundnut(?y), ModerateSeasonal(?x), CurrentStage(?Flowering), hasRainfallRequirement(?y, ?x) ->

✓ Synchronising

Figure 6.5: Rules interface containing some user-defined SWRL rules.

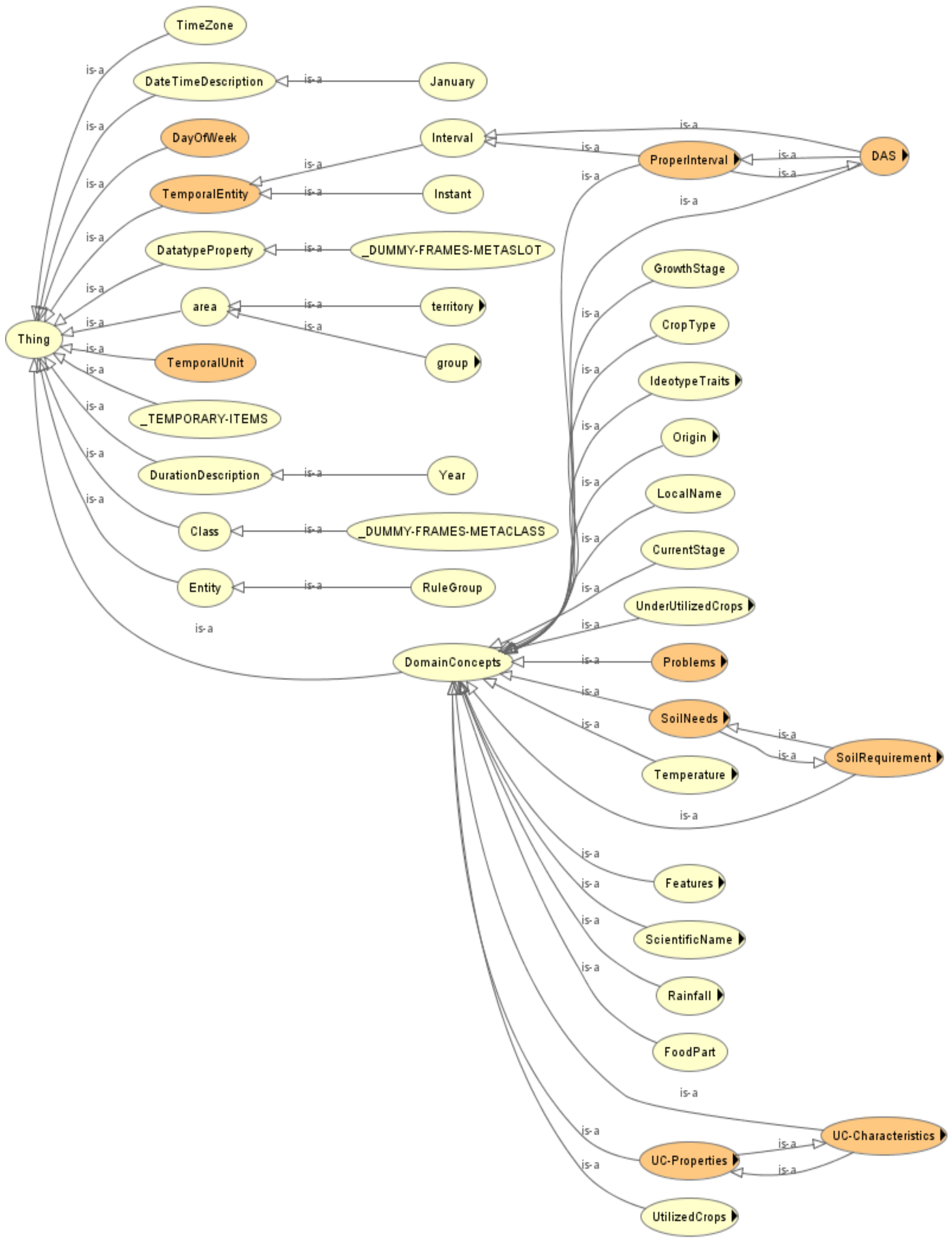
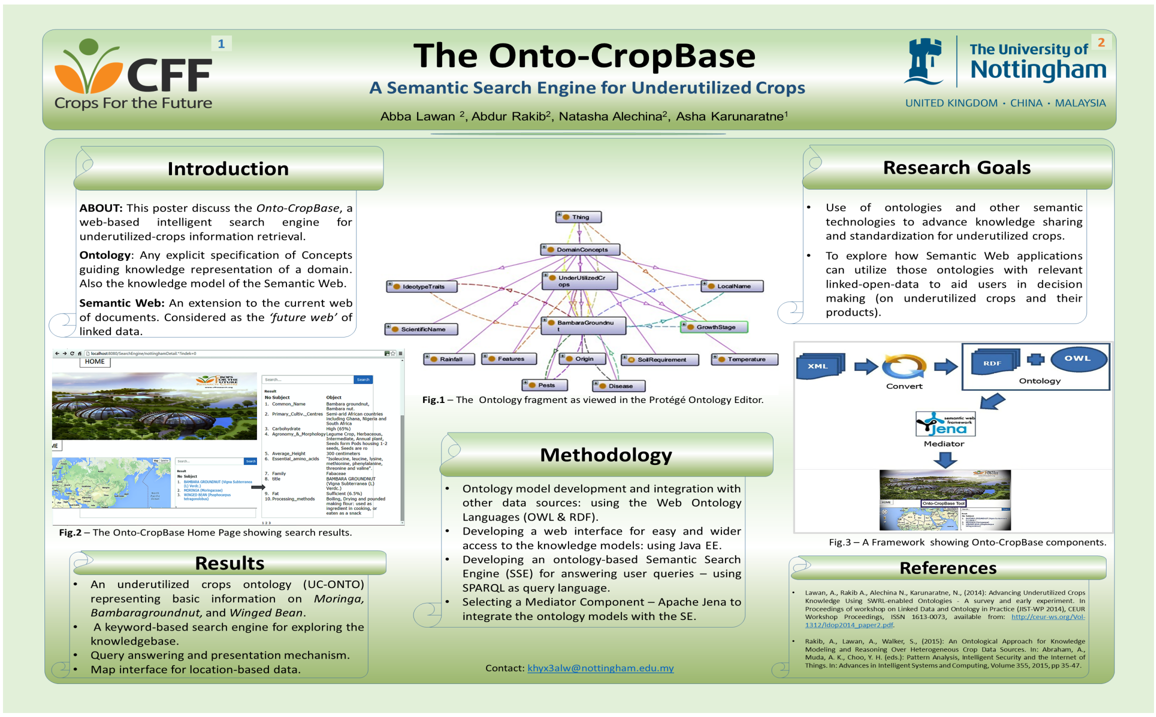


Figure 6.6: Graph-viz Protégé plug-in showing top-level Concepts of the UC-ONTO with their hierarchies



The Onto-CropBase

A Semantic Search Engine for Underutilized Crops

Abba Lawan [2], Abdur Rakib[2], Natasha Alechina[2], Asha Karunaratne[1]

Introduction

ABOUT: This poster discuss the *Onto-CropBase*, a web-based intelligent search engine for underutilized-crops information retrieval.

Ontology: Any explicit specification of Concepts guiding knowledge representation of a domain. Also the knowledge model of the Semantic Web.

Semantic Web: An extension to the current web of documents. Considered as the *'future web'* of linked data.

No	Subject	Object
1.	Common_Name	Bambara groundnut, Bambara nut.
2.	Primary_Cultiv._Centres	Semi-arid African countries including Ghana, Nigeria and South Africa
3.	Carbohydrate	High (65%)
4.	Agronomy_&_Morphology	Legume Crop, Herbaceous, Intermediate, Annual plant, Seeds form Pods housing 1-2 seeds, Seeds are ro
5.	Average_Height	300 centimeters
6.	Essential_amino_acids	"Isoleucine, leucine, lysine, methionine, phenylalanine, threonine and valine".
7.	Family	Fabaceae
8.	title	BAMBARA GROUNDNUT (Vigna Subterranea (L) Verdc.)
9.	Fat	Sufficient (6.5%)
10.	Processing_methods	Boiling, Drying and pounded making flour: used as ingredient in cooking, or eaten as a snack

Fig.2 – The Onto-CropBase Home Page showing search results.

Fig.1 – The Ontology fragment as viewed in the Protégé Ontology Editor.

