



AGH University of Science and Technology

Computer Science Laboratory

Department of Automatics

Al. Mickiewicza 30

30-059 Kraków, POLAND

**Overview of selected approaches to rule representation
on the Semantic Web**

Weronika T. Adrian, Grzegorz J. Nalepa, Krzysztof Kaczor, Marta Noga

AGH University of Science and Technology

Department of Automatics

Kraków, POLAND

wta, kk, gjn@agh.edu.pl, marnog@student.agh.edu.pl

Published online: 4.10.2010

CSL Technical Report No. 2/2010

© by AGH University of Science and Technology, Kraków, Poland, 2008-9.

Computer Science Laboratory Technical Reports (CSL TR) series serves as an accessible platform for publishing research results of the CS Lab staff members at Department of Automatics of AGH University of Science and Technology.

See the <http://cslab.ia.agh.edu.pl/csltr:start> for the series homepage.

The editorial board:

- *Main Editor*: Marcin Szpyrka, Ph. D., D. Sc.
- Prof. Antoni Ligeza, Ph. D.
- Prof. Tomasz Szmuc, Ph. D.
- Grzegorz J. Nalepa, Ph. D.

Cover design: Marcin Szpyrka, Ph. D., D. Sc.

L^AT_EXclass design: Marcin Szpyrka, Ph. D., D. Sc.

Contact us at: mszpyrka@agh.edu.pl

Overview of selected approaches to rule representation on the Semantic Web*

Weronika T. Adrian, Grzegorz J. Nalepa, Krzysztof Kaczor, Marta Noga

AGH University of Science and Technology

Department of Automatics

Kraków, POLAND

wta, kk, gjn@agh.edu.pl, marnog@student.agh.edu.pl

Abstract. The aim of this report is to give an overview of the research on rules in the Semantic Web. The report outlines theoretical foundations of rule-based knowledge representation, Horn logic and logic programming. It also explains the motivation and challenges for integrating rules and ontologies within the Semantic Web. In the report several research directions are investigated. A spectrum of solutions is presented, concerning various rule languages and systems. The main characteristics and differences are explained. Another perspective presented in the report concerns practical implementations and applications. Selected examples of semantic wiki systems using rules are examined. Approaches to combine classical rule engines with ontologies are briefly presented.

Keywords: semantic web, rules, ontologies, owl, rdf, swrl, ruleml, rif, semantic wikis, clips, jess, drools

*The paper is supported by the BIMLOQ project funded with 2010-2012 state resources for science as a research project.

Contents

1	Introduction	3
2	Rules and Logic	4
2.1	Horn Clausal Logic and Logic Programs	4
2.2	Various Sorts of Rules and Inference Strategies	5
2.3	Datalog	6
2.4	F-Logic	7
3	Approaches to Rule Representation for the Semantic Web	8
3.1	Combining Rules with Ontologies	8
3.1.1	Challenges for Rules and Ontologies Integration	8
3.1.2	Hybrid Approach and Homogeneous Approach	10
3.2	Overview of the selected Rule Systems and Languages for the Semantic Web	10
3.2.1	\mathcal{AL} -log	10
3.2.2	CARIN	11
3.2.3	Answer Set Programming	11
3.2.4	Description Logic Programs	11
3.2.5	Web Rule Language (WRL)	12
3.2.6	Semantic Web Rule Language (SWRL)	12
3.2.7	DL-safe Rules	13
3.2.8	TRIPLE Language	13
3.3	Rule Markup and Interchange	13
3.3.1	RuleML	14
3.3.2	Rule Interchange Format	16
3.3.3	R2ML Language	19
3.4	Summary	19
4	Rule Representation in Semantic Wikis	20
4.1	AceWiki	20
4.2	KnowWE	23
4.3	SMW+	24
4.4	TaOPis	26
4.5	Internet Business Logic	27
4.6	PIWiki	29
4.7	Summary	31
5	Combining Classical Rule-based Systems and Ontologies	31
5.1	R-DEVICE	31
5.2	O-DEVICE	36
5.3	Integrating ontologies and rules within Protegé ontology editor	37
5.3.1	Historical solutions	37
5.3.2	SWRL Bridge in Protegé	37
5.3.3	Combining OWL and SWRL with Jess	38
5.3.4	DroolsTab for Protegé	40
6	Summary and trends	41

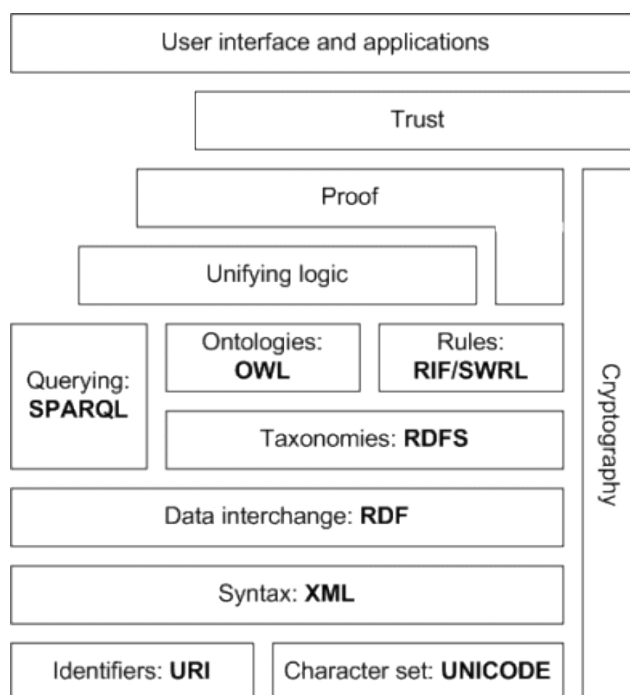


Figure 1: Semantic Web stack

1 Introduction

Rules constitute a powerful and flexible knowledge representation method. Rules in computer systems are based on various logical calculi, have different forms and usage scenarios. Rule-based systems constitute a mature technology, with numerous optimized algorithms and implementations.

The Semantic Web is a worldwide initiative inspired by the vision presented in [8]. The main idea of the Semantic Web is to represent the meaning of data stored on the Internet in a standardized form. Such representation should be possible for the machines to interpret and process. This would enable more advanced searching and planning mechanisms done in an automated way. Semantic Web technologies include set of knowledge representation standards, each with different goals and expressive power. They work in a layered architecture, on consequent levels of abstraction (see Fig. 1). One of the general-purpose languages for representing information in the Web is RDF [45]. With the help of RDF the information can be described using statements about resources in the form of triples (subject, object, predicate). What is more, the description of only resources is not enough. Hence, on the higher level of abstraction the formal definitions of relations between resources classes are described with the help of ontologies. The main formalism for ontologies on the Semantic Web are Description Logics (DL) [4]. They use subsets of Predicate Calculus to express relationships among concepts and individuals in the conceived universe. The Web Ontology Language is called OWL [61] and is based on the Description Logics variant.

Rules on the Semantic Web is an active research area. On one hand the motivation for using rules lays in possibility of augmenting the knowledge representation. On the other, the dynamic nature of the Web requires some sort of actions which may be defined by means of rules. Moreover, there is an increase in the number of applications (often Web-based) which use rules to encode their behavior. It would be useful to enable for interoperability between them. In [14] several usage scenarios with rules are described, including: defining user views over ontologies, interoperability among different

data sources, personalization Services for the Semantic Web and Ontology-based Resource Matching.

There exist concrete challenges for rule representation on the Semantic Web. A number of solutions and proposal has been formulated. Theoretical solutions may be implemented in various semantic systems. One of the applications is using rules in semantic wikis, collaborative systems for knowledge authoring and sharing. Combining rules with existing Semantic Web technologies may require interaction between a rule components and an ontology. Classical rule engines may be adapted to cooperate with knowledge representation with ontologies.

The report is organized as follows. In this Section the motivation for using rules in the Web context is shortly explained. Horn logic and various sorts of rules are introduced in Section 2. Extensive Section 3 deals with *state-of-the-art* of research on rule representation for the Semantic Web. The issues of combining rules with ontologies is shortly discussed in Section 3.1, followed by a set of chosen rule systems and languages in Sect. 3.2. Markup languages for the Web are also presented in Sect. 3.3. Two following Sections deal with practical applications of rules within the Semantic Web context. In Section 4 the rule representation in semantic wikis is discussed. Section 5 surveys the implementations combining classic rule-based systems with ontologies. The report is summarized in Section 6.

2 Rules and Logic

2.1 Horn Clausal Logic and Logic Programs

In logic a *clause* is a disjunction of literals. They are used both in propositional and predicate calculus. Clauses are used in resolution theorem proving [41]. A special kind of clauses, namely the *Horn clauses* has been pointed to be very useful by Horn in [30]. According to [57, 58],

Definition 1 A clause (i.e., a disjunction of literals) is called a *Horn clause* if it contains at most one positive literal. Horn clauses are usually written as

$$A_1, \dots, A_n \rightarrow A \quad (\equiv \neg A_1 \vee \dots \vee \neg A_n \vee A) \quad (1)$$

or

$$A_1, \dots, A_n \rightarrow (\equiv \neg A_1 \vee \dots \vee \neg A_n), \quad (2)$$

where $n \geq 0$ and A is the only positive literal.

Definition 2 A definite clause is a *Horn clause* that has exactly one positive literal.

Definition 3 A *Horn clause* without a positive literal is called a *goal*.

Definition 4 A *Horn clause* with no positive literals

$$\neg A_1 \vee \dots \vee \neg A_n \quad (3)$$

can be written as

$$A_1 \wedge \dots \wedge A_n \rightarrow False \quad (4)$$

Such sentences are called *integrity constraints*.

Horn clauses express a subset of statements of first-order logic. They are used extensively in logic programming. They are very useful in rule-based systems, because "Every Horn clause can be written as an implication whose premise is a conjunction of positive literals and whose conclusion is a single positive literal" [57].

The idea of logic programming is to use a computer for drawing conclusions from declarative descriptions. It has its roots in the research on automatic theorem proving [49]. Declarative logic programs (LP) is a knowledge representation, whose semantics underlies in a large part several rule systems, including SQL relational databases and Prolog [13]. Prolog is built on Horn clauses. Prolog programs are composed of definite clauses and any question in Prolog is a goal [58].

Generally, logic programs consist of definite clauses. According to [23],

Definition 5 An ordinary *Logic Program* is a set of rules each having the form:

$$H \leftarrow B_1 \wedge \dots \wedge B_m \wedge \neg B_{m+1} \wedge \dots \wedge \neg B_n \quad (5)$$

where H , B_i are atoms (atomic formulae), and $n \geq m \geq 0$. H is called the head (or consequent) of the rule; $B_1 \wedge \dots \wedge B_m \wedge \neg B_{m+1} \wedge \dots \wedge \neg B_n$ is called the body (or antecedent) of the rule.

Such rules can be transformed into

$$H \vee B_{m+1} \dots \vee B_n \leftarrow B_1, \dots, B_m \quad (6)$$

These rules are called disjunctive rules [47].

Definition 6 A definite *Logic Program* is an ordinary *Logic Program* in which negation-as-failure does not appear, i.e. a set of rules each having a form:

$$H \leftarrow B_1 \wedge \dots \wedge B_m \quad (7)$$

where H , B_i are atoms (atomic formulae), and $m \geq 0$. Such rules are called non-disjunctive rules.

Definite Logic Programs are closely related syntactically and semantically to the Horn fragment of First Order Logic, Horn Clausal Logic. However, the expressiveness of Logic Programming is not entirely within the expressiveness of FOL (see Fig. 2). For instance, procedural attachments and negation-as-failure are features not expressible in Predicate Calculus.

2.2 Various Sorts of Rules and Inference Strategies

In various rule-based systems different reasoning techniques are used. Two main inference schemes are backward chaining and forward chaining inference [57, 12]. *Backward-chaining*, or *goal-driven* inference is used in diagnostic systems, where the number of hypothesis is limited, and the system tries to prove each of them. During reasoning the system gathers necessary information. The other inference scheme is *forward-chaining* inference, or *data-driven* inference. This kind of reasoning is used in the systems, where the number of possible solutions is huge or unlimited. In order to avoid the computational explosion, the data driven approach executes only certain rules.

Rules may be classified as derivation, production or reaction ones. Derivation rules may be used in both forward- and backward-chaining reasoning. They define what conclusions may be derived from the given premises. These conclusions serve to prove the goals, stated in the program or generate new facts based on those stored in the knowledge base. In forward chaining approach *production rules* and *reaction rules* are more appropriate. Once a rule is fired, the action specified in its head is to be executed or an inferred fact is to be asserted in a knowledge base. The reaction rules are also called "ECA" – "event-condition-action", or "trigger" rules [10].

In the context of Semantic Web all kinds of rule may enhance the Web pages and XML documents. As pointed out in [10], "Derivation rules allow the dynamic inclusion of derived facts, while reaction rules allow the specification of behavior in response to browser events".

The importance of the difference between the derivation and production, or reaction rules is emphasized in the design of various rule languages. An example may be the Rule Interchange Format (RIF) with its variant dialects (see Section 3.3.2). The distinction between them is also important with respect to the rule *monotonicity*.

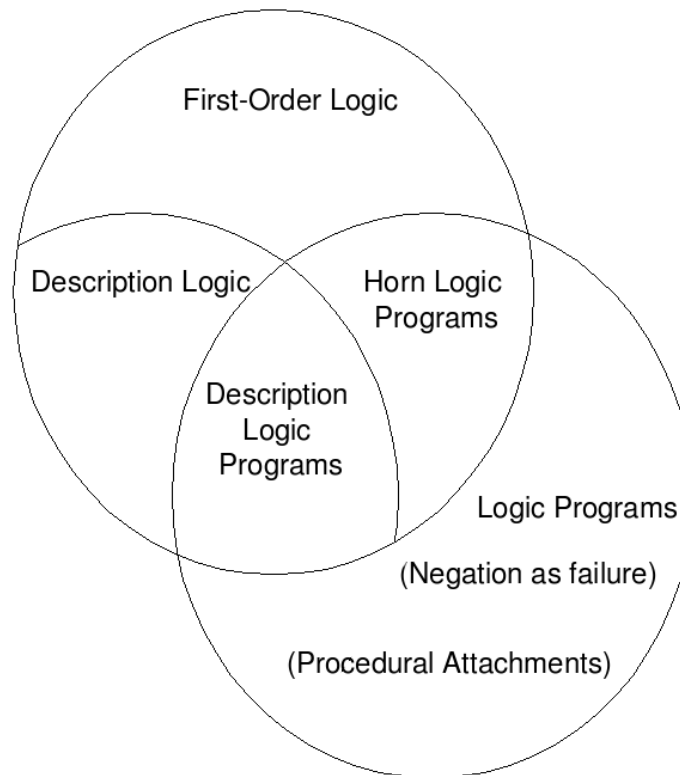


Figure 2: Expressive overlap of DL with LP (based on [23]).

