RULE-BASED CHAINING WITH ONTOLOGICAL CONSTRAINT CHECKING: A HYBRID REASONING SYSTEM FOR SEMANTIC WEB

Hataichanok Unphon[1], Ekawit Nantajeewarawat[1*] and Photchanan Ratanajaipan[2]

Received: May 16, 2005; Revised: Nov 11, 2005; Accepted: Nov 21, 2005

Abstract

An integrated system for knowledge representation, consisting of a structural component and a relational component, is presented, and its application to Semantic Web is outlined. Grounded upon Description Logics (DL), the structural component enables representation of ontologies along with Resource Description Framework (RDF) annotations of Web contents, and various kinds of ontology-based reasoning. The relational component, by contrast, allows one to describe relationships between individuals using an extended form of Horn rules, in which terminological membership constraints can be specified. Reasoning process in the integrated system is driven by an extended backward-chaining algorithm embedded in the relational component. The two components interact by means of knowledge base unsatisfiability checking, which is supported by a state-of-the-art DL system, called RACER.

Keywords: Semantic web, ontology, description logics, horn rules, hybrid reasoning

Introduction

The key idea of Semantic Web (Berners-Lee, 2000; Hendler, 2001) is to have resources on the Web defined and linked in such a way that they are not only interpretable by humans but also by software agents. A major approach towards the realization of this idea is to annotate Web resources with metadata using Resource Description Framework (RDF) (Decker *et al.*, 2000) with embedded XML. Such annotations, however, will be of limited value unless they employ a common vocabulary that a group of agents agrees upon beforehand and that has well-defined semantics.

The concept of an *ontology* was introduced in the area of knowledge engineering and information integration as a formal, explicit specification of shared conceptualizations (i.e., meta-information) that describe the semantics of data

[1] *Information Technology Program, Sirindhorn International Institute of Technology, Thammasat University, P.O. Box 22, Thammasat-Rangsit Post Office, Pathumthani 12121, Thailand, E-mail: {unphon,ekawit}@siit.tu.ac.th*

[2] *Computer Science Program, Shinawatra University, Pathumthani Campus, 99 Moo 10, Bangtoey, Samkok, athumthani 12160, Thailand, E-mail: photchanan@shinawatra.ac.th*

[*] *Corresponding author*

(Uschold and Gruninger, 1996; Fensel, 2001; Staab and Studer, 2004). As such, an ontology provides a source of shared and precisely defined terms. In its computational form, an ontology typically comprises definitions of concepts organized in a partially ordered generalization taxonomy along with a set of relationships that hold among them. These constructs collectively impose a structure on the domain being represented and constrain possible interpretations of terms.

Recognition of the need for ontologies in Semantic Web brought about several proposals of Web ontology languages, among which OWL Web Ontology Language (Antoniou and van Harmelen, 2004) has emerged as a de facto standard notation. OWL exploits existing Web standard markup languages, i.e. XML and RDF (Decker et al., 2000), as its underlying syntax, and Description Logics (DL) (Baader *et al.,* 2002) as the formal underpinnings of its semantics. The connection to DL enables formalization of an OWL ontology together with Web resource annotations as a DL knowledge base $<\mathcal{T}, \mathcal{A}>$, where the terminology part (TBox) $\mathcal{T}$ corresponds to the ontology and the assertion part (ABox) $\mathcal{A}$ represents the annotations. Consequently, currently existing implemented DL systems, e.g. FaCT (Horrocks, 1998), DLP (Patel-Schneider and Horrocks, 1999) and RACER (Haarslev and Möller, 2001), readily lend their deduction services (Donini *et al.,* 1996; Baader *et al.,* 2002) as reasoning supports to OWL applications (e.g. for determination of semantic relationships between syntactically different terms in an application domain).

Notwithstanding several advantages, formalization of Semantic Web contents as a DL knowledge base imposes certain limitations on representational expressiveness. While DL is suitable for describing the schema of an application domain, its expressive power in representing rules involving relations between individuals is inherently restricted. The restriction is due in large part to the necessity of preserving the tree model property (Vardi, 1997), without which the decidability of DL may be lost (Calvanese *et al.,* 2001). This requirement severely constrains the way variables and quantifiers can be used, e.g. it is impossible to describe classes whose instances are related to anonymous individuals via different property paths. Accordingly, some rules that can be simply represented in the Horn fragment of the first-order logic cannot be asserted in DL; e.g. one cannot assert that individuals who live and work at the same location are "home workers", whereas this assertion can be given by using a simple Horn rule

$\forall$w,x,y,z: homeWorker(x) < work(x,y), live(x,z), locate(y,w), locate(z,w).