Research Goals

- Use of ontologies and other semantic technologies to advance knowledge sharing and standardization for underutilized crops.
- To explore how Semantic Web applications can utilize those ontologies with relevant linked-open-data to aid users in decision making (on underutilized crops and their products).

Fig.3 – A Framework showing Onto-CropBase components.

Results

- An underutilized crops ontology (UC-ONTO) representing basic information on *Moringa, Bambaragroundnut,* and *Winged Bean*.
- A keyword-based search engine for exploring the knowledgebase.
- Query answering and presentation mechanism.
- Map interface for location-based data.

Methodology

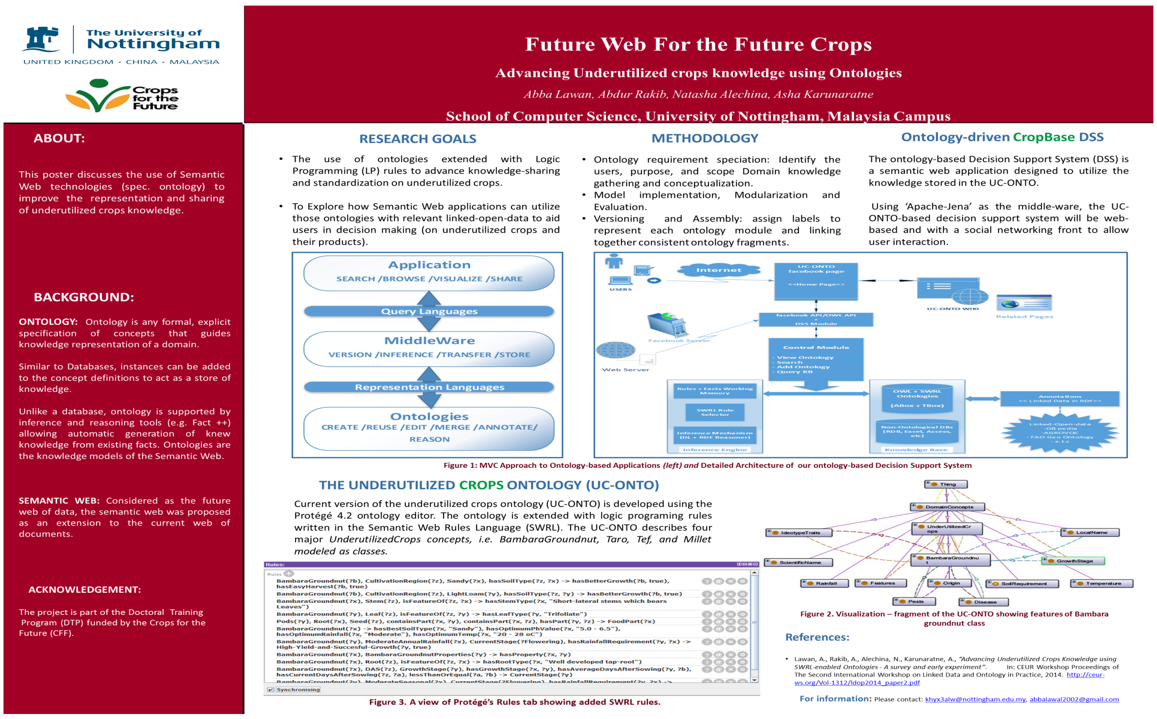
- Ontology model development and integration with other data sources: using the Web Ontology Languages (OWL & RDF).
- Developing a web interface for easy and wider access to the knowledge models: using Java EE.
- Developing an ontology-based Semantic Search Engine (SSE) for answering user queries – using SPARQL as query language.
- Selecting a Mediator Component – Apache Jena to integrate the ontology models with the SE.

Contact: khyx3alw@nottingham.edu.my

References

- Lawan, A., Rakib A., Alechina N., Karunaratne, N., (2014): Advancing Underutilized Crops Knowledge Using SWRL-enabled Ontologies - A survey and early experiment. In Proceedings of workshop on Linked Data and Ontology in Practice (JIST-WP 2014), CEUR Workshop Proceedings, ISSN 1613-0073, available from: http://ceur-ws.org/Vol-1312/ldop2014_paper2.pdf.
- Rakib, A., Lawan, A., Walker, S., (2015): An Ontological Approach for Knowledge Modeling and Reasoning Over Heterogeneous Crop Data Sources. In: Abraham, A., Muda, A. K., Choo, Y. H. (eds.): Pattern Analysis, Intelligent Security and the Internet of Things. In: Advances in Intelligent Systems and Computing, Volume 355, 2015, pp 35-47.

Figure 6.7: 2015 Research Showcase Poster - Onto-CropBase



Future Web For the Future Crops

Advancing Underutilized crops knowledge using Ontologies

Abba Lawan, Abdur Rakib, Natasha Alechina, Asha Karunaratne

School of Computer Science, University of Nottingham, Malaysia Campus

ABOUT:

This poster discusses the use of Semantic Web technologies (spec. ontology) to improve the representation and sharing of underutilized crops knowledge.

BACKGROUND:

ONTOLOGY: Ontology is any formal, explicit specification of concepts that guides knowledge representation of a domain.

Similar to Databases, instances can be added to the concept definitions to act as a store of knowledge.

Unlike a database, ontology is supported by inference and reasoning tools (e.g. Fact ++) allowing automatic generation of knew knowledge from existing facts. Ontologies are the knowledge models of the Semantic Web.

SEMANTIC WEB: Considered as the future web of data, the semantic web was proposed as an extension to the current web of documents.

ACKNOWLEDGEMENT:

The project is part of the Doctoral Training Program (DTP) funded by the Crops for the Future (CFF).

RESEARCH GOALS

- The use of ontologies extended with Logic Programming (LP) rules to advance knowledge-sharing and standardization on underutilized crops.
- To Explore how Semantic Web applications can utilize those ontologies with relevant linked-open-data to aid users in decision making (on underutilized crops and their products).

METHODOLOGY

- Ontology requirement speciation: Identify the users, purpose, and scope Domain knowledge gathering and conceptualization.
- Model implementation, Modularization and Evaluation.
- Versioning and Assembly: assign labels to represent each ontology module and linking together consistent ontology fragments.

Ontology-driven CropBase DSS

The ontology-based Decision Support System (DSS) is a semantic web application designed to utilize the knowledge stored in the UC-ONTO.

Using 'Apache-Jena' as the middle-ware, the UC-ONTO-based decision support system will be web-based and with a social networking front to allow user interaction.

Figure 1: MVC Approach to Ontology-based Applications *(left) and* Detailed Architecture of our ontology-based Decision Support System

THE UNDERUTILIZED CROPS ONTOLOGY (UC-ONTO)

Current version of the underutilized crops ontology (UC-ONTO) is developed using the Protégé 4.2 ontology editor. The ontology is extended with logic programing rules written in the Semantic Web Rules Language (SWRL). The UC-ONTO describes four major *UnderutilizedCrops concepts, i.e. BambaraGroundnut, Taro, Tef, and Millet modeled as classes.*

```
Rules:
BambaraGroundnut(?b), CultivationRegion(?z), Sandy(?x), hasSoilType(?z, ?x) -> hasBetterGrowth(?b, true),
hasEasyHarvest(?b, true)
BambaraGroundnut(?b), CultivationRegion(?z), LightLoam(?y), hasSoilType(?z, ?y) -> hasBetterGrowth(?b, true)
BambaraGroundnut(?x), Stem(?z), isFeatureOf(?z, ?x) -> hasStemType(?x, "Short-lateral stems which bears
Leaves")
BambaraGroundnut(?y), Leaf(?z), isFeatureOf(?z, ?y) -> hasLeafType(?y, "Trifoliate")
Pods(?y), Root(?x), Seed(?z), containsPart(?x, ?y), containsPart(?x, ?z), hasPart(?y, ?z) -> FoodPart(?x)
BambaraGroundnut(?x) -> hasBestSoilType(?x, "Sandy"), hasOptimumPhValue(?x, "5.0 - 6.5"),
hasOptimumRainfall(?x, "Moderate"), hasOptimumTemp(?x, "20 - 28 oC")
BambaraGroundnut(?y), ModerateAnnualRainfall(?x), CurrentStage(?Flowering), hasRainfallRequirement(?y, ?x) ->
High-Yield-and-Succesful-Growth(?y, true)
BambaraGroundnut(?x), BambaraGroundnutProperties(?y) -> hasProperty(?x, ?y)
BambaraGroundnut(?x), Root(?z), isFeatureOf(?z, ?x) -> hasRootType(?x, "Well developed tap-root")
BambaraGroundnut(?x), DAS(?z), GrowthStage(?y), hasGrowthStage(?x, ?y), hasAverageDaysAfterSowing(?y, ?b),
hasCurrentDaysAfterSowing(?z, ?a), lessThanOrEqual(?a, ?b) -> CurrentStage(?y)
Synchronising
```

Figure 3. A view of Protégé's Rules tab showing added SWRL rules.