Monotonicity of rules means that adding a fact to the knowledge base will not make a previously valid conclusion invalid. Monotonic rules are a subset of predicate calculus called Horn Logic [2]. Monotonic rule languages include Datalog [60], DLP [23] and SWRL [33]. On the other hand, if a change in the knowledge base (for instance a new assertion) changes the result of the inferencing the rules are non-monotonic. These sort of rules is useful when the information is incomplete. In the case of non-monotonic rules, even if all the premises hold rule may not be fired. The rules are defeasible, because they can be defeated (overridden) by other rules. To resolve potential conflicts between the rules, priorities are usually defined. The priorities may be assigned based on the source of the rules, the fact that one is more recent than another, or more specific and thus stronger. Non-monotonic reasoning is used e.g. in Prolog.

Another sort of rules is represented by so-called *integrity constraints*. They specify the constraints of objects in a formal way in order to ensure the accuracy and consistency of e.g. relational databases and Unified Modeling Language (UML) [51] models. A dedicated language for specifying constraints for objects in the UML models is Object Constraint Language (OCL) [43].

2.3 Datalog

Datalog is a "syntactically restricted subset of logic programming where function symbols are not allowed" [14]. Datalog was originally defined as a query and rule language for *deductive databases*. Query evaluation in Datalog usually follows bottom-up strategies, which operate efficiently even for large databases.

The vocabulary of the language consists of predicate symbols, constants and variables. Datalog does not allow function predicates. In its pure form it also does not allow negation in the rules body. However, a negated fact may be deduced, i.e., the presence of a negated literal in the head of the rule is possible [15]. Atoms and clauses are defined as in the Horn Logic. Datalog knowledge base is a

set of clauses. There are two sorts of the clauses: facts and rules.

The following example is a Witch-proof based on Monty Python's Episode in "Holy Grail".¹ Set of rules and facts in a pseudo-code is as follows:

```

Rule [1]:
  IF BURNS(x) AND WOMAN(x) THEN WITCH(x)
Fact [2]:
  WOMAN(GIRL)
Rule [3]:
  FOR ALL x, IF ISMADEOFWOOD(x) THEN BURNS(x)
Rule [4]:
  FOR ALL x, IF FLOATS(x) THEN ISMADEOFWOOD(x)
Fact [5]:
  FLOATS(DUCK)
Rule [6]:
  FOR ALL x,y IF FLOATS(x) AND SAMEWEIGHT(x,y) THEN FLOATS(y)
% and, by experiment
Fact [7]:
  SAMEWEIGHT(DUCK,GIRL)

```

This set of facts and rules in Datalog looks as follows:

```

witch(X) <- burns(X), woman(X).
woman(girl).
burns(X) <- ismadeofwood(X).
ismadeofwood(X) <- floats(X).
floats(duck).
floats(Y) <- floats(X), sameweight(X,Y).
sameweight(duck,girl).

```

The applications of Datalog are mostly developed at research and university centers. The main application area is conceptual modeling and deductive querying of databases. There exist several implementation of deductive databases, such as bddbldb – BDD-Based Deductive DataBase,² ConceptBase,³ or DES – Datalog Education System.⁴ Several extensions to pure Datalog has been proposed, for example use of function symbols and built-in predicates in IRIS reasoner,⁵ or disjunctive head clauses in DLV.⁶ Datalog has been implemented in XSB – a Logic Programming and Deductive Database system.⁷ XSB is a Prolog like language, extended with tabled resolution and support for higher order logic programming [59].

2.4 F-Logic

Frame logic or F-Logic for short [35] is a formalism designed for object-oriented and frame-based systems. According to [35], "F-Logic stands in the same relationship to the object-oriented paradigm as classical predicate calculus stands to relational programming". There are several rule languages, whose semantics is based on F-Logic (see Sect. 3.1).

¹Based on: <http://www.netfunny.com/rhf/jokes/90q4/burnher.html>.

²See <http://bdbddb.sourceforge.net>.

³See <http://conceptbase.cc/>.

⁴See <http://des.sf.net>.

⁵See <http://iris-reasoner.org/>.

⁶See <http://www.dbai.tuwien.ac.at/proj/dlv/>.

⁷See <http://xsb.sourceforge.net/>.

One of them is OntoBroker,⁸ a Semantic Web Middleware supporting OWL, RDF, RDFS and SPARQL, as well as implementing an F-Logic inference engine. The system allows for developing ontology-based applications, and integrating data from existing databases. Other projects include:

- FLORA-2⁹ – An Object-Oriented Knowledge Base Language and application development environment developed at Stony Brook University. FLORA-2 language is a dialect of F-Logic which also supports HiLog [16]. The system is based on XSB engine. FLORA-2 is an Open Source software.
- TRIPLE¹⁰ – RDF query, inference, and transformation language, open source developed at Stanford and ISI.
- FLORID/FloXML¹¹ – FLORID was one of the early implementations of F-logic. Currently the development is carried on by Freiburg University.
- KAON2¹² – an infrastructure for managing OWL-DL, SWRL, and F-Logic ontologies, produced by the joint effort of Information Process Engineering (IPE)¹³ at the Research Center for Information Technologies (FZI), Institute of Applied Informatics and Formal Description Methods (AIFB)¹⁴ at the University of Karlsruhe, and Information Management Group (IMG)¹⁵ at the University of Manchester. KAON2 supports the function-free subset of F-Logic, with limited support for default negation.

3 Approaches to Rule Representation for the Semantic Web

3.1 Combining Rules with Ontologies

Rules and Ontologies are complimentary approaches to knowledge representation and reasoning [27]. In ontologies one can capture class properties and define complex classes. Rule languages are designed mainly to define how to synthesize new facts from those stored in the knowledge base. There are things that rules cannot really express or infer, like inferencing complex relationships among classes. Generally asserting negation (complement of classes) or disjunctive information or existential quantification is not possible as well [2]. On the other hand, the axioms possible to express in DL have certain tree structure [23]. Thus, there are things that cannot be expressed in ontologies or only in a very complicated manner, for example complex Horn rules (see Section 2). Various use-cases have shown that applications often require both approaches, the DL one and the rule one.

3.1.1 Challenges for Rules and Ontologies Integration

There are some important differences between ontologies and logic programming or rule-based systems. They include the following issues:

Description Logics and Horn logic (rule systems) are orthogonal in the sense that neither of them is a subset of the other [2].

Unique Name Assumption (UNA) in logic programming denotes that an object can be identified by a name. In ontologies and Description Logics the UNA does not hold, and the same resource may be referenced to by different names and descriptions [26].

⁸See <http://www.ontoprise.de/en/home/products/ontobroker/>.

⁹See <http://flora.sourceforge.net/>.

¹⁰See <http://triple.semanticweb.org/>.

¹¹See <http://user.informatik.uni-goettingen.de/~may/florid/>.

¹²See <http://kaon2.semanticweb.org/>.

¹³See <http://www.fzi.de/ipe/eng/index.html>.

¹⁴See <http://www.aifb.uni-karlsruhe.de/english>.

¹⁵See <http://img.cs.manchester.ac.uk/>.

Closed-World Assumption (CWA) in databases and logic programming systems. The main difference between these approaches is the attitude to the incomplete knowledge. If a statements cannot be proved to be true it is assumed false in systems with Closed-World Assumption. Contrary, Open-World Assumption does not draw a conclusion of falsity of such a statement [26].

The choice between open world semantics and closed world semantics, as well as Unique Name Assumption is a recurring issue in the discussion on the reasoning for the Semantic Web. "Traditionally, systems such as databases and logic programming systems have tended to support closed-worlds and unique names, while knowledge representation systems and theorem provers support open-worlds and non-unique names" [26]. In his paper [32] I. Horrocks sees this lack of agreement as an important danger for the Semantic Web. He claims that turning to closed-world assumption and unique name assumption, which is not compatible with RDF(S) and OWL semantics will lead to an emergence of two concurrent versions of the Semantic Web. These two Semantic Webs would have little or no semantic interoperability, even at the RDF level. This results in building two towers of the Semantic Web (see Fig. 3).

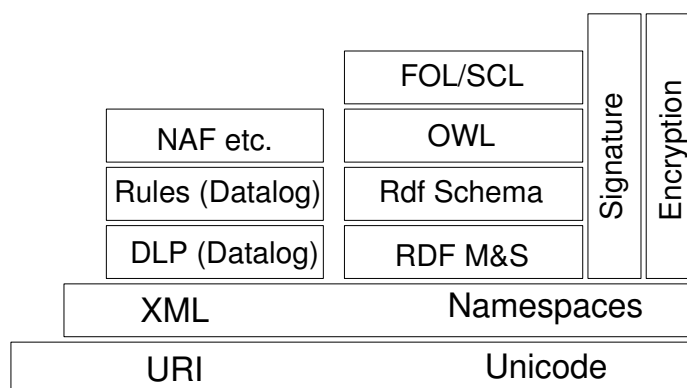


Figure 3: Alternative versions of the semantic stack - "two towers" [32].

On the other hand, closed-world assumption, though not universal, seems appropriate for certain use case scenarios. In constraint checking, database-like queries and tasks it is natural to take the closed-world approach. A unified logic framework for both open- and closed-world assumptions is concerned, but it is not clearly yet, how this can be achieved [32].

Challenges for the reasoning on the Web include the complexity, decidability and tractability of the system resulting from combining rules and ontologies. A number of approaches has been proposed. Selected proposals are described in Section 3.1.

Overcoming the OWL expressiveness limitation can be basically acquired in two ways. "One way to address this problem would be to extend OWL with a more powerful language for describing properties (...) An alternative way to overcome some of the expressive restrictions of OWL would be to extend it with some form of *rules language*" [31].

First attempts to combine rules with Description Logics were conducted in the CLASSIC system [4]. The rules, however, were given "a weaker semantic treatment than axioms asserting sub- and super-class relationships; they were only applied to individuals, and did not affect class-based inferences such as the computation of the class hierarchy" [31].

OWL-DL and function-free Horn rules are decidable fragments of first-order logic with interesting, yet orthogonal expressive power" [47]. The combination of them may be homogeneous or hybrid (heterogeneous). One solution is to create a unified language and a uniform reasoner, and the other to keep modularity and allow separate reasoners to make inference in their parts.

A comprehensive study of the effects of combining Description Logics with Datalog rules is presented in [40] and summarized in [47] as follows:

- Answering conjunctive queries over $\mathcal{ALCN}\mathcal{R}$ DL (sort of Description Logic, see [4] for explanation) knowledge bases is decidable.
- Query answering in the extension of $\mathcal{ALCN}\mathcal{R}$ DL with non-recursive datalog rules, where both concepts and roles can occur only in the body of rules is decidable.
- If rules are recursive, answering queries becomes undecidable.
- Decidability can be regained by disallowing certain combinations of constructors in the logic.
- Decidability can be regained by requiring rules to be role-safe, i.e. at least one variable from each role literal must occur in some non-DL-atom.

3.1.2 Hybrid Approach and Homogeneous Approach

Heterogeneous (modular) approach offers loose integration through strict semantic separation. The resulting system consists of an ontology component based on a DL variant, and a rule component, which usually is a variant of Datalog. The flow of information may be unidirectional or bidirectional. In the unidirectional one, the rules are interpreted such that they must satisfy the ontology constraints (predicates). DL entailments are inputs to the rule reasoner. Examples of such systems are \mathcal{AL} -log, and CARIN. If the flow of information is bidirectional, then iterative reasoning is performed on both components until no more conclusions can be drawn. Examples of this approach include DL+log [56] and Answer Set Programming [20, 19]. Hybrid systems are characterized by high, intractable worst-case complexity.

Homogeneous approach results in designing a single logical language. No syntactic or semantic distinctions are made between the ontology and the rule part, both can be interpreted by the same reasoning engine. A basic idea consists in a mapping (typically recursive) from one language to the other. The language is typically either an expressive union of the component languages or the intersection of them. Expressive union, the union of the entire LP and DL fragments within FOL, often leads to undecidability. Examples include Description Logic Programs (DLP) [23], Semantic Web Rule Language (SWRL) [33] and Web Rule Language (WRL) [1].

3.2 Overview of the selected Rule Systems and Languages for the Semantic Web

Numerous solutions for rule languages and systems have been proposed. They differ in terms of syntax and semantics, as well as the overall system architecture, including interaction between rules and ontologies. In this section selected proposals are

3.2.1 \mathcal{AL} -log

\mathcal{AL} -log [18] is a hybrid system combining \mathcal{ALC} Description Logic and Datalog. It comprises of a structural part (TBox and ABox expressed in \mathcal{ALC}) and a deductive one (Datalog rules). DL concepts appear as predicates in the body of rules. Rule safety conditions are introduced to maintain decidability: each variable that appears in the head of rule must appear in the body of the rule, and only concepts are allowed as constraints in the relational component. Reasoning for \mathcal{AL} -log is based on *constrained SLD-resolution* combined with a tableaux algorithm for \mathcal{ALC} to deal with constraints [47, 42].

3.2.2 CARIN

In CARIN system [40] rules and Description Logics were integrated in a way that sound and complete reasoning was still possible. The system uses $\mathcal{ALCN}\mathcal{R}$ DL, significantly weaker than OWL, but more one that allows typical and popular ontology constructs. CARIN allowed both concepts and roles to appear in the body of rules. It placed severe syntactic restrictions on the occurrence of Description Logic terms in the heads of rules. Reasoning was similar to the one proposed for \mathcal{AL} -log. However, as opposed to \mathcal{AL} -log, the hybridization (interaction between the ontology and rule components) in CARIN is not safe [42].

3.2.3 Answer Set Programming

Another approach is presented by combining DL with Answer Set Programming [20, 19]. Answer Set Programming denotes knowledge representation and declarative programming paradigm which "includes features from non-monotonic logics, as well as support for reasoning with constraints and preferences" [20]. ASP is fully declarative, decidable and well scalable.