While not all Horn rules can be represented in DL, it has been shown in Grosof *et al.* (2003) that unrestricted use of basic DL class constructors (i.e., disjunction, universal restriction, existential restriction, and negation) can also result in DL statements that cannot be expressed by Horn rules. DL and Horn rules are thus different fragments of the first-order logic, and, as depicted in Figure 1, neither of them includes each other. This motivates an attempt to extend one fragment with features supported by the other fragment, with expectation of a more expressive representation formalism.

To improve the representational adequacy and deductive power of knowledge representation formalisms for Semantic Web, a hybrid system consisting of a DL-based structural subsystem and a rule-based relational subsystem is proposed. Its architectural overview is shown in Figure 2. Deduction process in this hybrid system is driven by an extended backward-chaining algorithm embedded in the relational subsystem. As its distinctive feature, in addition to computing answers to a given query using a traditional goal-directed method based on the resolution inference rule, the extended algorithm collectively constructs lists of terminological constraints (concept-assertion constraints), which are then used for verification of the computed answers with respect to the ontology defined in the structural part. An interaction between the two subsystems takes place through the knowledge base unsatisfiability checking. The RACER system (Haarslev and Möller, 2001) is used as the inference engine of the structural subsystem.

The paper is organized as follows. The second†section describes a structural subsystem and a relational subsystem in terms of the representation languages they provide, and presents the extended backward-chaining algorithm and a hybrid reasoning method. The third section presents some experimental results. The fourth section discusses related works, and finally the fifth section draws the conclusions.

Methods

Representation languages of the two subsystems will be presented first, and then an extended backward-chaining algorithm will be described. To simplify the presentation, a simple concept language in the DL family, called $\mathcal{ALCQ}$ (Attributive Language with Full Complement and Number Qualification), is used as a basis in the structural part.

Knowledge Representation in a Structural Subsystem

A structural subsystem represents an ontology together with RDF annotations of Web contents. As will be elaborated below, it can be formulated as a DL knowledge base, consisting of a terminology part (TBox) and an assertion part (ABox), where the former is a set of inclusion statements, and the latter is a set of membership statements. To define the syntax of statements of the two kinds, assume that $\mathcal{CN}$ is a set *of concept names (class names),* $\mathcal{RN}$ a set of *role names,* and O a set of *individual names,* and that $\mathcal{CN}$ and $\mathcal{RN}$ are disjoint. *Concept expressions* are constructed using the syntax rules in Figure 3, where C and D are concept expressions, A is a concept name in $\mathcal{CN}$, and R is a role name in $\mathcal{RN}$.

An *inclusion statement* is an expression of the form $C \sqsubseteq D$, where C and D are concept expressions; it is intended to mean "the extension of C (the set of all instances of C) is always a subset of the extension of D (the set of all instances of D)". A pair of inclusion statements $C \sqsubseteq D$ and $D \sqsubseteq C$ will be denoted by $C \equiv D$, which is intended to mean "the extension of C is always identical to that of D". A *membership statement* is an expression of the form a:C or of the form $\langle a,b \rangle$:R, where a and b are individual names in O, C is a concept expression, and R is a role name in $\mathcal{RN}$. A membership statement of the first form, which is also called a concept assertion, is intended to mean "an individual a is an instance of C", while that of the second form, which is also called a *role assertion,* is intended to mean "an individual a is related to an individual b by a role R".[3]

Having its formal semantics defined based on DL, an OWL ontology can be translated into a set of inclusion statements. Table†1 illustrates translation of some of the OWL class constructors into the DL concept constructors, and Table 2 illustrates that of OWL axioms into DL inclusion statements. Accordingly, the OWL assertion in Figure 4(a), for example, which specifies "not being taken by any underclassman" as a necessary condition for being an AdvCourse (advanced course), will be translated into the TBox statement

AdvCourse $\sqsubseteq$ Course $\sqcap \neg\exists$takenBy. Underclassman.

An RDF annotation will be translated into one or more ABox membership statements. For example, the RDF statement in Figure 4(b) will be translated into the concept assertion John: Prof, along with the role assertion ⟨John, PhD⟩: highest Degree.