Figure 2. Visualization – fragment of the UC-ONTO showing features of Bambara groundnut class

References:

- Lawan, A., Rakib, A., Alechina, N., Karunaratne, A., *"Advancing Underutilized Crops Knowledge using SWRL-enabled Ontologies - A survey and early experiment"*. In: CEUR Workshop Proceedings of The Second International Workshop on Linked Data and Ontology in Practice, 2014. http://ceur-ws.org/Vol-1312/ldop2014_paper2.pdf

For information: Please contact: khyx3alw@nottingham.edu.my, abbalawal2002@gmail.com

Figure 6.8: 2014 Research Showcase Poster - Future Web for the Future Crops

A2. Domain Expert's Evaluation

Name (Optional)	
Department/ Specialization	
Position	
Age (Optional)	

Tasks:

Kindly explore the *Onto-CropBase Semantic Search Engine* for relevant information as guided by the Underutilized Crops Ontology (UC-ONTO) introduced earlier and perform the following tasks:

1) Find all underutilized crops available in the UC-ONTO.
2) Find the details of the Bambara groundnut information.
3) Find and compare the Optimum germination temperature of Moringa and Bambara groundnut.

Component/Features	**Strongly Disagree**	**Slightly Disagree**	**Neutral**	**Slightly Agree**	**Strongly Agree**
Domain coverage and Datasets are sufficient					
Information presented is relevant					
Information presented is useful and relatively accurate.					
Searching (Query) style is clear and easy to use.					
User Interface (*labels, fonts, instructions*) is clear and Informative.					
Search results presentation is understandable.					
Navigating the search tool and results is easy					

Fill out below the features/components you like, the dislikes and those additional features you would like to see in the next version of the system.

Liked Features	**Dislikes**	**Additional Features**

Appendix B.

B1. Summary of Research Activities and Awards

A. Milestone achievements

No.	Planned Milestone	Milestone due (MM/YY)	Obstacles or Issues
1.	Initial Project Plan	Nov- Dec. 2013	•
2.	Project Requirements Design and Specification	Jan - Feb. 2014	•
3.	Formulation of Research Question.	March - May 2014	•
4.	Literature Review on Crop Ontologies and the use of Semantic technologies for domain knowledge modeling (to be published). Review on Logic Programming and Rule Languages – focusing on the Description Logics (DL), Horn-logic (HL) and their family of languages.	June - Oct. 2014	Not much is published on crop-related semantics.
5.	Initial Experiment: Development of Underutilized Crops Ontology Model -UC-ONTO version 1. Results presented at a workshop in Joint International Semantic-Web Conference (JIST 2014), Chiang Mai, Thailand.	Oct. - Dec. 2014	Lack of readily available domain (underutilized crops) data
6.	Ontology standardization and integration with other linked-open-data sources (e.g. FAO's geopolitical ontology).	Jan. - Mar. 2015	•
7.	Study on the 'Logic', 'Proof' and 'Trust' Layers of the Semantic Web	April - June 2015	Beyond the current research problems as understood during the literature review. Also the

			final results may not help towards achieving the research aim.
8.	Study on the linked-open data approaches and the Annotation of ambiguous concepts in the UC-ONTO with Natural Language.	June -July 2015	●
9.	Development of Ontology-based Knowledge Support System. (The Onto-Cropbase) – published. MYGeoExplorer Development: A collaboration work on the use of Ontologies and Linked-data to enhance the representation and querying of Geospatial information. A case study of the MyGDI (Malaysian Geographic Data and Information) System. Results presented in the International Conference on Research and Innovation in Information Systems (ICRIIS 2015), Malacca, Malaysia.	Aug. - Dec. 2015	●
10.	Preliminary Evaluation of the Prototype Application.	Jan -Feb. 2016	Lack of availability of domain experts for testing
11.	Second Year Report writeup, submission and viva	March – April 2016	-
12.	UN Research Showcase 2016 (Press release and Poster presentation)	April 2016	-
13.	Conference paper extension – Manuscript drafted on 'Ontology Development Practices in Agriculture'	March – April 2016	-
14.	Validation and Evaluation of The Onto-CropBase (Underutilized crops Semantic Search Engine).	May 2016	-
15.	Attended the BDAS'16 Conference in Krakow, Poland to present a paper on the Onto-CropBase Tool	29th May – 4th June 2016	-
16.	Break + Mobility to the UK campus (July – Sept)	June 2016	-
17.	Proposal of a language extension to the Semantic Web Rule Language (SWRL) Knowledge Representation formalism –employed in the development of the Underutilized Crops ontology.	July 2016	-
18.	Feasibility study on the proposed 'Complete Semantic Web Rule Language (C-SWRL)' – a proposal to unify all	July – Aug 2016	Proposal too large and vague for the allotted time. Also, not very

	SWRL extensions based on Translational Logic & Complexity analysis.		relevant in crops data modelling.
19.	Feasibility study on the proposed SWRL-DL extension (To provide language constructs for automatic ontology development and for ontology learning)	August 2016	Requires collaboration with experts on Machine Learning.
20.	Feasibility study on Agri-SWRL extension – a SWRL extension for converting agricultural vocabularies, e.g. Agrovoc, to Ontologies.	Sept – Oct 2016	Literature review shows that Agrovoc is already in OWL-DL format. Plus, ontology vs Thesauri Lexicalization study is too large (out of scope).
21.	Preparation of manuscript on the reviewed Semantic Web Rule Language (SWRL) extensions. Review and inclusion of the Probabilistic Extensions and Decidability studies of all reviewed Extensions.	Oct. – Nov. 2016	-
22.	Proposal and manuscript draft on Fuzzy Temporal Extension to the SWRL language (FT-SWRL)	Nov. - Dec. 2016	-
23.	Doctoral Training Partnership Week 2016 International Conference on Future Agriculture, CFF, Malaysia. • (Best Presentation Award)	Dec. 2016	-
24.	Thesis organization and submission.	Jan. - March 2017	-

B. Project Achievements.

Publications:

[1] Lawan, A., Rakib, A., Alechina, N.: *Towards Comprehensive Domain Modeling on the Semantic Web – a Review of SWRL Expressiveness Extensions.* Submitted to the Journal of Web Semantics, Elsevier, Manuscript Number: JWS-D-17-00081, March, 2017.

[2] Lawan, A., Rakib, A., Alechina, N.: FT-SWRL - *A Fuzzy Temporal Extension of the Semantic Web Rule Language*. Submitted to: International Journal of Approximate Reasoning, Elsevier, March, 2017

[3] Lawan, A., Rakib, A., Alechina, N., and Karunaratne, A., 2016: *The Onto-CropBase - A Semantic Web Application for Querying Crops Linked-Data*. In 'Beyond Databases,

Architectures and Structures. Advanced Technologies for Data Mining and Knowledge Discovery' Volume 613 of the series Communications in Computer and Information Science, pp 384-399.

[4] Rakib, A., Lawan, A., and Walker, Sue., 2014: *An ontological approach for knowledge modelling and reasoning over heterogeneous crop data sources.* In: Pattern Analysis, Intelligent Security and the Internet of Things, Advances in Intelligent Systems and Computing, Volume 355, 2015, pp 35-47.

[5] Panchanathan, S., Lawan, A., and Rakib, A. 2015: *MyGeo-Explorer: A semantic search tool for querying geospatial information.* In ARPN Journal of Engineering and Applied Sciences, Vol.10(23), 2015.

[6] Lawan, A., Rakib, A., Alechina, N., and Karunaratne, A., 2014: *Advancing Underutilized Crops Knowledge using SWRL-enabled Ontologies - A survey and early experiment*. In: CEUR Workshop Proceedings of The Second International Workshop on Linked Data and Ontology in Practice, Vol-1312, Pages 69-84, 2014.