The interaction between the subsystems is "enabled by exchanging only unit (i.e. non-disjunctive) ground consequences between the two components" [47]. The resulting semantics is incompatible with First Order semantics.

3.2.4 Description Logic Programs

Description Logic Programs (DLP) [23] are based on the intersection of a Description Logic with Horn Clause rules. The result is a decidable language, which is necessarily less expressive than either the Description Logic or rules language from which it is formed. In DLP it is possible to express:

- triples (aPb) as atoms of the form $P(a, b)$,
- declarations of the type $type(a, C)$ as concept assertions $C(a)$,
- subclass relations ($C \sqsubseteq D$) in a form of a rule $C(X) \rightarrow D(X)$,
- class equivalence `sameClasses` in a form of rules $C(X) \rightarrow D(X), D(X) \rightarrow C(X)$
- property equivalence,
- transitive properties,
- intersection of classes,
- intersection being a subclass, subclass of an intersection,
- union being a subclass.

Disjunction in the head of rules is not allowed in Horn Logic. Therefore, it is not possible to express a subclass of a union.

There are several advantages of the DLP approach. During the process of modeling it is possible to use either OWL or rules. From the implementation perspective, both DL reasoners and deductive rule systems may be used as a reasoning engine. On top of it, experience with OWL has shown that existing OWL ontologies rarely use constructs outside the DLP language [2].

DLP as proposed in [23] have standard First Order semantics and thus do not support Closed World Assumption. However, in the chosen subset, which is an intersection of Logic Programs and OWL, it is possible to treat DLP rules as having Datalog semantics based in Closed World Assumption. In this case, though, they are no longer semantically compatible with OWL nor even RDF [32]. Such an interpretation of DLP leads to the situation presented by Horrocks in [32].

3.2.5 Web Rule Language (WRL)

The Web Rule Language WRL [1] is a rule-based ontology language (as opposed to DL-based ontology language OWL). The authors locate it in the Semantic Web stack next to the OWL (see Fig. 4).

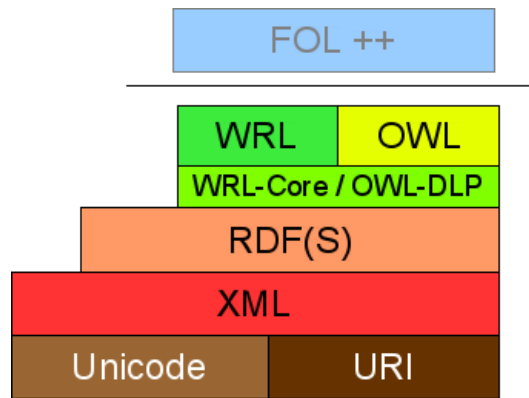


Figure 4: Layering Web Rule Language (WRL) in the semantic stack [1].

WRL enables for designing both ontologies and rules. Its syntax is based on RuleML. Meta model of the WRL ontology consists of concepts, relations, instances, and axioms. The language supports integrity constraints, Closed-World Assumption and Unique Name Assumption.

Three language variants are defined, namely Core, Flight and Full. The Core variant is based on the Description Logic Programs and thus constitutes the basic interoperability layer with OWL. WRL-Flight is a rule language based on the Datalog subset of F-Logic [35], with negation-as-failure under the Perfect Model Semantics [54]. The language allows classical implication and conjunction in the head of a rule and a disjunction in the body of a rule. The most expressive is WRL-Full, based on full Horn with negation-as-failure under the Well-Founded Semantics [22].

3.2.6 Semantic Web Rule Language (SWRL)

SWRL [33] is based on the expressive union of the function-free Horn logic and OWL-DL. It includes a high-level abstract syntax, a model-theoretic semantics, and an XML syntax based on RuleML. The language enables Datalog-like rules to be combined with an OWL knowledge base. Concepts and roles are used in rules as unary and binary atoms. It is worth mentioned that in RuleML predicates may have any arity. In SWRL, which is an extension of OWL with Datalog-like clauses, one can use only unary and binary predicates. "OWL class description can be viewed as unary predicates introduced by definition (...), a concept not supported by RuleML" [14]. Subsumption and query answering with respect to knowledge bases and programs is undecidable.

An *atom* in SWRL is an expression of one of the forms:

```
C(x)
P(x,y)
sameAs(x,y)
differentFrom(x,y)
builtIn(r,,x,...)
```

Rules in SWRL can be expressed as a part of ontology, for example:

```
<owl:Class rdf:about="Witch">
  <owl:intersectionOf>
    <owl:Class rdf:about="Burns"/>
```

```

    <owl:Class rdf:about="Woman"/>
  </owl:intersectionOf>
</owl:Class>

```

which corresponds to a DL formula: $Witch \equiv Burns \sqcap Woman$

Alternatively in SWRL one can formulate a rule, for instance:

```

<swrl:Imp rdf:about="Rule-6">
  <swrl:head>
    ...
  </swrl:head>
  <swrl:body>
  </swrl:body>
</swrl:Imp>

```

In the head of the SWRL rule disjunction of atoms is allowed. However, "this does not increase the expressive power of the language because any such rule may be equivalently replaced by a set of rules with just one atom as consequent" [14].

3.2.7 DL-safe Rules

In order to regain tractability subsets of SWRL has been proposed. Among them *DL-safe* rules have been presented in [47]. The authors "do not restrict the component languages, but only reduce the interface between them" [47]. DL-safe rules are applicable only to explicitly named objects. They are defined as follows [47]:

A (disjunctive) DL rule r is *DL-safe* if each variable occurring in r also occurs in a non-DL-atom in the body of r .

A (disjunctive) program P is DL-safe if all its rules are DL-safe.

DL-safe programs differ from SWRL in that non-DL-atoms are allowed in a rule. Moreover, only atomic concepts are allowed in a rule (however, this is a technical assumption, because one can introduce an additional atomic concepts equivalent to a complex one).

An algorithm for query answering in the *SHIQ* logic extended with DL-safe rules is based on a reduction of the knowledge base to a disjunctive program. A tool supporting *DL-safe* rules and implements the algorithm is KAON2.¹⁶ The algorithm allows applying deductive database techniques, to DL reasoning which results in high optimization of the reasoning.

3.2.8 TRIPLE Language

TRIPLE adopts Datalog-like rules, cast in and RDF-based data model, but it offers more sophisticated constructs than SWRL [14]. TRIPLE is a hybrid rule language that permits interaction with external reasoning components, such as those developed for Description Logic. The language has no fixed semantics for classes and class hierarchies, but it allows such features to be defined within the language. TRIPLE is based on F-Logic [35].

3.3 Rule Markup and Interchange

Many markup languages have been developed and found applications in industry, government, and academia. This began in the 1985-1998 time frame as SGML applications have subsequently migrated to XML using XML DTDs, W3C XML Schemas, or other schema languages. For a list of XML applications and initiatives see <http://xml.coverpages.org/xmlApplications.html>.

¹⁶See <http://kaon2.semanticweb.org/>.

3.3.1 RuleML

RuleML [10] is a markup language based on Datalog, which was designed for publishing and sharing knowledge bases on the web. The language supports a family of rule markup languages, including derivation rules, integrity constraints and reaction rules [2]. The language has been being developed by the Rule Markup Initiative, an open network of individuals and groups from both industry and academia. The authors' intention is to design a "shared Rule Markup Language (RuleML), permitting both forward (bottom-up) and backward (top-down) rules in XML for deduction, rewriting, and further inferential-transformational tasks. The goal of the Rule Markup Initiative is to develop RuleML as the canonical Web language for rules using XML markup, formal semantics, and efficient implementations".¹⁷ The RuleML language offers XML syntax for rules Knowledge Representation, interoperable among major commercial and non-commercial rules systems [10].

The predecessors of the RuleML were:

SRML (Simple Rule Markup Language)¹⁸ – A General XML Rule Representation for Forward-chaining Rules; generic rule language consisting of the subset of language constructs common to the popular forward-chaining rule engines, an XML Rule Representation for Java Rule Engines.

XRML (Extensible Rule Markup Language)¹⁹ – An extension of XML with additional capabilities of representation for rule structure, automatic knowledge processing by exchanged rule, and consistency maintenance of knowledge. The language has three components: the Rule Structure Markup Language (RSML), the Rule Identification Markup Language (RIML), and the Rule Triggering Markup Language (RTML).

The languages are no more supported and has practically been replaced by the RuleML.

RuleML covers the entire rule spectrum, from derivation rules to transformation rules to reaction rules. The kernel of the language is Datalog, for which RuleML provides an XML syntax [2]. An example of a fact and a rule encoded in RuleML is presented below:

```
<Atom>
  <Rel>Woman</Rel>
  <Ind>girl</Ind>
</Atom>
<Implies>
  <head>
    <Atom>
      <Rel>Witch</Rel>
      <var>X</var>
    </Atom>fix-point
  </head>
  <body>
    <And>
      <Atom>
        <Rel>Burns</Rel>
        <var>X</var>
      </Atom>
      <Atom>
        <Rel>Woman</Rel>
```

¹⁷See <http://ruleml.org/>.

¹⁸See <http://xml.coverpages.org/srml.html>.

¹⁹See <http://xml.coverpages.org/extensibleRuleML.html>.

```

    <var>X<var>
  </Atom>
  </And>
</body>
</Implies>

```

RuleML syntax is modularized. The language ”encompasses a hierarchy of rules, from reaction rules (event-condition-action rules), via integrity-constraint rules (consistency-maintenance rules) and derivation rules (implicational-inference rules), to facts (premiseless derivation rules)” [10]. The most general sorts of rules that the language supports are presented in Fig. 5.

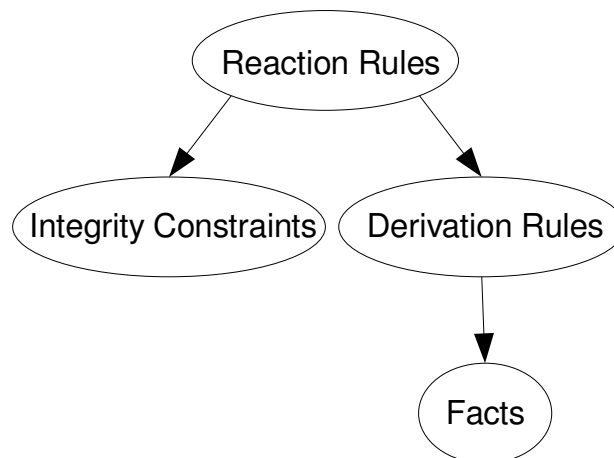


Figure 5: The RuleML hierarchy (based on [10]).

According to [10],

Integrity constraints are considered as ”denials” or special reaction rules whose only possible kind of action is to signal inconsistency when certain conditions are fulfilled.

Derivation rules are considered as special reaction rules whose action happens to only add or ’assert’ a conclusion when certain conditions (premises) are fulfilled. This asserting of conclusions can be regarded as a purely declarative step, as used for model generation and fix-point semantics. Such rules can thus also be applied backward for proving a conclusion from premises.

Facts are considered as special derivation rules that happen to have an empty (hence, *true*) conjunction of premises.

For each kind of rules there exist special RuleML tags (see [10] for more details). The language allows for defining rule priorities, in order to solve potential conflicts.

RuleML supports two kinds of negation, namely *weak* and *strong* negation. The former is used to express *non-truth*, while the latter denotes *falsity* of an expression [10]. In RuleML, the weak negation connective is denoted by `not` and the strong negation connective by `neg`. This distinction applies to uncomplete predicates, such as ”like”. For example an expression with *weak* negation is: ”I don’t like snakes”, whereas an expression with a *strong* negation is: ”I dislike snakes”. In case of complete predicates, such as parity of numbers, both negations collapse (”not odd(x)” is equivalent to ”neg odd(x)”). A discussion of consequences of using these two sorts of negation may be found in [10].

Among other goals and requirements for the RuleML, it should support the design and implementation of Semantic Web software agents. The full specification of an agent with use of RDF, ontology languages (RDF, OWL) and RuleML should be possible. This includes:

- an RDFS-based taxonomy for defining the schema of its mental state,
- a set of RDF facts for specifying its factual (extensional) knowledge,
- a set of RuleML integrity constraints for excluding non-admissible mental states,
- a set of RuleML derivation rules for specifying its terminological and heuristic (intensional) knowledge, and
- a set of RuleML reaction rules for specifying its behavior in response to communication and environment events.

RuleML rule-base may be included in a XML or RDF document. It requires using appropriate `ruleml: namespace`. XML/RDF-based syntax facilitates the interoperability of the RuleML and the Semantic Web languages and technologies.

3.3.2 Rule Interchange Format

The goal of the RIF initiative is to create a language that will express the rules like a rule language and enable for an exchange of rules among different engines. Rule-based systems can be very different, with respect to the rule semantics. They are based on various type of model theories, proof systems, etc. In order to enable interchange between various models a need for a core language and language dialects has been identified.

RIF Framework for Logic Dialects The general framework for logic dialects has been presented in [9]. "The RIF Framework for Logic Dialects (RIF-FLD) is a formalism for specifying all logic dialects of RIF, including the RIF Basic Logic Dialect. (...) All logic RIF dialects are required to be derived from RIF-FLD by specialization" [9]

RIF-FLD has the following main components:

Syntactic framework defining the mechanisms for specifying the formal presentation syntax of RIF logic dialects by specializing the presentation syntax of the framework. The presentation syntax is used in RIF to define the semantics of the dialects and to illustrate the main ideas with examples. This syntax is not intended to be a concrete syntax for the dialects, it leaves out the syntactic details. Since RIF is an interchange format, it uses XML as its concrete syntax.

Semantic framework describing the mechanisms that are used for specifying the models of RIF logic dialects.

XML serialization framework defining the general principles that logic dialects are to use in specifying their concrete XML-based syntaxes. For each dialect, its concrete XML syntax is a derivative of the dialect's presentation syntax. It can be seen as a serialization of that syntax.

The syntactic framework defines six types of RIF terms, formally defined in [9]:

- Constants and variables,
- Positional terms,
- Terms with named arguments,
- Frames (assertions about objects and their properties),

- Classification (subclass and class membership relationships),
- Equality, and
- Formula terms.

Semantic framework defines the notion of a semantic structure or interpretation. It is used to interpret formulas and to define logical entailment. As with the syntax, this framework includes a number of mechanisms that RIF logic dialects can specialize to suit their needs [9].

XML serialization framework defines the general principles for mapping the presentation syntax of RIF-FLD to the concrete XML interchange format.