Figure 5 provides an example of a structural part encoded in DL. Consider the TBox in the figure. The first and the second statements altogether specify that the concept Person subsumes the concept FM (faculty member), which in turn subsumes the concept Prof (professor). The third statement defines the concept DFM (doctoral faculty member) as the set of faculty members whose highest degrees are doctoral degrees. The fourth statement provides a necessary condition for AdvCourse. The fifth statement specifies that Underclassman

[3] *The formal meanings of concept expressions, inclusion statements and membership statements are normally defined based on Tarski style model theoretic semantics. The reader is referred to Baader et al. (2002) for details.*

is a subclass of Undergrad. The sixth and the seventh statements altogether assert that Undergrad and Grad (graduate) are disjoint subclasses of Person. It is noteworthy that the use of negation in the definition of AdvCourse excludes the possibility of defining this concept in Horn logic; furthermore, the constraint that the extensions of two concepts are not overlapping, as specified by the seventh statement, cannot be represented using Horn rules. The ABox in Figure 5 partially populates concepts and partially relates individuals by roles; e.g. it asserts that PhD and DEng are doctoral degrees, and the highest degree of John is PhD.

Knowledge Representation in a Relational Subsystem

A relational subsystem describes the relationships between individuals using a set of extended Horn rules and ground facts. Rules and facts used in this subsystem are defined as follows. Let $\mathcal{PN}$ be a set of *predicate symbols,* and assume that $\mathcal{PN}$, $\mathcal{CN}$, and $\mathcal{RN}$ are pairwise disjoint. An *atom* is an expression of the form p(t_1, ..., t_k), where p is a predicate symbol in $\mathcal{PN}$, $k \geq 0$, and each of the t_i is either a constant or a variable. A *constraint* is an expression of the form *a:C,* where a is either an individual name in *O* or a variable, and *C* is a concept expression constructed out of concept names in $\mathcal{CN}$ and role names in $\mathcal{RN}$.

An *extended Horn rule* (or *rule,* for short) *r* takes the form

$h <\ \ b_1, b_2, ..., b_m, con_1, con_2, ..., con_n,$

where $m, n \geq 0$, h and the b_i are atoms, and the con_j are constraints. The atom *h* is called the *head* of *r*, denoted by *Head*(*r*); the list $[b_1, b_2, ..., b_m]$ is called the *body* of *r*, denoted by Body(*r*); and the list $[con_1, con_2, ..., con_n]$ is called the *constraint part* of *r*, denoted by *Constr*(*r*). When *Body*(*r*) and *Constr*(*r*) are both empty, *r* will be called a *unit clause* and the symbol <=will often be omitted. A *fact* is a ground (variable-free) unit clause.[4] It is assumed in this paper that each rule is *safe,*[5] i.e. for each rule *r*, each variable occurring in *Head*(*r*) must occur in *Body*(*r*). This assumption ensures that from a relational part consisting of a finite set of rules and facts, the set of all derivable conclusions will always be finite.

An example of a relational part is given in Figure 6 (variables are written in *italics*). It represents a goal and part of the knowledge base of an agent. Rule R1 specifies the goal of finding persons who are experts in a topic of interest of the agent. Rules R2, R3 and R4 altogether state that an individual *A* is regarded as an expert in a topic *T* if at least one of the following conditions holds: (1) A is a doctoral faculty member who is an author of a publication with a keyword *T*; (2) A is a doctoral faculty member who has *T* as one of his/her research interests and supervises some graduate students; (3) *A* is a lecturer of an advanced course that covers the topic *T*. Facts F1 and F2 state, respectively, that the agent is interested in DL and IR. It is assumed that Facts F3-F7 represent the information related to publications and courses that the agent has gathered so far. Facts are typically collected from RDF statements; for example, F3 and F4 may be obtained from the RDF statement in Figure 7.

An Extended Backward-Chaining Algorithm

Backward chaining (Russell and Norvig, 2003) is a well-known form of goal-directed reasoning with conventional Horn rules. To deal with constraints in extended Horn rules, an extended backward-chaining algorithm is devised. Basically, in addition to the construction of a proof tree for finding the set of all substitutions that make a given query satisfied, the extended algorithm collects all constraints that are imposed upon by the rules involved in the construction. Collection of such constraints demands modification of the parameter-passing and answer-returning structure of a conventional backward-chaining algorithm.