Conferences Attended:

Dec. 2016	3rd Doctoral Training Partnership (DTP) Week 2016, Crops for the Future (CFF), Malaysia.
28th May – 4th June 2016	12th International Conference 'Beyond Databases, Architectures and Structures (BDAS 2016), Ustron, Poland.
April, 2016	Research Showcase 2016, UNMC, 1-day Poster Presentation session, by Research Training and Development Unit of UNMC.
8th – 10th Dec. 2015	4th International Conference on Research and Innovation in Information Systems (ICRIIS) 2015, Malacca, Malaysia. By Malaysian Association for Information Systems Chapter (MyAIS).
11th Feb. 2015	Annual Research Talk, Faculty of Science, University of Nottingham, Malaysia Campus.
8th – 10th Dec. 2014	2014 Fourth World Congress on Information and Communication Technologies (WICT), Malacca, Malaysia. Machine Intelligence Research Labs (MIRLABS).
9th – 11th Nov. 2014	4th Joint International Semantic Technology (JIST 2014) Conference, Chiang Mai, Thailand. By Language and Semantics Laboratory, National Electronics and Computer Technology Centre (NECTEC), Thailand.
8th May, 2015	Research Showcase 2015, UNMC, 1-day Poster Presentation session, Research Training and Development Unit of UNMC.

Bibliography

[1] T. Berners-Lee, “Semantic Web and Linked Data,” 2009.

[2] T. Z. Berardini, “The Gene Ontology in 2010: Extensions and refinements,” *Nucleic Acids Research*, vol. 38, 2009.

[3] L. Cooper, R. L. Walls, J. Elser, M. a. Gandolfo, D. W. Stevenson, B. Smith, J. Preece, B. Athreya, C. J. Mungall, S. Rensing, M. Hiss, D. Lang, R. Reski, T. Z. Berardini, D. Li, E. Huala, M. Schaeffer, N. Menda, E. Arnaud, R. Shrestha, Y. Yamazaki, and P. Jaiswal, “The plant ontology as a tool for comparative plant anatomy and genomic analyses.,” *Plant & cell physiology*, vol. 54, p. e1, feb 2013.

[4] R. Shrestha, R. Mauleon, R. Simon, J. Balaji, S. Channelière, A. Alercia, M. Senger, K. Manansala, T. Metz, G. Davenport, R. Bruskiewich, G. McLaren, and E. Arnaud, “Development of GCP Ontology for Sharing Crop Information,” *Nature Precedings*, p. 6, apr 2009.

[5] J. M. A. Calero, A. M. Ortega, G. M. Perez, J. A. B. Blaya, and A. F. G. Skarmeta, “A non-monotonic expressiveness extension on the semantic web rule language,” *J. Web Eng.*, vol. 11, pp. 93–118, June 2011.

[6] J. H. Tim Berners-Lee and O. Lassila, “The Semantic Web,” *Scientific American Magazine*, may 2011.

[7] W. Groups, “Linked data,” 2015.

[8] J. Williams and N. Haq, *Global research on underutilized crops: An assessment of current activities and proposals for enhanced cooperation*. Bioversity International.

[9] A. W. Ebert, “Potential of underutilized traditional vegetables and legume crops to contribute to food and nutritional security, income and more sustainable production systems,” *Sustainability*, vol. 6, no. 1, pp. 319–335, 2014.

[10] S. Padulosi, T. Hodgkin, J. Williams, and N. Haq, “Underutilised crops: trends, challenges and opportunities in the 21st century,” in *Managing plant genetic diversity* (J. Engels, V. Rao, and M. Jackson, eds.), pp. 323–338, CAB International, 2002.

[11] A. Lawan, A. Rakib, N. Alechina, and A. Karunaratne, *The Onto-CropBase – A Semantic Web Application for Querying Crops Linked-Data*, pp. 384–399. Cham: Springer International Publishing, 2016.

[12] T. R. Gruber, “Technical Report KSL 92-71 Revised April 1993 A Translation Approach to Portable Ontology Specifications by A Translation Approach to Portable Ontology Specifications,” no. April, 1993.

[13] N. F. Noy and D. L. Mcguinness, “Ontology Development 101 : A Guide to Creating Your First Ontology,” tech. rep., 2000.

[14] R. B. Mishra and S. Kumar, “Semantic web reasoners and languages,” *Artificial Intelligence Review*, vol. 35, no. 4, pp. 339–368, 2011.

[15] T. Eiter, G. Ianni, T. Krennwallner, and A. Polleres, “Reasoning web,” ch. Rules and Ontologies for the Semantic Web, pp. 1–53, Berlin, Heidelberg: Springer-Verlag, 2008.

[16] T. Berners-Lee, J. Hendler, and O. Lassila, “The semantic web,” *Scientific American*, pp. 29–37, May 2001.

[17] W. C. Rounds and G.-Q. Zhang, “Clausal logic and logic programming in algebraic domains,” *Inf. Comput.*, vol. 171, pp. 183–200, Jan. 2002.

[18] A. Horn, “On Sentences Which are True of Direct Unions of Algebras,” *J. Symb. Log.*, vol. 16, no. 1, pp. 14–21, 1951.

[19] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press, 2003.

[20] R. ROSATI, “On the decidability and complexity of integrating ontologies and rules,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, pp. 61–73, jul 2005.

[21] V. D. Dragan Gašević, Dragan Djurić, *Model Driven Architecture and Ontology Development*. Berlin: Springer, 2006.

[22] H. H. Wang, J. S. Dong, J. Sun, and J. Sun, “Reasoning support for semantic web ontology family languages using alloy,” *Multiagent and Grid Systems*, vol. 2, no. 4, pp. 455–471, 2006.

[23] N. F. Noy and D. L. McGuinness, “Ontology development 101: A guide to creating your first ontology,” tech. rep., Stanford, 2001.

[24] H. H. Wang, N. Noy, A. Rector, M. Musen, T. Redmond, D. Rubin, S. Tu, T. Tudorache, N. Drummond, M. Horridge, *et al.*, “Frames and owl side by side,” in *Presentation Abstracts*, p. 54, 2006.

[25] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harlemen, M. Klein, S. Staab, R. Studer, E. Motta, and I. Horrocks, “The ontology inference layer oil,” 2000.

[26] W3C, “OWL 2 Web Ontology Language Document Overview, World Wide Web Consortium, recommendation rec-owl2-overview-20121211,” Decenmber 2012.

[27] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, “OWL 2 Web Ontology Language Profiles,” 2009.

[28] M. Boris, C. G. Bernardo, I. Horrocks, W. Zhe, F. Achille, and L. Carsten, “OWL 2 Web Ontology Language Profiles (Second Edition),” *W3C Recommendation*, no. December, pp. 1–43, 2012.

[29] G. Meditskos and N. Bassiliades, “DLEJena: A practical forward-chaining OWL 2 RL reasoner combining Jena and Pellet,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 8, no. 1, pp. 89–94, 2010.

[30] K. Dentler, R. Cornet, A. Ten Teije, and N. De Keizer, “Comparison of reasoners for large ontologies in the OWL 2 EL profile,” *Semantic Web*, vol. 2, no. 2, pp. 71–87, 2011.

[31] S. M. Q., P. C., S. K. A., and W. A. Y., “SNOMED Clinical Terms: Overview of the development process and project status,” 2001.

[32] G. Shen, Z. Huang, X. Zhu, L. Wang, and G. Xiang, “Using description logics reasoner for ontology matching,” in *Intelligent Information Technology Application, Workshop on*, pp. 30–33, IEEE, 2007.

[33] O. Bodenreider, “Biomedical ontologies in action: role in knowledge management, data integration and decision support,” *Yearbook of medical informatics*, p. 67, 2008.

[34] O. Corcho and A. Gómez-Pérez, “A roadmap to ontology specification languages,” in *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, EKAW ’00, (London, UK, UK), pp. 80–96, Springer-Verlag, 2000.

[35] S. Staab and R. Studer, *Handbook on Ontologies*. Springer Science & Business Media, 2010.

[36] C. Keßler, M. Raubal, and C. Wosniok, “Semantic rules for context-aware geographical information retrieval,” *Smart Sensing and Context*, vol. 5741 LNCS, pp. 77–92, 2009.

[37] G. Guizzardi and G. Wagner, “Using the Unified Foundational Ontology (UFO) as a foundation for general conceptual modeling languages,” in *Theory and Applications of Ontology: Computer Applications*, pp. 175–196, Springer Netherlands, 2010.