Logic Cores and Dialects. BLD and PRD. Besides the general framework, a need for a standard core language and language dialects has been pointed out. Possible logic dialects include F-Logic, production rules, fuzzy or probabilistic logic. However, even this model is insufficient, because the difference between production rules and "traditional" logic systems is too large [27]. Therefore, a hierarchy of cores is necessary. This hierarchy comprises of a Basic Logic Dialect BLD [34] and Production Rule Dialect PRD [25] and they serve as "cores" for families of languages. A common RIF Core [52] aims at binding these two cores.

From a theoretical perspective, RIF-Core corresponds to the language of definite Horn rules without function symbols (Datalog) with a standard first-order semantics. RIF-Core is a subset of BLD, and a language of production rules where conclusions are interpreted as assert actions, thus also is a subset of RIF-PRD. It is then the intersection of RIF-BLD and RIF-PRD, though not the maximal one [52].

The basic difference between BLD and PRD can be summarized as follows: In the BLD (Basic Logic Dialect) the rules are treated as being of the form:

```
if condition is true then the conclusion is true
```

where conditions may include functions, hierarchies etc. On the other hand, the PRD (Production Rule Dialect) includes rules of the form:

```
if condition is true then do something
```

RIF BLD example in Presentation Syntax:

```
Document (
  Prefix(cpt http://example.com/concepts#)
  Prefix(ppl http://example.com/people#)
  Prefix(bks http://example.com/books#)
  Group
  (
    Forall ?Buyer ?Item ?Seller (
      cpt:buy(?Buyer ?Item ?Seller):-
      cpt:sell(?Seller ?Item ?Buyer)
    )
    cpt:sell(ppl:John bks:LeRif ppl:Mary)
  )
)
```

The Presentation Syntax is much more compact than the XML one. A mapping from the Presentation Syntax to the XML syntax is described in [34]. The general form of a rule in RIF BLD XML syntax is as follows:

- Subclassing and typing of BLD are equivalent to their RDFS counterparts.
- The datatypes are almost identical to OWL 2.

3.3.3 R2ML Language

R2ML²⁰ is a comprehensive XML-based rule format that allows enriching ontologies by rules, and interchanging rules between different systems and tools. With existing dedicated tools it is possible to visualize, verbalize, verify and validate the rules encoded in R2ML.

R2ML integrates the Object Constraint Language (OCL), a standard used in information systems engineering and software engineering, the Semantic Web Rule Language (SWRL), a proposal to extend the Semantic Web ontology language OWL by adding implication axioms, and the Rule Markup Language (RuleML), a proposal based on Datalog/Prolog. The language includes four rule categories: derivation rules, production rules, integrity rules and ECA/reaction rules. An example of a simple derivation rule is presented below:

```
<r2ml:RuleBase xmlns:r2ml="http://www.rewerse.net/I1/2006/R2ML"...>
  <r2ml:DerivationRuleSet ... >
    <r2ml:DerivationRule r2ml:ruleID="DER_WitchRule">
      <r2ml:conditions>
        <r2ml:ObjectClassificationAtom r2ml:class="Woman">
          <r2ml:ObjectVariable r2ml:name="girl"/>
        </r2ml:ObjectClassificationAtom>
        <r2ml:ObjectClassificationAtom r2ml:class="Burns">
          <r2ml:ObjectVariable r2ml:name="girl"/>
        </r2ml:ObjectClassificationAtom>
      </r2ml:conditions>
      <r2ml:conclusion>
        <r2ml:ObjectClassificationAtom r2ml:class="Witch">
          <r2ml:ObjectVariable r2ml:name="girl"/>
        </r2ml:ObjectClassificationAtom>
      </r2ml:conclusion>
    </r2ml:DerivationRule>
  </r2ml:DerivationRuleSet>
</r2ml:RuleBase>
```

R2ML supports two kinds of negation: weak negation – expressing negation-as-failure or non-truth, and strong negation – expressing explicit negative information or falsity. R2ML example rule set provides markup for rules written in various languages like: Prolog, F-Logic, SQL, OCL, RuleML, SWRL and others. It has been maintained by a REWERSE Working Group.²¹

3.4 Summary

In this Section an overview of selected rule languages and systems for the Semantic Web have been presented. Their prime objective is to operate on knowledge expressed with ontologies and rules. The approaches may be classified under various perspectives. The first factor is the architecture of the proposed system. Hybrid systems offering loose integration between the ontology and the rule components include: \mathcal{AL} -log, CARIN, ASP and hybrid MKNF knowledge bases. Two former proposals can be regarded as historical solutions and serve more as didactic systems than applied Semantic Web applications. Two latter ones introduce several interesting features, such as support for non-monotonic rules in ASP and belief revision in MKNF. Homogeneous approach has lead to

²⁰See <http://oxygen.informatik.tu-cottbus.de/reverse-i1/?q=node/6>.

²¹See <http://oxygen.informatik.tu-cottbus.de/reverse-i1/>.

the emergence of various ontology and rule languages. Most of them are either an extension or a fragment of OWL. They adhere to the Open-World (SWRL, DL-safe rules, ELP, DLP) or the Closed-World semantics (WRL). There are several advantages of the DLP approach. During the process of modeling it is possible to use either OWL or rules, and both DL reasoners and deductive rule systems may be used as a reasoning engine. Moreover, experience with OWL has shown that existing OWL ontologies rarely use constructs outside the DLP language [2]. In terms of modeling, SWRL is more similar to DL than LP, production or ECA rules. DL-safety restricts reasoning to ground facts (named individuals), but ensures decidability. WRL supports *negation-as-failure* (NAF), used in databases and Prolog programming. Some of the proposals concerned with augmenting OWL with rules are incorporated into the new version of the language, OWL 2. A visibly different language is TRIPLE which is based on F-Logic. The approaches support various sorts of rules. Most of the languages only permit deduction monotonic rules. Non-monotonicity is allowed in ASP approach. In pure OWL there are no constructs to modify the knowledge base, and consequently to support non-monotonic reasoning. Chosen features of the languages and systems presented in this Section have been summarized in Table 1.

To summarize, there is no one obvious way to introduce rules into the Semantic Web. There is lots of controversy around the issue, various formalisms and alternative solutions have been proposed. Challenges include layering the rules on existing stack technologies, and reconciling the ontology-based and rule-based modeling and reasoning paradigms. Simple union of OWL and rules leads to undecidability, which can be regained by introducing DL-safe rules. Several algorithms, including those based on reduction to Datalog programs can help regain tractability. For various sorts of rules there exist rule markup languages.

4 Rule Representation in Semantic Wikis

Semantic Wikis are one of the applications of the Semantic Web technologies. They are popular because of taking the collaborative and parallel nature of the web into consideration. A lot of implementations of semantic Wikis are developed²². Differences between them lie on the various levels, from the annotation mechanism through the subject granularity to knowledge representation. This section focuses on the wikis that support rules as the knowledge representation method.

4.1 AceWiki

AceWiki [39] is a new type of semantic wiki. It tries to use the controlled natural language Attempto Controlled English (ACE) for representing its content. ACE is based on English, but it has a restricted grammar and a formal semantics. Due to this fact, ACE avoids ambiguities of natural language.

The knowledge is retrieved by ACE parser²³ (APE) that translates the ACE text into *Discourse Representation Structures* (DRS). This representation is a syntactical variant of the first-order logic. Moreover, AceWiki supports translation of ACE language into OWL, which allows for reasoning with existing OWL reasoners. Mapping ACE to OWL covers all of OWL 1.1 except data properties and complex class descriptions.

AceWiki does not introduce any semantic annotation mechanism. On the user interface level AceWiki uses only one language that simultaneously constitutes a formal language. From the user point of view, learning the ACE language may be more complicated than semantic annotation. This is why AceWiki guides the user during creation of new sentences in a strict way. It provides a predictive editor (see Fig. 6) that does not allow for creation of incorrect sentences. What is more, the editor allows for checking a semantic correctness to some degree i.e. If the verb `meets` is defined in the ontology as a some relation between humans, then the editor will prevent user from writing sentences

²²http://semanticweb.org/wiki/Semantic_Wiki_State_Of_The_Art

²³<http://attempto.ifi.uzh.ch/ape>

Table 1: Rule Languages for Semantic Web comparison

	Logic	Rules	Integration	Vocab.Constructs	Semantics	Reasoning	CWA/OWA	Restrictions
<i>AC-log</i> [18]	<i>ACC</i> DL	Datalog	hybrid, unidir.	DL concepts		decidable		each variable in the head must appear in the body, only concepts as relational component
CARIN [40]	<i>ACC/R</i> DL	Datalog	hybrid, unidir.	concepts, roles		sound & complete		restrictions on DL terms in the head of rules
ASP [19]	DL	non-monotonic, constraints	hybrid		not compatible with FOL			
DLP [23]	DL	Horn Clauses	homogeneous	atoms concept assertions, subclass relations, equivalence, intersection, trans. properties	FOL	decidable	OWA	no disjunction in the head of rules
WRL [1]	DL (Core)	DLP (Core), Datalog subset of F-Logic(Flight), full Horn(Full)	homogeneous	concepts, relations, instances, axioms	UNA, Perf. Model(Flight), Well-Founded(Full)		CWA	
SWRL [33]	<i>SHOIN</i> DL	Datalog	homogeneous	concepts, roles	FOL	undecidable	OWA	only DL-atoms
DL-safe [47]	<i>SHOIQ</i> DL	DL-safe	homogeneous	concepts, roles, non-DL atoms				variables in a rule must occur in non-DL atom in the rule body
TRIPLE [14]	F-Logic	Datalog	hybrid	RDF triples	no fixed semantics			

like a man meets a home. Of course, the ontology must contain information that home is not a human.

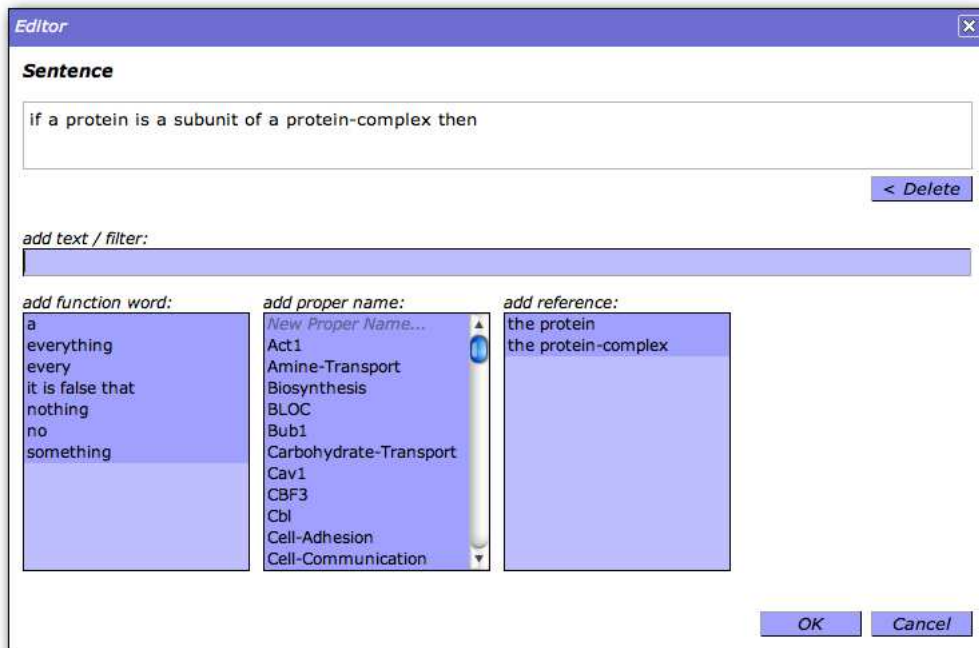


Figure 6: The screenshot of AceWiki predictive editor²⁴.

ACE Language provides its own representation of rules called AceRules[38]. AceRules is a rule system prototype that uses the ACE language as input and output. It is created and designed for forward-chaining interpreters that calculate the complete answer set. However, it can be easily adopted for backward-chaining inference.

AceRules reuses several existing ACE tools. First of all, the program is parsed by the ACE parser APE, which is written in SWI-Prolog and transformed into its DRS representation. This DRS representation is then translated by AceRules into an internal rule structure. Then different interpreter modules can be applied. From this step, a set of facts in an internal rule format is given. AceRules transforms this back into a DRS representation. Finally, this representation can be given to the existing ACE verbalizer module (that is used by APE for paraphrasing) returning the final answer in ACE language.

Let us consider an example. Firstly, some facts are inserted to the knowledge base:

```
Mary is a person.
John is a person.
Mary is a friend.
John is a shoplifter.
```

Next, using ACE language some rules are created:

```
Every friend is trustworthy.
Every shoplifter is dangerous.
```

The inference engine responds with the following output:

```
Mary is trustworthy.
Mary is a person.
John is dangerous.
Mary is a friend.
```

John is a shoplifter.
John is a person.

It is worth to show how the above rules are paraphrased by ACE verbalizer:

If there is a friend X1 then the friend X1 is trustworthy.
If there is a shoplifter X1 then the shoplifter X1 is dangerous.

and next transformed by ACE parser to:

```
trustworthy(A) <- [friend(A)].
dangerous(A) <- [shoplifter(A)].
person('John').
shoplifter('John').
friend('Mary').
person('Mary').
```

Using ACE language the queries can be executed as well. ACE supports two types of queries: *yes/no*-queries and *wh*-queries. *Yes/no*-queries establish the existence or non-existence of a specified situation. For example if user has specified:

A customer inserts a card.

then we can ask:

Does a customer insert a card?

The system responses with positive answer. Using *wh*-queries the text can be interrogated for some details of the specified situation:

A customer uses a card that is valid and that is owned by the customer.
The customer has an account that is activated.
The card belongs-to the account.
What is the code of the card?

Currently, ACE supports only a simple *wh*-queries, which contain only one *wh* word. This fact is relevant to OWL mapping. The individuals that can be proven to belong to the described type are the answers for such a query. Such simple queries can be directly answered by a reasoner like Pellet without any additional query language. The future works take the improvements of expression power of the queries into consideration.

4.2 KnowWE

The semantic wiki called KnowWE is developed as a diagnostic knowledge system. Its main goal is to allow integration of strong problem-solving knowledge into the wiki context.