[4] *Notice that a fact is a special form of a rule.*

[5] *The notion of a safe rule is adopted from Datalog (Ullman, 1989).*

The extended backward-chaining algorithm, called EXTENDED-BC, is presented in Figure 8 (the symbol '||' denotes list concatenation). The algorithm takes a set KB_R of extended Horn rules, a list qlist of atoms (queries), a list *clist* of constraints, and a substitution θ as inputs, and produces as its output a set of substitution-constraints pairs, each of which takes the form $\langle \sigma, cl\rangle$, where σ is a substitution and cl is a list of constraints. The lists *qlist* and *clist* can be thought of, respectively, as a stack of queries (waiting to be proved) and a queue of accumulated constraints. The algorithm works by taking the first query q in *qlist* and then finding every rule in the knowledge base KB_R whose head is unifiable with q. Each of such a rule creates a new recursive call in which the (instantiated) body of the rule is added to the query stack and the (instantiated) constraint part of the rule is added to the constraint queue. When a query unifies a fact (i.e. a rule with empty body), no new query is added to the stack and the query is solved. The algorithm uses four other algorithms, i.e. FIRST, REST, UNIFY, and COMPOSE, where for any non-empty list l, FIRST(l), and REST(l) return the first element and the tail of l, respectively; for any atoms a_1 and a_2, UNIFY(a_1, a_2) returns the most general unifier of a_1 and a_2 (if they are unifiable); and for any substitutions θ_1 and θ_2, COMPOSE(θ_1, θ_2) returns the composition of θ_1 and θ_2.

Assume that a structural knowledge base KB_S, consisting of a TBox and an ABox, a relational knowledge base KB_R, consisting of extended Horn rules, and a query q are given. By invoking EXTENDED-BC with the singleton list $[q]$, the empty list, and the identity substitution (the empty set of bindings) as the initial query stack, the initial constraint queue, and the initial substitution, respectively, one obtains a set answers of substitution-constraints pairs such that for each $\langle \theta, cl\rangle \in$ answers, $q\theta$ can be proved using KB_R solely (provided that the constraint parts of the rules in KB_R have not yet been taken into account). The constraints in *answers* will then be verified with respect to KB_S, and some answer substitutions may be discarded. The verification consists of two steps. First, collect all substitution-constraints pairs $\langle \sigma_1, cl_1\rangle, \langle \sigma_2, cl_2\rangle, ..., \langle \sigma_m, cl_m\rangle$ in answers such that $q\sigma_1 = q\sigma_2 = ... = q\sigma_m$ and each $q\sigma_i$ is a ground atom ($1 < i < m$). Then, check whether KB_S entails at least one of cl_1, cl_2, ..., cl_m. Based on the theoretical foundation given in Baader *et al.* (1990), this entailment checking can be reduced into a knowledge base unsatisfiability problem; that is, the entailment is satisfied if and only if for any combination of assertions a_1: C_1, a_2: C_2, ..., am: Cm such that for each i ($1<i<m$), a_i: C_i is a constraint in cl_i, KB_S is unsatisfiable when its ABox is augmented with the set $\{a_1{:}\neg C_1, a_2{:}\neg C_2, ..., a_m{:}\neg C_m\}$. Supposing that each cl_i contains at most k constraints, there will be at most k^m combinations of assertions, and, consequently, one can check the entailment by performing at most km unsatisfiability checks. The soundness and completeness of this query answering method follows from its correspondence with a proof by constrained resolution-based refutation (Donini *et al.,* 1998).

An Example

Let KB_S be the structural part in Figure 5 and KB_R the relational part in Figure 6. Consider the query find(X,Y), which, according to the rule R1 in KB_R, is intended to mean "find every pair of X and Y such that X is an expert in a topic Y and Y is a topic of interest of the agent in question". By calling EXTENDED-BC(KB_R, [find(X,Y)], [],$\varnothing$), the following substitution-constraints pairs are obtained:

P1. $\langle\{X$/Mary, Y/IR$\}$, [Mary: Person, Mary: DFM, jacm48p885: Publication]$\rangle$.

P2. $\langle\{X$/John, Y/DL$\}$, [John: Person, John: DFM $\sqcap\exists$supervise.Grad]$\rangle$.