[38] B. Smith and P. Grenon, “Basic formal ontology (bfo),” *INFOMIS Reports*, 2006.

[39] H. Herre, “General Formal Ontology (GFO): A foundational ontology for conceptual modelling,” in *Theory and Applications of Ontology: Computer Applications*, pp. 297–345, Springer Netherlands, 2010.

[40] R. Hoehndorf, F. Loebe, R. Poli, J. Kelso, and H. Herre, “GFO-Bio: A biomedical core ontology,” *Applied Ontology*, vol. 3, no. 4, pp. 219–227, 2008.

[41] E. Beisswanger, S. Schulz, H. Stenzhorn, and U. Hahn, “BIOTOP : An Upper Domain Ontology for the Life Sciences,” *Applied Ontology*, vol. 3, no. 4, pp. 205–212, 2008.

[42] C. Caracciolo, A. Morshed, A. Stellato, G. Johannsen, Y. Jaques, and J. Keizer, “Thesaurus maintenance, alignment and publication as linked data: The AGROOVOC use case,” in *Communications in Computer and Information Science*, vol. 240 CCIS, pp. 489–499, 2011.

[43] C. Caraccioloa, A. Stellatob, A. Morsheda, G. Johannsena, S. Rajbhandaria, Y. Jaquesa, and J. Keizera, “The agrovoc linked dataset,” *Semantic Web*, vol. 4, no. 3, pp. 341–348, 2013.

[44] M. T. Pazienza, A. Stellato, A. G. Tudorache, A. Turbati, and F. Vagnoni, “An architecture for data and knowledge acquisition for the semantic web: The AGROVOC use case,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7567 LNCS, pp. 426–433, 2012.

[45] A. Lawan, A. Rakib, N. Alechina, and A. Karunaratne, “Advancing underutilized crops knowledge using SWRL-enabled ontologies - A survey and early experiment,” in *CEUR Workshop Proceedings*, vol. 1312, (Chiang Mai), pp. 69–84, 2014.

[46] T. Pizzuti, G. Mirabelli, M. A. Sanz-Bobi, and F. Goméz-Gonzaléz, “Food Track & Trace ontology for helping the food traceability control,” *Journal of Food Engineering*, vol. 120, no. 1, pp. 17–30, 2014.

[47] J. Cantais, D. Dominguez, V. Gigante, L. Laera, and V. Tamma, "An example of food ontology for diabetes control," in *Working notes of the ISWC 2005 Workshop on Ontology Patterns for the Semantic Web*, p. 9, 2005.

[48] H. C. Li and W. M. Ko, "Automated food ontology construction mechanism for diabetes diet care," in *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, ICMLC 2007*, vol. 5, pp. 2953–2958, 2007.

[49] C. Schlenoff, T. Hong, C. Liu, R. Eastman, and S. Foufou, "A literature review of sensor ontologies for manufacturing applications," *2013 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pp. 96–101, oct 2013.

[50] R. L. Walls, B. Athreya, L. Cooper, J. Elser, M. a. Gandolfo, P. Jaiswal, C. J. Mungall, J. Preece, S. Rensing, B. Smith, and D. W. Stevenson, "Ontologies as integrative tools for plant science.," *American journal of botany*, vol. 99, pp. 1263–75, aug 2012.

[51] R. Bruskiewich, G. Davenport, T. Hazekamp, T. Metz, M. Ruiz, R. Simon, M. Takeya, J. Lee, M. Senger, G. McLaren, and T. Van Hintum, "Generation Challenge Programme (GCP): standards for crop data.," *Omics : a journal of integrative biology*, vol. 10, no. 2, pp. 215–9, 2006.

[52] L. Matteis, P. Chibon, H. Espinosa, M. Skofic, R. Finkers, R. Bruskiewich, G. Hyman, and E. Arnaud, "Crop Ontology: Vocabulary For Crop-related Concepts," in *Proceedings of the first international Workshop on Semantics for Biodiversity* (P. Larmande, E. Arnaud, I. Mougenot, C. Jonquet, T. Libourel, and M. Ruiz, eds.), vol. 979, 2013.

[53] R. Stevens and P. Lord, "Application of Ontologies in Bioinformatics," in *Handbook on Ontologies*, pp. 735–756, 2009.

[54] N. J. Deshpande and R. Kumbhar, "Construction and applications of ontology: Recent trends," *DESIDOC Journal of Library & Information Technology*, vol. 31, pp. 84–89, 2011.

[55] D. Gašević, D. Djurić, and V. Devedžić, *Model driven architecture and ontology development*. Springer Science & Business Media, 2006.

[56] T. Tudorache, C. Nyulas, N. F. Noy, and M. A. Musen, "Webprot&#233;g&#233;: A collaborative ontology editor and knowledge acquisition tool for the web," *Semant. web*, vol. 4, pp. 89–99, Jan. 2013.

[57] N. M. Yahia, A. Sahar, and A. AbdelWahab, "Automatic generation of OWL ontology from XML data source," *Computing Research Repositoty (CoRR)*, vol. abs/1206.0570, 2012.

[58] H. Bohring and S. Auer, "Mapping XML to OWL ontologies," in *Leipziger Informatik-Tage*, vol. 72, pp. 147–156, GI, 2005.

[59] M. J. O'Connor and A. Das, "Acquiring OWL ontologies from XML documents," in *Proceedings of the sixth international conference on Knowledge capture*, (New York, NY, USA), pp. 17–24, ACM, 2011.

[60] T. Rodrigues, P. Rosa, and J. Cardoso, "Mapping XML to existing OWL ontologies," in *Proceedings of the International Conference WWW/Internet 2006*, pp. 72–77, 2006.

[61] A. Rakib, A. Lawan, and S. Walker, "An ontological approach for knowledge modeling and reasoning over heterogeneous crop data sources," in *Pattern Analysis, Intelligent Security and the Internet of Things*, pp. 35 – 47, Springer International Publishing, 2015.

[62] A. Hogan, J. Z. Pan, A. Polleres, and Y. Ren, *Reasoning Web. Semantic Technologies for the Web of Data*, vol. 6848 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011.

[63] E. Jërôme and S. Pavel.

[64] N. N. Fridman and M. A. Musen, "Prompt: Algorithm and tool for automated ontology merging and alignment.," in *Proceedings of the International Conference on Agents and Artificial Intelligence*, pp. 450–455, AAAI Press / The MIT Press, 2000.

[65] J. Brank, M. Grobelnik, and D. Mladenić, “A survey of ontology evaluation techniques,” in *Proc. of 8th Int. multi-conf. Information Society*, pp. 166–169, 2005.

[66] K. Supekar, “A peer-review approach for ontology evaluation,” in *8th Int. Protégé Conference, Madrid, Spain*, July 2005.

[67] A. Bachir Bouiadjra and S. M. Benslimane, “Ontology evaluation - state of the art, new approach and perspectives,” in *International Conference on Knowledge Engineering and Ontology Development (KEOD)*, pp. 365–368, 2011.

[68] A. I. Walisadeera, A. Ginige, and G. N. Wikramanayake, *Ontology Evaluation Approaches: A Case Study from Agriculture Domain*, pp. 318–333. Cham: Springer International Publishing, 2016.

[69] C. Bezerra, F. Freitas, and F. Santana, “Evaluating ontologies with competency questions,” in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 3, pp. 284–285, Nov 2013.

[70] I. Horrocks, P. F. Patel-schneider, S. Bechhofer, D. Tsarkov, P. PATELSCHNEIDER, S. Bechhofer, and D. Tsarkov, “OWL rules: A proposal and prototype implementation,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, pp. 23–40, jul 2005.

[71] A. Cregan, M. Mochol, D. Vrandecic, and S. Bechhofer, “Pushing the limits of owl, rules and protg – a simple example.”

[72] B. Motik, U. Sattler, and R. Studer, “Query answering for owl-dl with rules,” *Web Semant.*, vol. 3, pp. 41–60, July 2005.

[73] M. Krötzsch, F. Maier, A. A. Krisnadhi, and P. Hitzler, “A Better Uncle for OWL: Nominal Schemas for Integrating Rules and Ontologies,” in *20th International World Wide Web Conference (WWW2011)*, p. 645, 2011.

[74] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits, “Well-founded semantics for description logic programs in the semantic web,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3323 LNCS, pp. 81–97, 2004.