The basic structure of content in KnowWE is typical for semantic wikis: every wiki article represents one concept of the ontology. The content of the article contains informal description of concept, usually inserted as a plain text. More formal information can also be included. The ontological knowledge can be defined by semantic annotations. Moreover, KnowWE allows to describe strong problem-solving knowledge, which can be introduced by definition of rules, decision trees and set-covering knowledge.

Two ontologies are formalized: task- and domain-specific. In the application ontology every concept is represented by a distinct wiki article with semantic annotations included. It contains knowledge about the domain and corresponds to approach in most semantic wiki systems.

The *task ontology* contains fundamental concepts to connect problem-solving knowledge elements and the described concepts. For example principle concepts of the problem-solving task, such as *user input* (fact inserted by users), *solution*, are defined.

Not only is the wiki used as a tool for knowledge engineering, but it also provides interfaces for interactive problem-solving. The distributed reasoning process enables the derivation of solutions over the entire wiki.

KnowWE allows for defining rules anywhere in the wiki article, inserted between `%%Rules` and `%`. The specialized compact markup for the definition of rules is developed. The goal was to create easy to use and intuitive language, which can be used by people with no or little training.

An example rule which states that if something is made of wood, it floats follows:

```
IF material = is made of wood
THEN floats = true
```

In KnowWE two basic types of rules can be used: *abstraction rules* and *scoring rules*.

Abstraction rules derives new facts, instances of findings, which are used in the rest of problem-solving process. If condition is fulfilled then in rule action some value is assigned to an input.

```
// Abstraction rule r1:
// If something floats then it is made of wood
IF floats = true
THEN material = is made of wood
```

Scoring rules are used for assigning confirmation weight for a particular solution. *Scores* tell how strong we are confirmed or disconfirmed of the solution. Seven positive and negative weights are possible: (P1, , P7) and (N1, , N7). Where P7 stands for categorical derivation of a solution, and N7 for categorical exclusion. The weights can be aggregated: two equal weights in sum gives next higher weight , e.g. $N3 + N3 = N4$, two equal weights with opposite sign results in nullify, e.g., $N3 + P3 = 0$.

```
// Scoring rule r2
// When woman burns then increase possibility that she is a witch
IF burns = true AND
   person = woman
THEN person is a witch = P7
```

4.3 SMW+

SMW+²⁵ is a semantic enterprise wiki that is distributed by Ontoprise GmbH²⁶ from Karlsruhe. SMW+ is a name for pre-packaged bundle which contain Semantic MediaWiki and Halo extension. Halo extension is developed on top of Semantic MediaWiki to enhance it with more user-friendly interfaces for adding semantic knowledge. Main features of the Halo extensions include enhanced wiki browsing (GUI-based ontology browser), improved knowledge authoring (auto-completion, Semantic Toolbar) and simplified knowledge retrieval (graphical query interface).

Within the Halo Project also the Rule Knowledge extension²⁷ was created. It provides a graphical editor for creating logical rules. Installation of Rule Knowledge extension requires pre-installed Semantic MediaWiki with Halo extension and a basic triple store. Currently, SMW+ provides connectors to the OntoBroker and the Jena triple stores.

Rules can be inserted within the wiki page using the rule tag. Default rule language is F-logic. An example rule which infer the finished tasks is as follows:

²⁵See http://wiki.ontoprise.de/smwforum/index.php/Main_Page.

²⁶See <http://www.ontoprise.de/index.php?id=135>.

²⁷See http://smwforum.ontoprise.com/smwforum/index.php/Help:Rule_Knowledge_Extension.

```
<rule name="myrule">
  FORALL x,y x[Has_finished_task->y] <- x[Has_task->y] AND
                                          y[Has_state->z] AND
                                          equal(z, "finished").
</rule>
```

F-logic syntax in SMW+ allows to express:

- less than/greater than – notation `less(10000, y)` means that `y` has value smaller than 10000
- less or equal than/grater or equal – notation `lessorequal(10000, y)`
- aggregation:

- the number of individuals connected by a certain property – `xcount`, e.g. the number of children of a person:

```
FORALL a,child,num
  a[prop#Has_number_of_children->num] <-
    a[prop#Has_child->child] AND
    xcount(a, f(a, child), child, num) .
```

- maximum value – `xmaximum` and maximum date – `xmaxdate`

F-logic rules are later translated to language of the rule store by the Triple Store Connector. Rules can be also inserted in native language of the rule store, e.g. Jena Triple Store or Ontobroker.

To simplify process of inserting rules the graphical rule editor is developed. After installing the Halo extension in edit mode the Semantic Toolbar is displayed. On property pages it includes the rules section.

The rule editor allows to create certain types of rules. These types are:

- a property chaining rule – infers relation between the first and the last part in the chain of properties, e.g. if person `X1` has parent, and the parent has brother `X3` then `X3` is an uncle of `X1` (Fig. 7).

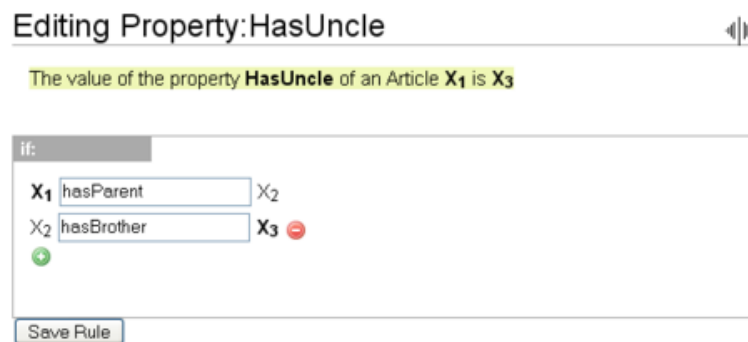


Figure 7: An example of a property chaining rule.

- a calculation rule – calculates value of property depending on other property values, e.g. the value of gravitation force basing on mass of an object and the gravitational acceleration (Fig. 8).
- a definition rule – makes an instance member of some category if it fulfills defined property constrains, e.g. all boys who are 18 years old and flirt with a girl belongs to category *Flirting18YearOldBoys* (Fig. 9).

Editing Property: Gravitational force

Gravitational force = ✓ (syntax checked) [edit formula](#)

Please specify the values of the following variables in your formula:

m is a property value

absolute term

g is a property value

absolute term

Fig. 7

Figure 8: An example of a calculation rule.

Editing Category: Flirting18YearOldBoys

Derive Category **Flirting18YearOldBoys** by complex rule

Head

All articles X_1 belonging to Category **Flirting18YearOldBoys** are defined by

Body

All articles X_1 belong to category **Boy**

AND

All articles X_1 have the property **Age** with value **18**

AND

All articles X_1 have the property **Flirts with** with value X_2

AND

All articles X_2 belong to category

Figure 9: An example of a definition rule.

All rules after saving are translated to F-logic rule and inserted within content of the wiki page. The rule is also displayed in the Rules part of the Semantic Toolbar and it is possible to edit the rule in the graphical editor.

4.4 TaOPis

The main purpose of the TaOPis (the **T**ransparent, **O**pen, **P**ublic, **A**utopoietic **I**nformation **S**ystem) is to provide a platform for self-organizing communities. Such communities can be either organizations or projects for which TaOPis provides suitable tools like semantic wiki systems with Niklas-like syntax, forums, blogs, ranking mechanisms, content filtering, tagging etc.

System constitutes a fully object oriented framework. Each concept becomes automatically a generic object, which can be specialized through attribute–value tags. TaOPis support building a class hierarchy by providing a special set of tags: `class`, `subclass`, etc. The system support semantic mechanism on the system organization level, not on the content level. This allows for adding metadata of the content. Any wiki page, forum post, blog entry, or user is treated as an object. Understanding the metadata by the system allows for building self-organizing structures.

TaOPis allows for querying a wiki knowledge base with the help of FLORA-2 [24] syntax. FLORA-2 is a rule-based knowledge representation and inference system. It is deeply rooted in F-logic [35], HiLog [16], and Transaction Logic [11]. F-logic provides object-oriented features: complex object, inheritance and class hierarchies. The contribution of HiLog is the enhancement of F-logic with meta-information processing capabilities. Finally, Translation Logic provides declarative

programming of "procedural knowledge".

The `[query]...[/query]` tag allows for entering a query in FLORA-2 syntax as well as defining query output format:

```
[query ?_:member[name -> ?na, surname -> ?s ]. ]
Members name is [b]?na[/b], and surname [b]?s[/b].
[/query]
```

The foregoing query returns the names and surnames of project/organization members and displays this information using bold font. F-logic allows also to create rules, for instance:

```
man(X) <- person(X) AND NOT woman(X) .
FORALL X, Y <- X:person[hasFather->Y <- Y:man[hasSon -> X] .
```

However, the TaOPis system is still in active development phase and the documentation is incomplete. Existing sources do not give an unambiguous answer if the F-logic rules can be directly used within the system.

4.5 Internet Business Logic

Internet Business Logic ²⁸ (IBL for short) is an advanced technology wiki for knowledge expressed in open vocabulary, executable English. The system can be used directly via a web browser or as an endpoint of the Service Oriented Architecture (SOA). Unlike other Natural Language systems, IBL does not need dictionary or grammar maintenance. It is not yet-another-controlled-English-system. The vocabulary used is open and can be easily extended. Users can write down executable English knowledge containing new terms. However, the system can be used to manage and query controlled vocabularies, taxonomies and ontologies.

System allows for writing specifications as English business rules, using open vocabulary. This specification can be run directly as though it was a program. An example of a rule may be given as follows:

```
for the retailer the term some-item1 has super-class some-class
for the manufacturer the term some-item2 has super-class that-class
-----
the retailer term that-item1 and the manufacturer term that-item2
agree - they are of type that-class
```

The expressions **some-...**, **a-...** are placeholders, or variables, that are filled in with actual values like "Jean Smith" or "5" when the rules are fired. Apart from the placeholders, the rest of the words in the rule are in open vocabulary. The rule defines the meaning of the last sentence in terms of the meanings of the first two. Due to this fact, a query that uses a given rule is the same as the last sentence of foregoing rule.

All the data are stored in the database. The rules are a sort of interface to the database. Using the foregoing rules the system generates (complex) SQL queries to retrieve the required information from the database. To avoid infinite regress, this process stops at the headings of data tables. The headings are similar sentences, and the number of placeholders in a heading is the number of columns in the table [62]. An example of the table can be as follows:

```
for this-region this-country we have this-number (km^3/yr) Annual Renewable Water Resource est: this-year this-source
-----
AFRICA                Algeria                14.3                1997                c f
AFRICA                Angola                184.0               1987                b
AFRICA                Benin                 25.8                2001                l
AFRICA                Botswana             14.7                2001                l
AFRICA                Burkina Faso         17.5                2001                l
AFRICA                Burundi              3.6                 1987                b
AFRICA                Cameroon             285.5               2003                m
AFRICA                Cape Verde           0.3                 1990                c
```

²⁸See http://semanticweb.org/wiki/Internet_Business_Logic.

AFRICA	Central African Republic	144.4	2003	m
AFRICA	Chad	43.0	1987	b
AFRICA	Comoros	1.2	2003	m
AFRICA	Congo	832.0	1987	b
AFRICA	Congo Democratic Republic (formerly Zaire)	1283	2001	l
AFRICA	Cote D'Ivoire	81	2001	l
AFRICA	Djibouti	0.3	1997	f
AFRICA	Egypt	86.8	1997	f
AFRICA	Equatorial Guinea	26	2001	l
AFRICA	Eritrea	6.3	2001	l
AFRICA	Ethiopia	110.0	1987	b
AFRICA	Gabon	164.0	1987	b
AFRICA	Gambia	8.0	1982	c
AFRICA	Ghana	53.2	2001	l
AFRICA	Guinea	226.0	1987	b
AFRICA	Guinea-Bissau	31.0	2003	m
AFRICA	Kenya	30.2	1990	c
AFRICA	Lesotho	5.2	1987	b
...				

The following examples show the rules that can be defined to query the given table. An explanation of the rules seems unnecessary, due to its intuitive form. For each rule, the query that uses the rule and its result are given.

1. Rule:

```
some-country has Renewable Water some-number (km^3/yr) and Withdrawal
some-amount (km^3/yr) that-amount is greater than or equal that-number
```

```
-----
that-country withdraws at least as much water per year as it can renew
```

The query can have the following form:

```
that-country withdraws at least as much water per year as it can renew
```

and the result of the query:

```
=====
Bahrain
Israel
Jordan
Kuwait
Libya
Oman
Qatar
Romania
Saudi Arabia
United Arab Emirates
Yemen
```

It is also possible to check if the given sentence is true or false. Using the query:

```
Bahrain withdraws at least as much water per year as it can renew
```

the system may response using one of the following forms:

```
(Yes, that is true)
(Sorry, No)
```

2. Rule:

```
some-country has Renewable Water some-number (km^3/yr) and Withdrawal
some-amount (km^3/yr) that-amount is greater than 10
```

```
-----
that-country-a major user-has Renewable Water that-number (km^3/yr)
and Withdrawal that-amount (km^3/yr)
```

The query can have the following form:

```
that-country-a major user-has Renewable Water that-number (km^3/yr)
and Withdrawal that-amount (km^3/yr)
```

and the result of the query:

```
=====
Afghanistan          65.0    23.26
Argentina            814.0   29.19
Australia            398.0   24.06
Azerbaijan           30.3    17.25
Bangladesh           1210.6  79.4
Brazil               8233.0  59.3
Canada               3300.0  44.72
Chile                 922.0   12.55
China                2829.6  549.76
Colombia              2132.0  10.71
Ecuador              432.0   16.98
Egypt                 86.8    68.3
France               189.0   33.16
Germany              188.0   38.01
Hungary              120.0   21.03
India                1907.8  645.84
Indonesia            2838.0  82.78
Iran                 137.5   72.88
...

```

3. Rule:

```
max a-year: some of the data in this this study is from the year
a-year=some-latest-year
-----
```

```
the latest year for data used in this study is that-latest-year
```

The query can have the following form:

```
the latest year for data used in this study is that-latest-year
```

and the result of the query:

```
=====
2005

```

From business rules, the system can generate and run SQL queries that would be too complicated to write reliably by hand, and it can explain the results, in English, at the business or scientific level.

4.6 PIWiki

PIWiki [36] is an extension of the DokuWiki with semantic facilities. The main goal of PIWiki is to deliver a generic and flexible solution. Instead of modifying an existing wiki engine or creating a new one a development of a plugin for DokuWiki has been chosen. To provide a rich knowledge representation and reasoning engine for the Semantic Web the SWI-Prolog environment was selected. The basic idea is to build a layered knowledge wiki architecture (see Fig. 10) where the expressive Prolog representation is used on the lowest knowledge level.