P3. $\langle\{X$/John, Y/DL$\}$, [John: Person, ITS413: AdvCourse]$\rangle$.

The collected constraints are verified with respect to KB_S as follows. First, the three substitution-constraints pairs are partitioned into two collections: {P1} and {P2, P3}. From the first collection, unsatisfiability of three augmented knowledge bases, i.e.

$KB_S \cup$ {Mary: $\neg$Person},

$KB_S \cup$ {Mary: $\neg$DFM},

$KB_S \cup$ {jacm48p885: $\neg$Publication},

are checked. By the ABox statements Mary: FM and ‹Mary, DEng›: highestDegree and the definition of DFM, KB_S entails Mary: DFM. Obviously KB_S also entails Mary: Person and jacm48p885: Publication. As a result, the three augmented knowledge bases are all unsatisfiable, and the substitution {X/Mary, Y/IR} is taken as an answer. Next, from the second collection, four augmented knowledge bases, i.e.

KB_S ∪ {John: ¬Person},

KB_S ∪ {John: ¬Person, ITS413: ¬AdvCourse},

KB_S ∪ {John: ¬(DFM ⊓ ∃supervise.Grad), John: ¬Person},

KB_S ∪ {John: ¬(DFM ⊓ ∃supervise.Grad), ITS413: ¬AdvCourse},

are considered. By the ABox statements ‹ITS413, Bob›: takenBy and Bob: Underclassman and the necessary condition for membership of AdvCourse, every model of KB_S requires ITS413 not to be an instance of AdvCourse. Furthermore, no assertion of KB_S prevents the possibility of constructing a model in which John does not supervise any students.[6] The fourth augmented knowledge base is thus not unsatisfiable. Accordingly, the substitution {X/John, Y/DL} is discarded.

Results

Two experiments were conducted. The objective of the first experiment is to investigate the feasibility of the approach in terms of the performance of EXTENDED-BC. The second experiment aims to compare the performance of RACER for constraint checking in the structural subsystem with that of traditional backward chaining. The experiments were all performed on a standard PC with Pentium IV processor (1.7 GHz, 512 MB RAM), running Windows-2000 operating system. Both EXTENDED-BC. and traditional backward chaining were implemented in the COMMONLISP language. RACER version 1.7 for Windows was used in the second experiment.

In the first experiment, two groups of tests (Groups-A1, and -A2) were conducted. Each group consists of 16 relational knowledge bases, with varying sizes of rule sets (from 10 to 85 rules). A rule in each set contains 3 body atoms and 4 distinct related variables. Rules in the same set are interrelated and are all employed for derivation of answers to tested queries (i.e. removal of any single rule would result in reduction of the number of answers). The number of ground facts that match each body atom of a rule involved in the lowest level of a proof tree is 2 for Group-A1 and 10 for Group-A2. Figure 9 shows the amount of time required by EXTENDED-BC. for collecting constraints for all possible answers to a query involving 2 unknown variables in the two groups. The figure indicates that the algorithm scales, in terms of size of the rule sets, and has a polynomial-bounded time complexity of order n^5, where n is the number of rules.

DL statements of some forms can be translated equivalently into Horn rules. The translation is illustrated in Figure 10, where C, C_1, C_2, and D are concept names, and R is a role name. In the second experiment, comparison between the performance of RACER with that of traditional backward chaining was made when a structural subsystem is restricted to contain only DL statements of these forms. Four groups of tests (Groups-B1, -B2, -B3, and -B4) were conducted. A DL inclusion statement in each group has the form

$$D \sqsubseteq C_1 \sqcap C_2 \sqcap \forall R.C_3,$$

which can be translated, according to Figure 10, into the three Horn rules

$$C_1(x) \leftarrow D(x),\ C_2(x) \leftarrow D(x),\ C_3(y) \leftarrow D(x) \wedge R(x,y).$$

A structural subsystem employed by each test is characterized by four parameters, *NI*1, *NI*2, *NR*, and *Depth*, where, as depicted by Figure 11, *NI*1 denotes the number of instances of the highest-level concept, *NI*2 denotes the number of individuals that are explicitly declared as instances of any other concept, *NR* denotes the number of role fillers that are used for constructing relationships among individuals involving the highest-level concept, and *Depth* denotes the number of concept-dependency levels. The parameters of the structural sub-system used in each group are given in Table 3.