[75] U. Hustadt, B. Motik, and U. Sattler, “Data Complexity of Reasoning in Very Expressive Description Logics,” in *Proc. of the 19th Int. Joint Conference on Artificial Intelligence (IJCAI 2005)* (L. P. Kaelbling and A. Saffiotti, eds.), pp. 466–471, Morgan Kaufmann Publishers, 2005.

[76] G. Antoniou, C. V. Damásio, B. Grosof, I. Horrocks, M. Kifer, J. Ma luszynski, and Peter, “Combining Rules and Ontologies. A survey.,” 2005.

[77] R. Rosati, “Integrating ontologies and rules: Semantic and computational issues,” *Reasoning Web*, pp. 128–151, 2006.

[78] A. Krisnadhi, F. Maier, and P. Hitzler, “OWL and Rules,” ... *Web. Semantic Technologies for the Web ...*, pp. 382–415, 2011.

[79] T. Eiter, G. Ianni, A. Polleres, R. Schindlauer, H. Tompits, U. Rey, and J. Carlos, “Reasoning with Rules and Ontologies,” *Reasoning Web 2006*, vol. 4126, pp. 93–127, 2006.

[80] R. Rosati, “Semantic and computational advantages of the safe integration of ontologies and rules,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3703 LNCS, pp. 50–64, 2005.

[81] R. Rosati, “Dl+ log: Tight integration of description logics and disjunctive datalog.,” *KR*, vol. 6, pp. 68–78, 2006.

[82] B. Motik, U. Sattler, and R. Studer, “Query answering for OWL-DL with rules,” *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, pp. 41–60, 2005.

[83] P. F. Patel-Schneider, D. L. McGuinness, R. J. Brachman, and L. A. Resnick, “The classic knowledge representation system: Guiding principles and implementation rationale,” *SIGART Bull.*, vol. 2, pp. 108–113, June 1991.

[84] A. Y. Levy and M.-C. Rousset, “Combining horn rules and description logics in {CARIN},” *Artificial Intelligence*, vol. 104, no. 12, pp. 165 – 209, 1998.

[85] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker, “Description logic programs: Combining logic programs with description logic,” in *Proceedings of the 12th International Conference on World Wide Web*, WWW ’03, (New York, NY, USA), pp. 48–57, ACM, 2003.

[86] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits, “Combining answer set programming with description logics for the Semantic Web,” *Artificial Intelligence*, vol. 172, pp. 1495–1539, aug 2008.

[87] A. Krisnadhi, F. Maier, and P. Hitzler, “Owl and rules,” in *Proceedings of the 7th International Conference on Reasoning Web: Semantic Technologies for the Web of Data*, RW’11, pp. 382–415, Springer-Verlag, 2011.

[88] F. Olken, M. Palmirani, and D. Sottara, *Rule - Based Modeling and Computing on the Semantic Web: 5th International Symposium, RuleML 2011 - America, Ft. Lauderdale, FL, USA, November 3-5, 2011, Proceedings*. Lecture Notes in Computer Science / Programming and Software Engineering, Springer, 2011.

[89] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, *et al.*, “Swrl: A semantic web rule language combining owl and ruleml,” *W3C Member submission*, vol. 21, p. 79, 2004.

[90] D. L. McGuinness, F. Van Harmelen, *et al.*, “Owl web ontology language overview,” *W3C recommendation*, vol. 10, no. 10, p. 2004, 2004.

[91] G. Antoniou and F. van Harmelen, eds., *Semantic Web Primer - Second Edition*. Cambridge, Massachusetts London, England: MIT Press, 2008.

[92] J. Mei and E. P. Bontas, "Reasoning paradigms for swrl-enabled ontologies," 2005.

[93] I. Horrocks, P. F. Patel-schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL : A Semantic Web Rule Language Combining OWL and RuleML," 2004.

[94] C. Golbreich, O. Dameron, O. Bierlaire, and B. Gibaud, "What reasoning support for ontology and rules? the brain anatomy case study," in *Workshop on OWL Experiences and Directions*, 2005.

[95] T. Eiter, G. Ianni, A. Polleres, R. Schindlauer, and H. Tompits, "Reasoning with rules and ontologies," in *Reasoning Web* (P. Barahona, F. Bry, E. Franconi, N. Henze, and U. Sattler, eds.), vol. 4126 of *Lecture Notes in Computer Science*, pp. 93–127, Springer Berlin Heidelberg, 2006.

[96] G. Meditskos and N. Bassiliades, "Clips-owl: A framework for providing object-oriented extensional ontology queries in a production rule engine," *Data and Knowledge Engineering*, vol. 70, no. 7, pp. 661–681, 2011.

[97] S. Ceri, G. Gottlob, and L. Tanca, "What you always wanted to know about datalog (and never dared to ask)," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 1, pp. 146–166, Mar 1989.

[98] R. Rosati, "Towards expressive kr systems integrating datalog and description logics: preliminary report.," *Description Logics*, vol. 22, 1999.

[99] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press, 2003.

[100] V. Lifschitz, "Nonmonotonic databases and epistemic queries," in *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'91, (San Francisco, CA, USA), pp. 381–386, Morgan Kaufmann Publishers Inc., 1991.

[101] T. Lukasiewicz, “A novel combination of answer set programming with description logics for the semantic web,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 11, pp. 1577–1592, 2010.

[102] M. O’Connor and A. Das, “Sqwrl: A query language for owl,” in *Proceedings of the 6th International Conference on OWL: Experiences and Directions - Volume 529*, OWLED’09, (Aachen, Germany, Germany), pp. 208–215, CEUR-WS.org, 2009.

[103] I. Horrocks, P. F. Patel-Schneider, S. Bechhofer, and D. Tsarkov, “Owl rules: A proposal and prototype implementation,” *Web Semant.*, vol. 3, pp. 23–40, July 2005.

[104] P. F. Patel-Schneider, P. Hayes, I. Horrocks, and F. Harmelen, “A proposal for a swrl extension to first-order logic,” *Proposal, DARPA DAML Program*, 2004.

[105] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits, “Combining answer set programming with description logics for the semantic web,” *Artif. Intell.*, vol. 172, pp. 1495–1539, Aug. 2008.

[106] M. Kifer and S. Brook, “Requirements for an Expressive Rule Language on the Semantic Web,” *W3C Workshop on Rules Languages*, no. April, 2005.

[107] R. Rosati, “Integrating ontologies and rules: Semantic and computational issues,” vol. 4126 of *Lecture Notes in Computer Science*, pp. 128–151, Springer, 2006.

[108] T. W. Wlodarczyk, C. Rong, M. O’Connor, and M. Musen, “Swrl-f: A fuzzy logic extension of the semantic web rule language,” in *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, WIMS ’11, (New York, NY, USA), pp. 39:1–39:9, ACM, 2011.

[109] L. Predoiu, “Probabilistic Models for the Semantic Web A Survey,” *International Journal of Approximate Reasoning*, pp. 288–307, 2007.

[110] J. Z. Pan, G. Stoilos, G. Stamou, V. Tzouvaras, and I. Horrocks, “f-swrl: a fuzzy extension of swrl,” pp. 28–46, 2006.

[111] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[112] S. Agarwal and P. Hitzler, "Modeling fuzzy rules with description logics," *CEUR Workshop Proceedings*, vol. 188, no. 5, 2005.

[113] X. Wang, Z. Ma, L. Yan, and X. Meng, "Vague-SWRL: a fuzzy extension of SWRL," *International Conference on Web Reasoning and Rule Systems*, pp. 232–233.

[114] W.-L. Gau and D. J. Buehrer, "Vague sets," *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 2, pp. 610–614, 1993.

[115] A. Paschke and H. Boley, "Rule markup languages and semantic web rule languages," in *Rule Markup Languages and Semantic Web Rule Languages*, pp. 1–24, IGI Global, 2010.

[116] X. Wang, Z. M. Ma, C. Xu, and J. Cheng, "Nonmonotonic fuzzy rules in the semantic web," in *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 2, pp. 275–279, Aug 2009.

[117] P. C. G. Da Costa, K. B. Laskey, and K. J. Laskey, "Pr-owl: A bayesian ontology language for the semantic web," in *Proceedings of the 2005 International Conference on Uncertainty Reasoning for the Semantic Web - Volume 173*, URSW'05, (Aachen, Germany, Germany), pp. 23–33, CEUR-WS.org, 2005.