PIWiki stores the knowledge in the Prolog files. A content of each page is parsed and the knowledge is extracted into a separate file. Each file contains the pure Prolog clauses that constitute the lowest level of the knowledge representation. It is assumed that a single page may correspond to the several concepts and properties.

PIWiki uses well known semantic annotation mechanism. The syntax is compatible with Semantic MediaWiki (SMW) [37]: The wiki user can use SMW syntax directly in the edit pane of the PIWiki. However, PIWiki extends the standard SMW markup. It provides `<p1>` `</p1>` tags that contain pure Prolog clauses. The plugin parses the entered text and maps it to the Prolog clauses that are asserted to the knowledge base.

Rules are the natural knowledge representation in Prolog. The combination of wiki and Prolog provides a very powerful mechanism for rules. The syntax of rules and queries is the same as in Prolog. Using the extended markup of PIWiki the rules can be defined:

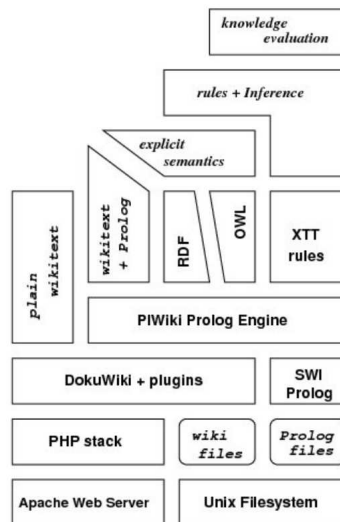


Figure 10: PIWiki architecture [48].

```
<pl cache="true">
  nordic_country(X) :-
    wiki_category(X, 'country' ),
    wiki_attribute(X, 'location', 'Northern Europe' ).
</pl>
```

The above code creates a rule that defines nordic countries. It can be easily used in a simple query:

```
<pl
  goal="nordic_country(X), write(X), nl, fail">
</pl>
```

The result of the foregoing query is the list of all nordic countries. It is also possible to specify a scope of the query:

```
<pl
  goal="nordic_country(X), write(X), nl, fail"
  scope="prolog:countries">
</pl>
```

The result of the query is depicted on the page in the same place as the query. The content of the result depends on the defined scope (in terms of namespaces) of the knowledge.

Prolog syntax allows for cross-references in queries. As opposed to SMW, PIWiki makes the ask about nordic country which capital is simultaneously its biggest city possible:

```
<pl
  goal="nordic_country(X), capital(X,Y), biggestcity(X,Y),
    write(X), nl"
  scope="prolog:countries">
</pl>
```

The cross-references in queries allows for creating very complex queries, which are not possible in the vast majority of described semantic wiki systems.

4.7 Summary

Semantic wikis constitute a popular application of the Semantic Web technologies. Although there exist quite a few implementations, not many support rule representation and processing. In this Section chosen semantic wiki systems were presented. The system provide support for various sorts of rules and queries. They usually use a dedicated rule representation language and markup. Some of the systems use variants of natural language (AceWiki, IBL), other use well established logical formalisms, such as F-logic (TaOPis, SMW+) or horn clauses in Prolog (PIWiki). In some systems the rules are preprocessed or translated into other formats before actual processing (SQL formulas in IBL, Prolog clauses in PIWiki). Rules can be processed either by the wiki system (KnowWE , PIWiki, IBL) or with help of external reasoners (Pellet in AceWiki, FLORA-2 in TaOPis) or Triple Stores (TripleStore Connector in SMW+), . Chosen features of the described systems are summarized in Table 2.

5 Combining Classical Rule-based Systems and Ontologies

In comparison with the classical expert system shells like CLIPS, Semantic Web technologies can be considered very young. Hence, although intensive work on these are in progress, they have still some limitations. To cope with this problem a lot of research is carried out. Some of it focuses on the integration of well-known classical expert system shells and the semantic technologies. This section describes chosen approaches to the integration.

5.1 R-DEVICE

The semantic description of the information is just a first step for reaching the goal of the Semantic Web. The next step is to provide the mechanism that will allow for reasoning over these information. Rules are mature (available theory and technology), natural knowledge representation that allows for automated reasoning. Hence, it seems natural to simply add the rules "on top" of the web technologies. However the authors in [3] claims that this approach can pose computationally and linguistically problems. Another approach consider to use RDF/RDFS and extend it by adding rules. This approach have two solutions:

- Build the inference engine that uses the RDF/RDFS directly.
- Use an existing rule system for reasoning over the RDF.

Building a new inference engine is possible but it means ignoring years of research. In this section, the other approach is presented with use of a deductive object-oriented knowledge base system, called R-DEVICE. R-DEVICE transforms RDF triples into objects and uses a deductive rule language for querying and reasoning about them. R-DEVICE provides OO-RDF data model[6] that maps RDF documents into COOL²⁹ (CLIPS Object Oriented Language) objects inside the CLIPS production rule system [5].

The architecture of the system is shown in Fig. 11 The R-DEVICE system consists of two major components:

- The RDF loader/translator that accepts from the user a RDF document and uses ARP parser to translate it to triples in the N-triple format.
- The rule loader/translator that translates a RuleML program into the native R-DEVICE rule notation and then into a set of CLIPS production rules.

The RDF to COOL translation is a complex process. The more detailed description can be found in [5, 7] Here only the simple facts concerning this translation are presented:

²⁹<http://www.gsi.dit.upm.es/docs/clipsdocs/clipshtml/vol1-Section-9.html>

Table 2: Rule support in chosen semantic wiki systems

	System characteristics	Rule representation	Supported rules	Queries	Reasoning	Interface
AceWiki	semantic wiki based on controlled natural language	AceRules – controlled language ACE	forward-chaining, (system can be adapted for backward-chaining)	yes/no queries, wh-queries	Pellet	wiki with a predictive editor
KnowWE	diagnostic knowledge system, integration of problem-solving knowledge, JSPWiki plugin	special markup, easy and intuitive, d3web XML	abstraction (derivation) rules, scoring rules	Yes (SPARQL queries prototype)	d3web engine	wiki with contextual editor, templates, auto completion
SMW+	Rule Knowledge Extension for Semantic MediaWiki	F-Logic	pre-defined rules based on inverse, symmetric and transitive properties, user-defined rules, such as calculation-, property chaining- and definition-rules	Yes (ASK, SPARQL)	TripleStore Connector	wiki with graphical editor for rules, query interface
TaOPis	object-oriented framework for self-organising communities	F-Logic, HiLog, Transaction Logic		Yes	FLORA-2	PHP, Facebook
IBL	wiki and SOA endpoint for business rules	open vocabulary, executable English	business rules	yes/no queries, SQL-like queries	internal rule processing, SQL queries	web forms, SOA endpoint
PIWiki	semantic plugin for DokuWiki	Prolog	cross-references	Yes	Prolog inference	wiki with a dedicated toolbar

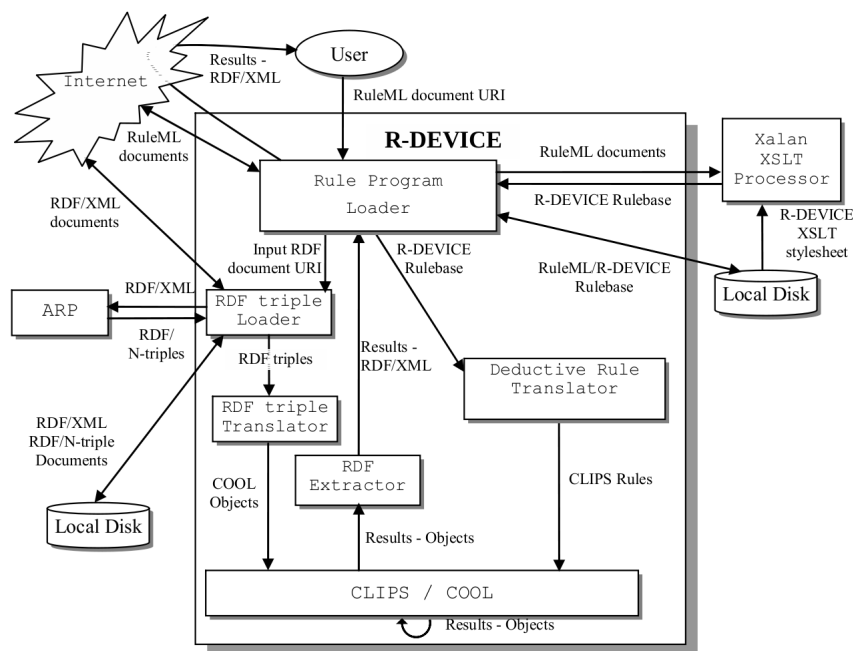


Figure 11: Architecture of the R-DEVICE system [5].

- Resource *classes* are represented both as COOL classes and as direct or indirect instances of the `rdfs:Class` class. This binary representation is due to the fact that COOL does not support meta-classes.
- All *resources* are represented as COOL objects, direct or indirect instances of the `rdfs:Resource` class.
- Properties are direct or indirect instances of the class `rdf:Property`.
 - Properties that have a single class domain are mapped to the slots of this class. The values of properties are stored inside resource objects as slot values.
 - Properties that have a multiple class domain are defined as a slot of class, which is defined as subclass of all classes of the domain.
- The `rdfs:range` constraint of properties defines the type of the values of slots.
 - When this constraint is absent, there is no type constraint for the slots.
 - If the value of the constraint is set, then the corresponding slot has the following type:
 - * `rdfs:Literal` → STRING.
 - * `xsd:integer`, `xsd:long` → INTEGER.
 - * `xsd:float`, `xsd:decimal` → FLOAT.
 - * When the range of a property is a class, then the type of the slot is INSTANCE.

Below, the Figures 12, 13 and 14 show the example of RDF/XML document and the R-DEVICE classes and objects that correspond to this document.

The descriptive semantics of RDF data may call for dynamic redefinitions of the OO schema, which are handled by R-DEVICE.

As it has been mentioned, the R-DEVICE uses a deductive rule language for querying and reasoning over RDF represented as objects. The conclusions of deductive rules represent derived classes. Each deductive rule comprise of two CLIPS production rules: one for inserting a derived object when the condition of the deductive rule is satisfied. The second for deleting the derived object when the

```

<!DOCTYPE rdf:RDF [
  <!ENTITY dmoz "http://directory.mozilla.org/rdf/">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dmoz="&dmoz;">
  <dmoz:Topic rdf:about="&dmoz;Top">
    <dmoz:catid>1</dmoz:catid>
    <dc:title>Top</dc:title>
    <dmoz:narrow rdf:resource="&dmoz;Top/Arts"/>
  </dmoz:Topic>
  <dmoz:Topic rdf:about="&dmoz;Top/Arts">
    <dmoz:catid>2</dmoz:catid>
    <dc:title>Arts</dc:title>
    <dmoz:link rdf:resource="http://www3.bc.sympatico.ca/PHILLIPSHOTGLASS/GlassPage.html"/>
  </dmoz:Topic>
  <dmoz:ExternalPage
    rdf:about="http://www3.bc.sympatico.ca/PHILLIPSHOTGLASS/GlassPage.html">
    <dc:title>John Phillips Blown glass</dc:title>
    <dc:description>A small display of glass by John Phillips</dc:description>
  </dmoz:ExternalPage>
</rdf:RDF>

```

Figure 12: Sample XML/RDF document.

```

(defclass rdfs:Resource
  (is-a RDF-CLASS)
  (multislot dmoz:narrow)
  (multislot dmoz:catid)
  (multislot dmoz:link)
  (multislot dc:title)
  ...
  (multislot rdf:type (type INSTANCE-NAME))
  ...
)
(defclass dmoz:ExternalPage
  (is-a rdfs:Resource)
)
(defclass dmoz:Topic
  (is-a rdfs:Resource)
)

```

Figure 13: R-DEVICE classes that correspond to the RDF document in the Figure 12

condition is no more satisfied, due to base object deletions and/or slot modifications. R-DEVICE includes features such as normal and generalized path expressions, stratified negation, aggregate, grouping, and sorting, functions. The rule language supports a second-order syntax, where variables can range over classes and properties.

The system is considered to be used as inference mechanism on top of an RDF repository or as on-the-fly RDF inferencing service.

DR-DEVICE DR-DEVICE is a system for defeasible reasoning on the Web. DR-DEVICE bases on the R-DEVICE system and is capable of reasoning about RDF data over multiple Web sources using defeasible logic rules.

In comparison with R-DEVICE, DR-DEVICE supports three types of rules that closely reflect defeasible logic: *strict rules*, *defeasible rules*, and *defeaters*. However, the DR-DEVICE system performs further translation of the rules, during which all the rules are translated into R-DEVICE rules. Each *defeasible rule* in DR-DEVICE is translated into five R-DEVICE rules:

- A *deductive* rule that generates the derived defeasible object when the condition of the defeasible rule is satisfied.

```

([dmoz] of rdfs:Resource
  (uri "http://directory.mozilla.org/rdf/")
  (source system)
  (rdfs:isDefinedBy [dmoz])
  (rdf:type [rdfs:Resource])
  (rdfs:label dmoz)
)
([dmoz:catid] of rdf:Property
  (source rdf)
  (rdf:type [rdf:Property])
  (rdfs:domain)
  (rdfs:range)
  (rdfs:subPropertyOf)
)
([dmoz:Topic] of rdfs:Class
  (source rdf)
  (rdf:type [rdfs:Class])
  (rdfs:subClassOf)
)
([dmoz:Top] of dmoz:Topic
  (source rdf)
  (dmoz:narrow [dmoz:Top/Arts])
  (dmoz:catid "1")
  (dmoz:link)
  (dc:title "Top")
  (rdf:type [dmoz:Topic])
)
([http://www3.../GlassPage.html] of dmoz:ExternalPage
  (source rdf)
  (dmoz:narrow)
  (dmoz:catid)
  (dmoz:link)
  (dc:description "A small display of glass by John Phillips")
  (dc:title "John phillips Blown glass")
  (rdf:type [dmoz:ExternalPage])
)
([dmoz:narrow] of rdf:Property
  (source rdf)
  (rdf:type [rdf:Property])
  (rdfs:domain)
  (rdfs:range)
  (rdfs:subPropertyOf)
)
([dmoz:link] of rdf:Property
  (source rdf)
  (rdf:type [rdf:Property])
  (rdfs:domain)
  (rdfs:range)
  (rdfs:subPropertyOf)
)
([dmoz:ExternalPage] of rdfs:Class
  (source rdf)
  (rdf:type [rdfs:Class])
  (rdfs:subClassOf)
)
([dmoz:Top/Arts] of dmoz:Topic
  (source rdf)
  (dmoz:narrow)
  (dmoz:catid "2")
  (dmoz:link [http://www3.../GlassPage.html])
  (dc:title "Arts")
  (rdf:type [dmoz:Topic])
)

```

Figure 14: R-DEVICE objects that correspond to the RDF document in the Figure 12

- An aggregate attribute *support rule* that stores the rule ids of the rules that can potentially prove positively or negatively the object.
- A derived attribute *overruled rule* that stores the rule id of the rule that has overruled the positive or the negative proof of the defeasible object, if the rule condition has been at least defeasibly proven, and if the rule has not been defeated by a superior rule.
- A derived attribute *defeasibly rule* that defeasibly proves either positively or negatively an object, if the rule condition has been at least defeasibly proven, if the opposite conclusion has not been definitely proven and if the rule has not been overruled by another rule.
- A derived attribute *defeated rule* that stores the rule id of the rule that has defeated overriding rules when the former is superior to the latter, if the rule condition has been at least defeasibly proven. A "defeated" rule is generated only for rules that have a superiority relation, i.e. they are superior to others.