[6] *The last statement in the ABox in Figure 5 only asserts that every person supervised by John is a graduate student.*

Each group in the second experiment consists of two tests with different number of concept-dependency levels. Two constraint-checking queries were performed in each test. Referring to Figure 11, the first query (Q1) checks whether an instance of the highest-level concept (e.g. the individual j_{01}) belongs to the leftmost concept at the lowest level (e.g. the concept An_1). The second query (Q2) checks whether an instance that is least-directly related to an instance at the highest-level concept (e.g., the individual jn_1 in the figure) belongs to the rightmost deepest concept (not shown in the figure). The tests were first performed using RACER, where instance checking is reduced into knowledge base unsatisfiability checking, i.e., checking whether an individual a is an instance of *a* concept *C* with respect to a given knowledge base is equivalent to checking whether addition of the assertion $a: \neg C$ makes the knowledge base unsatisfiable. In each test, RACER (Version 1.7) answered each query within less than 0.30 second. The same tests were next performed using a traditional backward chaining algorithm, and the results are shown in Table 4 (the last two columns show execution time). This experiment shows that RACER is more efficient than the traditional backward chaining algorithm for constraint-checking tasks. It thus supports the proposed hybrid reasoning approach based on an integration of RACER with the backward chaining through EXTENDED-BC, and also, more generally, an integration of an available optimized specialized reasoner with a general purpose one.

Discussion

In Nantajeewarawat and Wuwongse (2001) and Wuwongse and Nantajeewarawat (2002), the effect of implicit implication caused by a generalization taxonomy of concepts upon the semantics of Horn-clause-style declarative programs and the interaction between deduction and inheritance were investigated. A structural component considered therein, however, deals only with primitive concepts, i.e. a concept can only be described in terms of its (explicit) extension and its generalization relationship with other concepts, and there is no mechanism for defining a new concept by providing sufficient and necessary conditions for its membership. In such a structural part, one can assert, for example, that DFM is more specific than FM, but not that ìholding a doctoral degreeî is a sufficient and necessary condition for an individual of FM to be an instance of DFM. By using DL as its underlying formalism, a structural component discussed in this paper is far more expressive than that considered in Nantajeewarawat and Wuwongse (2001) and Wuwongse and Nantajeewarawat (2002), and terminological reasoning mechanisms on this component are also more sophisticated.

In most implementations of DL, all reasoning tasks are reduced into the task of determining knowledge base (un)satisfiability, e.g. a concept assertion is inferred if and only if a given knowledge base is not satisfiable when its ABox is augmented with the negation of that concept assertion. This technique, however, cannot be used directly to infer a role assertion, since most DL systems do not support the role negation. As a result, the traditional DL systems are rather weak in answering queries involving relationships among individuals.[7] This problem is sidestepped in this paper by augmenting a DL knowledge base with a rule-based relational component, and pushing (part of) representation of relations and reasoning in the level of individuals to the relational component. A different approach was taken in Horrocks and Tessaris (2000) and Tessaris (2001) based on a method of evaluating *boolean* conjunctive queries. A query of this type is a conjunction of concept assertions and/or role assertions extended by an incorporation of variables for imposition of equality constraints and for representation of anonymous individuals. A technique, called *rolling up* a query, was proposed for converting role assertions into concept assertions, which can then be evaluated through

[7] *Even conjunctive queries-the least expressive query languages usually considered in the database literature-are often not supported (Borgida, 1996).*

usual reduction to a knowledge base (un) satisfiability problem. Although this technique is useful for determining the truth value of a boolean conjunctive query, it does not yield any variable substitution as an answer. Consequently, retrieving individuals that make a boolean query true can only be achieved by a repeated application of boolean queries with all possible individual names substituted for variables. Obviously, such a retrieval tends to be prohibitively expensive.

In Grosof *et al.* (2003), the expressiveness intersection of DL and Horn logic programs was investigated, and an intermediate knowledge representation framework, called Description Logic Program (DLP), which is contained within the intersection, is defined. A bi-directional translation from the DLP fragment of DL to logic programs, and vice versa from the DLP fragment of logic programs to DL was demonstrated. This translation enables one to "build rules on top of ontologies", i.e. it enables a rule-based representation to have access to DL ontological definitions for vocabulary primitives used by rules. Conversely, it enables one to "build ontologies on top of rules", i.e. it enables ontological definitions to be supplemented by rules, or imported into DL from rules. Since DLP is contained in the intersection of DL and Horn logic programs, the approach presented in Grosof *et al.* (2003) extends neither the expressive power of DL nor that of Horn logic. There exist DL statements as well as Horn rules that are inherently outside the intersection of the two formalisms, and are thus not included by DLP.[8] In comparison, the hybrid system considered in this paper utilizes the full power of DL (in its structural subsystem), and extends the expressive power of Horn rules by incorporation of DL-based membership constraints.