[118] Z. Ding, Y. Peng, and R. Pan, "BayesOWL: Uncertainty modeling in semantic web ontologies," *Studies in Fuzziness and Soft Computing*, vol. 204, pp. 3–29, 2006.

[119] Y. Liu, S. Chen, S. Li, and Y. Wang, "Bayes-SWRL: A probabilistic extension of SWRL," *Proceedings - 9th International Conference on Computational Intelligence and Security, CIS 2013*, vol. 1, pp. 702–706, 2013.

[120] D. Darwiche, "Bayesian networks," *Communications of the ACM*, vol. 53, no. 12, pp. 80–90, 2010.

[121] J. M. A. Calero, A. M. Ortega, G. M. Perez, J. A. B. Blaya, A. F. G. Skarmeta, M. Günther, T. Wiemann, S. Albrecht, J. Hertzberg, J. Zhang, L. Zhang, S. Rockel, B. Neumann, J. Lehmann, K. S. R. Dubba, A. G. Cohn, A. Saffiotti, F. Pecora, M. Mansouri, Š. Konečný, M. Günther, S. Stock, L. S. Lopes, M. Oliveira, G. H. Lim, H. Kasaei, V. Mokhtari, L. Hotz, W. Bohlken, and S. Rudolph, "A Non-monotonic Expressiveness Extension on the Semantic Web Rule Language," *Artificial Intelligence*, vol. 11, no. May, pp. 1–36, 2011.

[122] W. Li and S. Tian, "XSWRL, an extended semantic web rule language and prototype implementation," *Expert Systems with Applications*, vol. 38, pp. 2040–2045, mar 2011.

[123] P. F. Patel-Schneider, "A proposal for a swrl extension towards first-order logic," *W3C Member Submission, April*, 2005.

[124] C. McKenzie, P. Gray, and A. Preece, "Extending SWRL to express fully-quantified constraints," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3323 LNCS, pp. 139–154, 2004.

[125] W. Kim, M. Lee, J. Hong, T. Wang, and H. Kim, "Merging Mathematical Constraint Knowledge with the Semantic Web using a Semantic Web Constraint Language," *Iconceptpress.Com*, vol. 2003, no. October 2009, 2003.

[126] H.-J. Kim, W. Kim, and M. Lee, "Semantic web constraint language and its application to an intelligent shopping agent," *Decision Support Systems*, vol. 46, no. 4, pp. 882–894, 2009.

[127] D. Elenius and M.-o. Stehr, "Rules and Computation on the Semantic Web."

[128] A. Sánchez-Macián, E. Pastor, J. E. de López Vergara, and D. López, "Extending swrl to enhance mathematical support," in *International Conference on Web Reasoning and Rule Systems*, pp. 358–360, Springer, 2007.

[129] M. J. O. Connor, “A Method for Representing and Querying Temporal Information in OWL,” *Communications in Computer and Information Science*, vol. 127 CCIS,, no. January 2011, pp. 97–110, 2011.

[130] D. Elenius and S. Riehemann, “Swrl-iq user manual,” 2012.

[131] “Swrl existentials.”

[132] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc., 1986.

[133] M. Arenas, B. Cuenca Grau, E. Kharlamov, S. Marciuska, D. Zheleznyakov, and E. Jimenez-Ruiz, “Semfacet: Semantic faceted search over yago,” in *Proceedings of the 23rd International Conference on World Wide Web*, WWW ’14 Companion, pp. 123–126, 2014.

[134] S. Panchanathan, A. Lawan, and A. Rakib, “Mygeo -explorer : A semantic search tool for querying geospatial information,” *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 23, pp. 18012–18020, 2015.

[135] R. Bhagdev, S. Chapman, F. Ciravegna, V. Lanfranchi, and D. Petrelli, “Hybrid search: Effectively combining keywords and semantic searches,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5021 LNCS, no. i, pp. 554–568, 2008.

[136] M. Debajyoti, B. Aritra, M. Sreemoyee, B. Jhilik, and Y. Kim, “A domain specific ontology based semantic web search engine,” *CoRR*, vol. abs/1102.0695, 2011.

[137] Y. Lei, V. Uren, and E. Motta, “Semsearch: A search engine for the semantic web,” in *Proc. 5th International Conference on Knowledge Engineering and Knowledge Management Managing Knowledge in a World of Networks, Lect. Notes in Comp. Sci., Springer, Podebrady, Czech Republic*, pp. 238–245, Springer-Verlag, 2006.

[138] J. Heflin and J. Hendler, “Searching the web with shoe,” in *In Artificial Intelligence for Web Search. Papers from the AAAI Workshop. WS-00-01*, pp. 35–40, AAAI Press, 2000.

[139] A. Hogan, A. Harth, J. Umbrich, S. Kinsella, A. Polleres, and S. Decker, “Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, pp. 365–401, dec 2011.

[140] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, “Swoogle: A search and metadata engine for the semantic web,” in *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM ’04, pp. 652–659, ACM, 2004.

[141] C. Ziqiang, “Uncertain temporal knowledge management,” in *Proceedings Fourth International Conference on Software Engineering and Knowledge Engineering*, pp. 94–100, Jun 1992.

[142] L. Deng, Y. Cai, C. Wang, and Y. Jiang, “Fuzzy temporal logic on fuzzy temporal constraint networks,” in *Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery - Volume 6*, FSKD’09, (Piscataway, NJ, USA), pp. 272–276, IEEE Press, 2009.

[143] A. M. Lai, S. Parsons, and G. Hripcsak, “Fuzzy temporal constraint networks for clinical information.,” in *AMIA Annual Symposium proceedings. AMIA Symposium (2008)*, pp. 374–378, February 2008.

[144] L. Godo and L. Vila, “Possibilistic temporal reasoning based on fuzzy temporal constraints,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI’95, (San Francisco, CA, USA), pp. 1916–1922, Morgan Kaufmann Publishers Inc., 1995.

[145] S. Schockaert, M. D. Cock, and E. E. Kerre, “Fuzzifying allen’s temporal interval relations,” *IEEE Transactions on Fuzzy Systems*, vol. 16, pp. 517–533, April 2008.

[146] S. Schockaert and M. De Cock, "Temporal reasoning about fuzzy intervals," *Artif. Intell.*, vol. 172, pp. 1158–1193, May 2008.

[147] D. Dubois, A. HadjAli, and H. Prade, "Fuzziness and uncertainty in temporal reasoning," *J. UCS*, vol. 9, no. 9, p. 1168, 2003.

[148] N. A. A. Manaf and M. R. Beikzadeh, "Representation and reasoning of fuzzy temporal knowledge," in *2006 IEEE Conference on Cybernetics and Intelligent Systems*, pp. 1–6, June 2006.

[149] V. Milea, F. Frasincar, and U. Kaymak, "towl: A temporal web ontology language," *Trans. Sys. Man Cyber. Part B*, vol. 42, pp. 268–281, Feb. 2012.

[150] "An Ontology of Time for the Semantic Web," *ACM Transactions on Asian Language Information Processing*, vol. 3, no. 1, pp. 66–85, 2004.

[151] Q. Zhou and R. Fikes, "A reusable time ontology," in *Proceeding of the AAAI Workshop on Ontologies for the Semantic Web*, 2002.

[152] S. J. Cox, "Time ontology extended for non-Gregorian calendar applications," *Semantic Web*, vol. 7, pp. 201–209, feb 2016.

[153] F. Pan, "A temporal aggregates ontology in owl for the semantic web," in *Proceedings of the AAAI fall symposium on agents and the semantic web*, pp. 30–37, 2005.

[154] C. Tao, W. Wei, G. Savova, and C. Chute, "A semantic web ontology for temporal relation inferencing in clinical narratives," in *Proceedings of the American Medical Informatics Association (AMIA) 2010 Annual Symposium. Washington DC*, 2010.

[155] E. Anagnostopoulos, E. G. M. Petrakis, and S. Batsakis, "Chronos: Improving the performance of qualitative temporal reasoning in owl," in *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, pp. 309–315, Nov 2014.

[156] G. Nagypál and B. Motik, "A fuzzy model for representing uncertain , subjective and vague temporal knowledge in ontologies," *The Move to*

Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE (2003), vol. 2888, pp. 906–923, 2003.