Strict rules are handled in the same way as defeasible rules, with an addition of a derived attribute rule that definitely proves either positively or negatively an object, if the condition of the strict rule has been definitely proven, and if the opposite conclusion has not been definitely proven.

Defeaters are much weaker rules that can only overrule a conclusion. Therefore, for a defeater only the *overruled rule* is created, along with a deductive rule to allow the creation of derived objects, even if their proof status cannot be supported by defeaters.

The system executes the rules according to their type. The order of execution is as follows: *deductive, support, definitely, defeated, overruled, defeasibly*.

5.2 O-DEVICE

O-DEVICE [44] is another approach to rule-based reasoning over the ontology by using classical rule-based systems. In comparison with R-DEVICE (see Sect. 5.1) O-DEVICE tries to transform OWL Lite [29] ontology into CLIPS Object Oriented Language COOL. On the one hand, in existing implementations of reasoners it is possible to manipulate ontologies using rule notation (SWRL), or perform queries over ontology (SPARQL). On the other hand, it is not possible (or it is inefficient) to define a complete rule program over ontology. O-DEVICE uses a rule system that is able to reason over ontologies and gives the opportunity to utilize the ontology information by building knowledge-based systems.

The main motivation behind the O-DEVICE is exploiting the basic features an OO environment. The native mechanism of COOL for subclass relationships supports class subsumption and transitivity, treating both single and multiple inheritance issues. This prevents users from costly procedure of handling hierarchical class relationships. Nevertheless the OO model is not able to capture all the semantics of the OWL language. More complex constructions such as class intersections etc. cannot be directly modeled in OO environment. For this reason, O-DEVICE provides entailment rules. The basic OWL relations can be directly modeled in COOL:

- Subclass relationships – COOL supports the single as well as multiple class inheritance.
- Property inheritance – in COOL properties are called slots or multislots. They are automatically inherited to subclasses.
- Object relationships – objects referencing slots are used to define OWL instance relations.

The designed OO schema is implemented in a way to reflect OWL axioms. Here are the four basic axioms:

- Each class is a direct or indirect subclass of the `owl:Thing` class.
- Every object belongs directly or indirectly to the `owl:Thing` class.
- For every role for which no domain class is defined, the system assumes `owl:Thing` class.
- For every role for which no range constraint is defined system assumes `owl:Thing`.

In the Figure 15

1:<Person type Class>	(defclass Person
2:<friendOf type ObjectProperty>	(is-a Thing)
3:<friendOf domain Person>	(multislot friendOf
4:<friendOf range Person>	(type INSTANCE-NAME))
5:<age type DatatypeProperty>	(multislot age (type INTEGER)))
6:<age domain Person>	
7:<age range int>	(make-instance [paul] of Person)
8:<paul type Person>	(make-instance [nick] of Person)
9:<nick type Person>	(send [paul] put-friendOf [nick])
10:<paul friendOf nick>	

Figure 15: Transformation example [44].

The reasoning process is performed with the help of two types of rules. One type of rules is called static and applies OWL semantics on class and property definitions. The second type of rules is the production rules that are dynamically generated from the templates and are "filled" with actual ontology values. Here is the example of rule template:

```
(defrule <rule-name>
  (object (is-a <p-domain>) (name ?obj1)
   (<p> $? ?obj2 &: (transitive ?obj1 ?obj2 <p>) $?))
```

```
=> (bind $?v1 (send ?obj1 get-<p>))
      (bind $?v2 (send ?obj2 get-<p>))
      (send ?obj1 put-<p> (union $?v1 $?v2)))
```

The O-DEVICE provides deductive rule language that supports queries over the OWL instances represented as objects. The conclusions of deductive rules represent derived classes, whose objects are generated by evaluating these rules over the current set of objects. Each deductive rule is implemented as CLIPS production rule that asserts a derived object when the condition of the deductive rule is satisfied. The query language tries to simplifying the syntax of queries. Provides a simpler syntax than CLIPS and eliminate any lisp-like syntax. Below an example of the deductive rule language is presented. The query retrieves all the instances of the class `UndergraduateStudent` that have the value `Course0` in the property `takesCourse`:

```
(deductiverule r1
  ?id<- (UndergraduateStudent (takesCourse $? [Course0] $?))
  => (result (uGradStud ?id)))
```

In the future the O-DEVICE is planned to be integrated with a visual editor, which allows for defining queries using a RuleML-like syntax.

5.3 Integrating ontologies and rules within Protegé ontology editor

5.3.1 Historical solutions

Protegé's extensible plug-in architecture makes it possible to develop various extensions. Protegé-II, released in 1990, was designed specifically to exploit the CLIPS expert system shell. Because CLIPS supports Object-Oriented paradigm, Protegé was originally designed to work with frames. Before switching to OWL, the ontology knowledge base in Protegé was written in Protegé format: `.pont` files for class definitions, and `.pins` files containing information about instances. In Table 3 the information about integrating Protegé with chosen rule engines are summarized. The implementations may be considered historical, because they deal with older knowledge representation formats. However, current solutions may benefit from them.

	CLIPSTab [53]	JessTab [21]	Algernon [28]
Rule Engine	CLIPS	Jess	Algernon Abstract Machine
Implementation Language	C	Java	Java
Inference strategy	forward chaining	forward chaining	forward chaining backward chaining
Running production rules over Protegé KB	yes	yes	yes
Truth maintenance	yes	yes	no

Table 3: Rule engine plug-ins for Protegé

The solutions differ in functionality, implementation language and supported inference strategies. Detailed information about each solution may be found in [53, 21, 28].

5.3.2 SWRL Bridge in Protegé

Protegé-OWL³⁰ ontology editor provides a development environment for SWRL rules called SWRLTab. Within this environment several software components exist, including:

³⁰See <http://protege.stanford.edu/>.

- SWRL Editor and Built-in libraries – for SWRL rule creation, editing and management in an OWL ontology
- SWRL Built-in bridge – for user-defined SWRL built-ins
- SWRL Bridge – to provide an infrastructure for incorporating rule engines into Protegé-OWL.

SWRL Bridge allows interaction between a rule engine and OWL ontology with SWRL rules. `SWRLRuleEngine` interface has methods providing mechanisms to:

- import OWL/SWRL knowledge into a rule engine,
- allow rule engine to perform inferences and assert new facts into the bridge,
- write the new facts back into the OWL/SWRL ontology.

The methods may be implemented for various rule engines. The user need not to be aware of the engine used by the SWRL Bridge. Target rule engine implementations should be implemented as plugins, according to instructions given on the project website.³¹

Target rule engine must implement the `TargetSWRLRuleEngine` interface.³² There must be methods for representing OWL axioms and SWRL rule in the particular rule engine. Internally, SWRL Bridge uses OWLAPI-like representation for OWL classes, properties, individuals and SWRL rules.³³ The target rule engine must be able to handle this representation and translate it into the engine native format.

By default SWRL Bridge executes all the rules available in the ontology. In order to use a subset of the rules one can partition them into separate ontologies using SWRL Factory.³⁴

SWRL Bridge provides methods to deal with SWRL built-ins and user-defined built-ins. For target rule engines it provides mechanism to access Java-implementations of built-ins, as well as dynamic loading mechanism to find implementations of the built-ins at run-time.

It is also possible to use Protegé SWRLTab GUI to interact with a specialized rule engine bridge. Detailed requirements and information can be found on the Protegé's website.

5.3.3 Combining OWL and SWRL with Jess

One of the engines used within Protegé to operate on SWRL rules is Jess. SWRL Jess Bridge³⁵ is a plugin to the SWRL Tab in Protegé-OWL. It executes SWRL rules using the Jess rule engine. Its code is publicly available from Protegé Subversion repository.³⁶ SWRL Jess Bridge is provided in the Protegé-OWL distribution and is part of Protegé-OWL 3.4. The Jess engine must be installed separately.

SWRL Jess Bridge allows interaction between OWL ontology with SWRL rules and Jess. The main class constructor takes an instance of the `OWLModel` class, representing the OWL knowledge base with its associated SWRL rules, and an instance of a Jess `Rete` object which represents an instantiation of the Jess rule engine.

The inference process is controlled by the user. It is up to them, when the OWL/SWRL knowledge should be imported into Jess, when the engine performs inference, and when the asserted facts are written into the OWL knowledge base. The inference process can be done iteratively, facts asserted by one rule can be used in other rule's preconditions and new inference may be done.

³¹See <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLRuleEngineBridgeFAQ>.

³²See <http://protege.stanford.edu/protege/3.4/docs/api/owl/edu/stanford/smi/protege/owl/swrl/bridge/TargetSWRLRuleEngine.html>.

³³See <http://protege.stanford.edu/plugins/owl/api/>.

³⁴See <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLFactoryFAQ>.

³⁵See <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessBridge>.

³⁶See <http://smi-protege.stanford.edu/svn/swrl-jess-bridge/trunk/>.

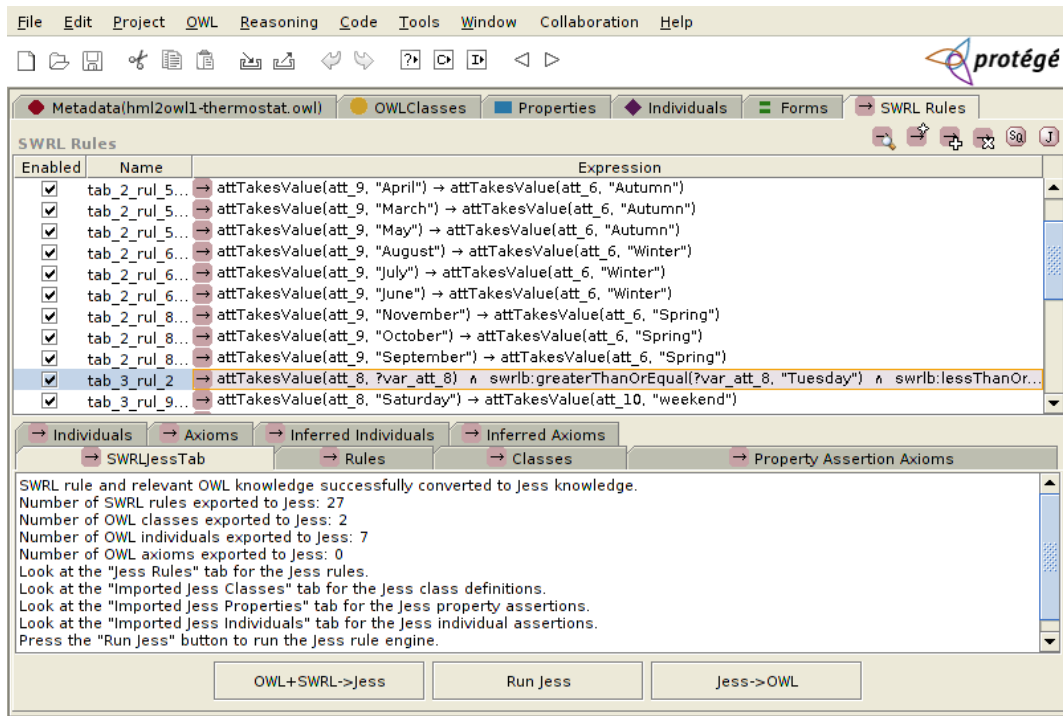


Figure 16: SWRLJesSTab in Protegé

When translating the knowledge from the OWL ontology and rules into the bridge, all the classes, properties and individuals referred to by the rules are copied into the bridge. SWRLJesSTab supports the following OWL axioms:

- OWL declaration axiom
- OWL class assertion axiom
- OWL property assertion axiom
- OWL subclass axiom
- OWL sub property axiom
- OWL equivalent class axiom
- OWL equivalent property axiom
- OWL different individuals axiom
- OWL same individuals axiom

Not all the OWL axioms are supported and, consequently, some inferences might not be done. Once the Jess is started, it executes the rules and concludes new facts. These facts can be then added to the ontology.

Currently the SWRL Jess Bridge is not integrated with an OWL reasoner. Therefore, new facts may be contradicted with existing OWL axioms. It is left to the user to discover the inconsistencies, for instance by running an OWL reasoner.

The user interface for the SWRL Jess Bridge is called SWRLJesSTab. It can be observed in Fig. 16. The SWRLJesSTab has several tabs allowing for controlling the inference process.