Conclusions

It has long been realized that the Web could benefit from having its content understandable in a machine processable form, and it is widely agreed that ontologies will play a key role in providing much enabling infrastructure to achieve this goal. While DL is well suited for representation of ontologies and reasoning in the level of concepts, its capabilities of reasoning with instances are rather limited. This area is a stronghold of rules, which offer extensive facilities for instance reasoning. It is therefore interesting to explore the possibility of combining DL with the rule paradigm in order to support expressive instance queries with respect to terminological knowledge bases. A hybrid system comprising a DL-based structural subsystem and a rule-based relational subsystem is discussed. Whereas it inherits appealing computational characteristics from conventional backward chaining with Horn rules, the presented system can easily be adapted to most existing (and future) DL implementations.

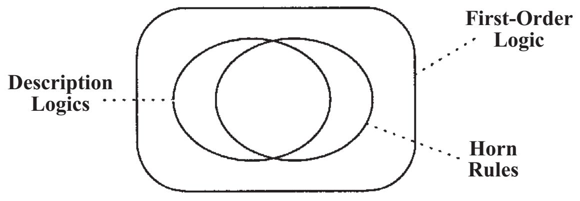

Figure 1. Expressiveness overlap of DL and Horn rules

[8] *Among others, in DLP, neither disjunction (DL ∪) nor existential restriction (DL ∃) may occur in the right-hand side of an inclusion statement, no universal restriction (DL ∀) may occur in its left-hand side, and DL negation is completely not allowed.*

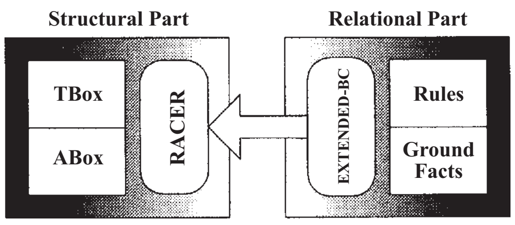

Figure 2. An overview of the proposed hybrid system

C, D →	*A*	(primitive concept)
	| top	(most general concept)
	| bottom	(absurd concept)
	| $C \cap D$	(conjunction)
	| $C \cup D$	(disjunction)
	| $\neg C$	(negation)
	| $\forall R.C$	(universal quantification)
	| $\exists R.C$	(existential quantification)
	| $\geq n\ R.C$	(at-least restriction)
	| $< n\ R.C$	(at-most restriction)

Figure 3. Syntax of basic concept expressions

```
<owl:Class rdf:about="kb1.owl#AdvCourse">
  <rdfs:label>Advanced Course</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf>
        <owl:Class rdf:about="kb1.owl#Course"/>
        <owl:ComplementOf>
          <owl:Restriction>
             <owl:onProperty rdf:resource="kb1.owl#takenBy"/>
             <owl:someValuesFrom>
                <owl:Class rdf:about="kb1.owl#Underclassman"/>
             </owl:someValuesFrom>
          </owl:Restriction>
        </owl:ComplementOf>
      </owl:intersectionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

(a) An OWL statement