[157] J. Allen, "Maintaining knowledge about temporal intervals.," *Communications of ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[158] R. N. Sahay, *An Ontological Framework for Interoperability of Health Level Seven (HL7) Applications: The PPEPR Methodology and System*. PhD thesis, NUI) Galway, Ireland, 2012.

[159] M. Fernández-López, A. Gómez-Pérez, and N. Juristo, "Methontology: from ontological art towards ontological engineering," in *Proc. Symposium on Ontological Engineering of AAAI*, 1997.

[160] H. S. Pinto, C. Tempich, and S. Staab, "Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engingeering of ontologies," in *Proceedings of the 16th European Conference on Artificial Intelligence*, pp. 393–397, IOS Press, 2004.

[161] C. Bezerra, F. Freitas, and F. Santana, "Evaluating ontologies with competency questions," in *International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 3, pp. 284–285, Nov 2013.

[162] N. F. Noy, "Ontology management with the PROMPT plugin," in *Proceedings of the 7th International Protégé Conference*, July 2004.

[163] D. Gromann, "Terminology-Based Patterns for Natural Language De nitions in Ontologies 2 Natural Language De nition ODP," *CEUR Workshop Proceedings*, vol. Vol-1188, 2009.

[164] H. Bohring and S. Auer, "Mapping xml to owl ontologies," *Leipziger Informatik-Tage*, 2005.

[165] H. Wache, T. Voegele, T. Visser, H. Stuckenschmidt, H. Schuster, G. Neumann, and S. Huebner, "Ontology-based integration of information - a survey of existing approaches," in *IJCAI-01 Workshop: Ontologies and Information* (H. Stuckenschmidt, ed.), pp. 108–117, 2001.

[166] I. F. Cruz and H. Xiao, “The role of ontologies in data integration,” *Journal of Engineering Intelligent Systems*, vol. 13, pp. 245–252, 2005.

[167] H. Steve and S. Andy, “Sparql 1.1 query language: W3c recommendation,” 2013.

[168] Jena-ARQ, “Arq - extending query execution,” 2015. [Online; accessed 3-February-2016].

[169] A. Stoughton, “A functional model-view-controller software architecture for command-oriented programs,” in *Proceedings of the ACM SIGPLAN Workshop on Generic Programming*, WGP ’08, (New York, NY, USA), pp. 1–12, ACM, 2008.

[170] M. Horridge and S. Bechhofer, “The owl api: A java api for owl ontologies,” *Semant. web*, vol. 2, pp. 11–21, Jan. 2011.

[171] K. Holger, “Protege-owl api programmer’s guide,” 2010.

[172] A. Jena, “Apache jena - a free and open source java framework for building semantic web and linked data applications,” 2015.

[173] W. Waterfeld, M. Weiten, and P. Haase, “Ontology management infrastructures.,” in *Ontology Management* (M. Hepp, P. D. Leenheer, A. de Moor, and Y. Sure, eds.), vol. 7 of *Semantic Web And Beyond Computing for Human Experience*, pp. 59–87, Springer, 2008.

[174] S. Bechhofer, R. Volz, and P. Lord, “Cooking the semantic web with the owl api,” pp. 659–675, Springer, 2003.

[175] K. Cerans, G. Barzdins, R. Liepins, J. Ovcinnikova, S. Rikacovs, and A. Sprogis, “Graphical schema editing for stardog owl/rdf databases using owlgred/s.,” in *OWLED* (P. Klinov and M. Horridge, eds.), vol. 849 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2012.

[176] C. J. Swanevelder, “Bambara food for Africa,” *National Department of Agriculture ARC - Grain Crops Institute*, pp. 5–16, 1998.

[177] G. Stoilos, G. Stamou, V. Tzouvaras, J. Z. Pan, and I. Horrocks, "Fuzzy OWL: Uncertainty and the Semantic Web," *CEUR Workshop Proceedings*, vol. 188, 2005.

[178] J. Pan, G. Stoilos, and G. Stamou, "f-SWRL: A fuzzy extension of SWRL," *Journal on Data . . .* , no. Icann, pp. 1–12, 2006.

[179] R. T. Snodgrass, M. H. Böhlen, C. S. Jensen, and A. Steiner, "Adding valid time to sql/temporal," *ANSI X3H2-96-501r2, ISO/IEC JTC*, vol. 1, 1996.

[180] W. KARWOWSKI and R. W. EVANS, "Fuzzy concepts in production management research: a review," *International Journal of Production Research*, vol. 24, pp. 129–147, jan 1986.

[181] L. A. Zadeh, "Fuzzy Sets *," *INFORMATIO AND CONTROL*, vol. 8, pp. 338–353, 1965.

[182] J. Miliauskaite, "The membership function construction in view-based framework," in *11th International Baltic Conference on Database and Information Systems (Baltic DB&IS 2014)*, (Tallinn:), pp. 125–132, Tallinn University of Technology Press, 2014.

[183] M. O'Connor and A. Das, "SQWRL: A query language for OWL," in *CEUR Workshop Proceedings*, vol. 529, 2009.

[184] A. Lawan, A. Rakib, N. Alechina, and A. Karunaratne, "Advancing underutilized crops knowledge using SWRL-enabled ontologies - A survey and early experiment," vol. 1312, pp. 69–84, 2014.

[185] N. Bansal and S. K. Malik, "A framework for agriculture ontology development in semantic web," in *Proceedings - 2011 International Conference on Communication Systems and Network Technologies, CSNT 2011*, pp. 283–286, 2011.

[186] M. OConnor and A. Das, "A mechanism to define and execute swrl built-ins in protege-owl," in *9th International Protege Conference, Stanford, California*, July 2006.

[187] S. Kim, M. Iglesias-Sucasas, and V. Viollier, "The fao geopolitical ontology: a reference for country-based information," *Journal of Agricultural & Food Information*, vol. 14, no. 1, pp. 50–65, 2013.

[188] L. Cooper, R. L. Walls, J. Elser, M. A. Gandolfo, D. W. Stevenson, B. Smith, J. Preece, B. Athreya, C. J. Mungall, S. Rensing, *et al.*, "The plant ontology as a tool for comparative plant anatomy and genomic analyses," *Plant and Cell Physiology*, vol. 54, no. 2, pp. e1–e1, 2013.

[189] L. Matteis, P.-Y. Chibon, H. Espinosa, M. Skofic, R. Finkers, R. Bruskiewich, G. Hyman, and E. Arnaud, "Crop ontology: vocabulary for crop-related concepts," *Semantics for Biodiversity (S4BioDiv 2013)*, p. 37, 2013.

[190] K. S. Iglesias-Sucasas, M. and V. Viollier, "The fao geopolitical ontology: a reference for country-based information," 2015.

[191] X. Rong, A. Fourney, R. N. Brewer, M. R. Morris, and P. N. Bennett, "Managing uncertainty in time expressions for virtual assistants," 2017.

[192] A. Lawan, A. Rakib, N. Alechina, and A. Karunaratne, "Advancing underutilized crops knowledge using swrl-enabled ontologies-a survey and early experiment," in *The 2nd International Workshop on Linked Data and Ontology in Practice (LDOP 2014)*, vol. 1312, pp. 69–84, CEUR Workshop Proceedings (CEUR-WS.org) http://ceur-ws.org/Vol-1312/, 2014.

[193] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 51 – 53, 2007. Software Engineering and the Semantic Web.

[194] K. Elbedweihy, S. N. Wrigley, and F. Ciravegna, "Evaluating semantic search query approaches with expert and casual users," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7650 LNCS, no. PART 2, pp. 274–286, 2012.

[195] D. Harman, "Is the cranfield paradigm outdated?," in *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, (New York, NY, USA), pp. 1–1, ACM, 2010.

[196] D. Harman, "The text retrieval conferences (trecs): Providing a test-bed for information retrieval systems," *Bulletin of the American Society for Information Science and Technology*, vol. 24, no. 4, pp. 11–13, 1998.

[197] K. Elbedweihy, S. N. Wrigley, F. Ciravegna, D. Reinhard, and A. Bernstein, "Evaluating semantic search systems to identify future directions of research," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7540, no. Iwest, pp. 148–162, 2015.

[198] D. Salah, R. Paige, and P. Cairns, *An Evaluation Template for Expert Review of Maturity Models*. Springer International Publishing, 2014.

[199] M. J. O'Connor and A. Das, "Acquiring OWL ontologies from XML documents," *Proceedings of the sixth international conference on Knowledge capture - K-CAP '11*, p. 17, 2011.