Mapping of the OWL axioms and SWRL rules into Jess is described in [50]. Basically, the Jess template facility, with its hierarchy and slots, is used for representing OWL classes, for example:

```
(deftemplate OWLThing (slot name))
(deftemplate Person extends OWLThing)
(deftemplate Man extends Person)
```

Using the definitions given above the OWL individual can be asserted as a member of the class as follows:

```
(assert (Man (name Fred)))
```

OWL properties are represented as Jess facts. This includes same-as and different-from relationships between individuals.

```
(assert (hasUncle Fred Joe))
(assert (sameAs Fred Frederick))
```

Similar approach is used for representing XML Schema data types.

```
(assert (xsd:unsignedInt ?x))
```

Built-ins are represented using the Jess `test` mechanism. For example, the SWRL built-in `greaterThan` applied to two integers can be written as:

```
(test (> 10 34))
```

Using the translations presented above, an SWRL rule:

```
Person(?x) ^ Man(?y) ^ hasSibling(?x,?y) ^
hasAge(?x,?age1) ^ hasAge(?y,?age2) ^
swrlb:greaterThan(?age2,?age1) ->
hasOlderBrother(?x,?y)
```

is represented in Jess as:

```
(defrule aRule (Person (name ?x))(Man (name ?y))
                (hasSibling ?x ?y)(hasAge ?x ?age1)
                (hasAge ?y ?age2)(test (> ?age2 ?age1))
=> (assert (hasOlderBrother ?x ?y))
```

5.3.4 DroolsTab for Protegé

Another example is integration of Drools with Protegé with a DroolsTab plugin. An example implementation realized with DroolsTab is Intelligent Geo-Information System³⁷ It can be downloaded from http://oogis.ru/component/option,com_remository/Itemid,34/func,fileinfo/id,2/lang,en/ The system is mainly a simulation plugin for spatial scenarios (for example ships collisions, air traffic). Its main objective is to study complex spatial processes by simulation and modeling. Rules are primarily used to process knowledge. It uses an open source Java library for developing OpenMap geo-information systems³⁸ and an open source rule-based system Jboss-Rules.³⁹ It can be used for visual authoring of complex spatial process simulation scenarios and general rule base authoring. It includes several demos.

³⁷See <http://protege.cim3.net/cgi-bin/wiki.pl?IntelligentGIS>.

³⁸See <http://openmap.bbn.com/>.

³⁹See <http://labs.jboss.com/portal/jbosrules/>.

6 Summary and trends

The aim of this report is to give an overview of research regarding rules and the Semantic Web. In the report the motivation for using rules on the Semantic Web was presented in Section 1. Various sorts of rules, as well as inference strategies were introduced in Section 2. Challenges of combining rules with ontologies were briefly discussed. The diversity of research proposals regarding rules languages and systems for the Semantic Web was presented in Section 3.

These theoretical solutions are gradually applied in real-life applications. One of them are semantic wiki systems which facilitate knowledge authoring, reusing and sharing. The wiki solutions were presented in Section 4, with the stress on representing and using rules.

Classical rule-based systems constitute a mature technology and several optimized algorithms and inference engine exist for years. Therefore, one section of the report is devoted to combining popular rule engines with ontology-based knowledge representation. Subjective selection of systems and tools has been presented in Section 5.

There are visible trends in the research on rules on the Semantic Web. Firstly, there are initiatives that aim at representing some sorts of rules *within* an ontology. Such an approach can be observed in Description Logic Programs, which are incorporated in the new version of Web Ontology Language, OWL 2 RL [46]. Such a reasoning can be provided by standard Description Logic reasoners.

Another approach is to build Datalog-like rules (see Section 2.3) using the OWL formulas as atoms in the rule head and body. An original union of OWL and Datalog rules is called Semantic Web Rule Language (SWRL). It is undecidable in a general case, so there has been research carried out to regain the decidability. Restricted versions of SWRL include DL-safe rules. The DL-safety is also taken into consideration in the OWL 2 language. DL-safe SWRL rules are supported by some of the DL reasoners, for instance Pellet.⁴⁰

In applications like semantic wikis, there is a need for lightweight reasoning. Rules definitions enable querying the knowledge in the wiki. The rules are usually formulated in a dedicated language, but at least a subset of them may be transformed into other representations. For instance, they may be mapped onto OWL concepts and thus be executed by a DL reasoner. In some systems the rules are based on F-logic and can be interpreted by tools such as FLORA-2. Some solutions are based on RDF rather than OWL and have built-in connectors to RDF Triple Stores, like Jena. The rules may also be interpreted and executed by a dedicated reasoner.

The last research track outlined in this report is using classical inference engines to work on ontologies. This problem requires some sort of mapping between ontology knowledge representation and languages used by inference engines (see Section 5). The inference uses optimized algorithms of the rule engines. The inference scenario may differ depending on the solutions. Such solutions may be used within systems, where knowledge is represented by means of ontologies.

Over the last decade there has been a steady increase in the number of systems using semantic technologies. Reasoning in these systems is either based on Description Logic reasoning or other logic. Some solutions have been adopted in certain classes of systems. It seems that it is the real-life use cases that will determine which of the approach is important and useful.

References

- [1] J. Angele, H. Boley, J. de Bruijn, D. Fensel, P. Hitzler, M. Kifer, R. Krummenacher, H. Lausen, A. Polleres, and R. Studer. Web rule language (WRL). W3C member submission, W3C, Sept. 2005. <http://www.w3.org/Submission/WRL/>.
- [2] G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. The MIT Press, 2008.
- [3] G. Antoniou and G. Wagner. Rules and defeasible reasoning on the Semantic Web. In *RuleML Workshop 2003*, pages 111–120. Springer-Verlag, 2003.

⁴⁰See <http://clarkparsia.com/pellet/>.

- [4] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
- [5] N. Bassiliades and I. Vlahavas. R-DEVICE: An object oriented knowledge base system for RDF metadata. *International Journal of Semantic Web and Information Systems*, pages 24–90, 2006.
- [6] N. Bassiliades and I. Vlahavas. Capturing RDF descriptive semantics in an object oriented knowledge base system. In *12th Int. WWW Conf. (WWW2003)*, Budapest, 2003.
- [7] N. Bassiliades and I. Vlahavas. R-DEVICE: An object-oriented knowledge base system for RDF metadata. Technical report, LPIS Group, Dept. of Informatics, Aristotle University of Thessaloniki, Greece, 2003.
- [8] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
- [9] H. Boley and M. Kifer. RIF framework for logic dialects. Candidate recommendation, W3C, Oct. 2009. <http://www.w3.org/TR/2009/CR-rif-fld-20091001/>.
- [10] H. Boley, S. Tabet, and G. Wagner. Design rationale of RuleML: A markup language for semantic web rules. In *SWWS'01*, Stanford, 2001.
- [11] A. J. Bonner and M. Kifer. *Logics for Databases and Information Systems*, chapter A logic for programming database transactions, pages 117–165. Kluwer Academic Publishers, March 1998.
- [12] R. Brachman and H. Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann, 1st edition, 2004.
- [13] I. Bratko. *Prolog Programming for Artificial Intelligence*. Addison Wesley, 3rd edition, 2000.
- [14] K. Breitman, M. Casanova, and W. Truszkowski. *Semantic Web: Concepts, Technologies and Applications*. Springer-Verlag, 2007.
- [15] 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*, 1(1):146–166, 1989.
- [16] W. Chen, M. Kifer, and D. S. Warren. HiLog: A foundation for higher-order logic programming. *Journal of Logic Programming*, 15(3):187–230, 1993.
- [17] J. de Bruijn. RIF rdf and OWL compatibility. W3C working draft, W3C, July 2008. <http://www.w3.org/TR/2008/WD-rif-rdf-owl-20080730/>.
- [18] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. \mathcal{AL} -log: integrating Datalog and Description Logics. *J. of Intelligent and Cooperative Information Systems*, 10:227–252, 1998.
- [19] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. *Artificial Intelligence*, 172(12–13), 2008.
- [20] T. Eiter, G. Ianni, A. Polleres, R. Schindlauer, and H. Tompits. Reasoning with rules and ontologies. In *Proceedings of Summer School Reasoning Web 2006, Lisbon, Portugal (4th–8th September 2006)*, volume 4126 of *LNCS*, pages 93–127. REWERSE, 2006.
- [21] H. Eriksson. Using JessTab to integrate Protegé and Jess. *IEEE Intelligent Systems*, 18(2):43–50, 2003.

- [22] A. V. Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of Applied Computer Methods*, 38(3):620–650, 1991.
- [23] B. N. Grosz, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: combining logic programs with Description Logic. In *Proceedings of the Twelfth International World Wide Web Conference, WWW2003*, pages 48–57, 2003.
- [24] C. Z. Guizhen Yang, Michael Kifer. FLORA-2: A rule-based knowledge representation and inference infrastructure for the Semantic Web. In *In Second International Conference on Ontologies, Databases and Applications of Semantics (ODBASE, volume 18, 2003*.
- [25] G. Hallmark, A. Paschke, and C. de Sainte Marie. RIF production rule dialect. Candidate recommendation, W3C, Oct. 2009. <http://www.w3.org/TR/2009/CR-rif-prd-20091001/>.
- [26] J. Hendler and F. van Harmelen. *Handbook of Knowledge Representation*, chapter The Semantic Web: Webizing Knowledge Representation. Elsevier, New York, 2008.
- [27] I. Herman. Some W3C SW technologies to watch. <http://www.w3.org/2009/Talks/0316-Amsterdam-IH/>, 16 March 2009. Ivan Herman’s Public Presentations.
- [28] M. Hewett. The Algernon abstract machine. Technical Report No. AI2000-284, University of Texas at Austin Artificial Intelligence laboratory, 2000.
- [29] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. OWL 2 Web Ontology Language – primer. W3C recommendation, W3C, October 2009.
- [30] A. Horn. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 16(1):14–21, 1951.
- [31] I. Horrocks. OWL Rules, OK? In *W3C Workshop on Rule Languages for Interoperability*, April 27-28 2005.
- [32] I. Horrocks, B. Parsia, P. Patel-Schneider, and J. Hendler. Semantic web architecture: Stack or two towers? In F. Fages and S. Soliman, editors, *Principles and Practice of Semantic Web Reasoning*, number 3703 in LNCS, pages 37–41. Springer, 2005.
- [33] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosz, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML, W3C member submission 21 may 2004. Technical report, W3C, 2004.
- [34] M. Kifer and H. Boley. RIF basic logic dialect. Candidate recommendation, W3C, Oct. 2009. <http://www.w3.org/TR/2009/CR-rif-bld-20091001/>.
- [35] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.
- [36] M. Kotra. Design of a prototype knowledge wiki system based on Prolog. Master’s thesis, AGH University of Science and Technology in Kraków, 2009.
- [37] M. Krötzsch, D. Vrandečić, M. Völkel, H. Haller, and R. Studer. Semantic wikipedia. *Web Semantics*, 5:251–261, 2007.
- [38] T. Kuhn. Acerules: Executing rules in controlled natural language. In *Proc. First International Conference on Web Reasoning and Rule Systems (RR2007, volume 4524, pages 299–308, Heidelberg, 2007*. Springer.

- [39] T. Kuhn. AceWiki: A Natural and Expressive Semantic Wiki. In *Proceedings of Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges*. CEUR Workshop Proceedings, 2008.
- [40] A. Y. Levy and M.-C. Rousset. Combining horn rules and description logics in CARIN. *Artif. Intell.*, 104(1-2):165–209, 1998.
- [41] A. Ligęza. *Logical Foundations for Rule-Based Systems*. Uczelniane Wydawnictwa Naukowo-Dydaktyczne AGH w Krakowie, Kraków, 2005.
- [42] F. A. Lisi and F. Esposito. Building rules on top of ontologies? Inductive Logic Programming can help! In G. Semeraro, E. Di Sciascio, C. Morbidoni, H. Stoermer, G. Semeraro, E. Di Sciascio, C. Morbidoni, and H. Stoermer, editors, *SWAP*, volume 314 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.
- [43] Object Management Group. Object Constraint Language version 2.0. Technical report, OMG, May 2006.
- [44] G. Meditskos and N. Bassiliades. O-DEVICE: An object-oriented knowledge base system for OWL ontologies. In G. Antoniou, G. Potamias, C. Spyropoulos, and D. Plexousakis, editors, *SETN*, volume 3955 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 2006.
- [45] E. Miller and F. Manola. RDF primer. W3C recommendation, W3C, Feb. 2004. <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>.
- [46] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language: Profiles. W3C recommendation, W3C, October 2009.
- [47] B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. In *Journal of Web Semantics*, pages 549–563. Springer, 2004.
- [48] G. J. Nalepa. PIWiki - a generic semantic wiki architecture. In N. T. Nguyen, R. Kowalczyk, and S.-M. Chen, editors, *ICCCI*, volume 5796 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2009.
- [49] U. Nilsson and J. Maluszynski. *Logic, Programming, and PROLOG*. John Wiley & Sons, Inc., New York, NY, USA, 1995.
- [50] M. O’connor, H. Knublauch, S. Tu, and M. Musen. Writing rules for the Semantic Web using SWRL and Jess. In *8th International Protegé Conference, Protegé with Rules Workshop*, 2005.
- [51] OMG. UML notation guide version 1.1. Technical report, Object Management Group, september 1997.
- [52] A. Paschke, D. Reynolds, G. Hallmark, H. Boley, M. Kifer, and A. Polleres. RIF core dialect. Candidate recommendation, W3C, Oct. 2009. <http://www.w3.org/TR/2009/CR-rif-core-20091001/>.
- [53] K. Pooloth and R. Jetley. Integrating the CLIPS rule engine with Protegé. In A. A. et al., editor, *Proc. of the first ICGST International Conference on Artificial Intelligence and Machine Learning AIML 05*, volume 05, pages 134–140, Cairo, Egypt, Dec. 2005. ICGST, ICGST.
- [54] T. C. Przymusiński. On the declarative and procedural semantics of logic programs. *J. Autom. Reason.*, 5(2):167–205, 1989.
- [55] M. Rogula. Metody inżynierii wiedzy – projekt: HML rules. AGH UST, 2008. Supervised by G. J. Nalepa, Ph. D.

- [56] R. Rosati. DL+log: Tight integration of description logics and disjunctive Datalog. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 68–78, 2006.
- [57] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2nd edition, 2003.
- [58] A. Sakharov. Horn clause. <http://mathworld.wolfram.com/HornClause.html>. MathWorld – A Wolfram Web Resource, created by Eric W. Weisstein.
- [59] T. Swift, D. Warren, K. Sagonas, J. Freire, P. Rao, B. Cui, E. Johnson, L. de Castro, R. Marques, D. Saha, S. Dawson, and M. Kifer. The XSB system version 3.2. volume 1: Programmer’s manual, march 2009.
- [60] J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press, 1988.
- [61] F. van Harmelen and D. L. McGuinness. OWL Web Ontology Language overview. W3C recommendation, W3C, Feb. 2004. <http://www.w3.org/TR/2004/REC-owl-features-20040210/>.
- [62] A. Walker. A Wiki for Business Rules in Open Vocabulary, Executable English.