```
<rdf:Description rdf:about="kb1.owl#John">
  <rdfs:label>John</rdfs:label>
  <rdf:type>
     <owl:Class rdf:about="kb1.owl#Prof"/>
  </rdf:type>
  <highestDegree rdf:resource="kb1.owl#PhD"/>
</rdf:Description>
```

(b) An RDF statement

Figure 4. An OWL DL axiom and an RDF annotation

TBox
$\mathrm{FM} \sqsubseteq \mathrm{Person}$
$\mathrm{Prof} \sqsubseteq \mathrm{FM}$
$\mathrm{DFM} \equiv \mathrm{FM} \sqcap \exists \mathrm{highestDegree.DoctoralDegree}$
$\mathrm{AdvCourse} \sqsubseteq \mathrm{Course} \sqcap \neg\exists \mathrm{takenBy.Underclassman}$
$\mathrm{Underclassman} \sqsubseteq \mathrm{Undergrad}$
$\mathrm{Undergrad} \sqcup \mathrm{Grad} \sqsubseteq \mathrm{Person}$
$\mathrm{Undergrad} \sqcap \mathrm{Grad} \sqsubseteq \mathrm{bottom}$
ABox
PhD: DoctoralDegree, DEng: DoctoralDegree,
John: Prof, Mary: FM,
jacm48p885: Publication,
ITS413: Course,
Bob: Underclassman,
$\langle$John, PhD$\rangle$: highestDegree, $\langle$Mary, DEng$\rangle$: highestDegree,
$\langle$ITS413, Bob$\rangle$: takenBy,
John: $\forall$supervise.Grad

Figure 5. A structural part

R1. find(A,T) $\Leftarrow$ myInterest(T), expert(A, T), A: Person.
R2. expert(A,T) $\Leftarrow$ keyword(P, T), author(P, A), A: DFM, P: Publication.
R3. expert(A,T) $\Leftarrow$ researchInterest(A, T), A: DFM $\sqcap$ $\exists$supervise.Grad.
R4. expert(A,T) $\Leftarrow$ cover(C, T), teach(A, C), C: AdvCourse.

F1. myInterest(DL).
F2. myInterest(IR).
F3. author(jacm48p885, Mary).
F4. keyword(jacm48p885, IR).
F5. cover(ITS413, DL).
F6. researchInterest(John, DL).
F7. teach(John, ITS413).

Figure 6. A relational part

```
<rdf:Description rdf:about="papers.owl#jacm48p885">
        <author rdf:resource="persons.owl#Mary"/>
        <keyword rdf:resource="subjects.owl#IR"/>
</rdf:Description>
```

Figure 7. An RDF document

```
procedure EXTENDED-BC($KB_R$, $qlist, clist, \theta$)
input a set $KB_R$ of extended Horn rules; a list qlist of atoms;
        a list clist of constraints; a substitution $\theta$
output a set of substitution-constraints pairs
local variable
        an atom $q$; a list nextql of atoms; a list nextcl of constraints;
        a set ans and a set answers of substitution-constraints pairs
begin
1. answers := ∅
2. if qlist is empty then return {<$\theta$, clist>}
3. $q$ := FIRST(qlist)
4. for each rule $r$ in $KB_R$ such that $q$ and Head($r$) are unifiable
5.    do begin
6.       $s$ := UNIFY($q$, Head($r$))
7.       nextql := Body($r$)$\sigma$ || REST(qlist)$\sigma$
8.       nextcl := clist$\sigma$ || Constr($r$)$\sigma$
9.       ans := EXTENDED-BC($KB_R$, nextql, nextcl, COMPOSE($\theta$,$\sigma$))
10.      answers := answers ∪ ans
11.   end
12. return answers
end
```

Figure 8. The extended backward-chaining algorithm

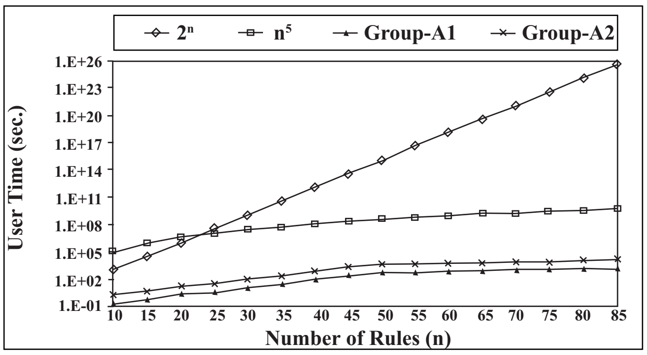

Figure 9. Performance of EXTENDED-BC

DL	Horn rule
$C \subseteq D$	$D(x) \leftarrow C(x)$
$a: C$	$C(a)$
$\langle a,b \rangle: RR(a,b)$	
$C \equiv D$	$D(x) \leftarrow C(x)$
	$C(x) \leftarrow D(x)$
$C_1 \cap C_2 \subseteq D$	$D(x) \leftarrow C_1(x) \wedge C_2(\mathrm{x})$
$\mathrm{D} \subseteq \mathrm{C1} \cap \mathrm{C2}$	$C_1(x) \leftarrow D(x)$
	$C_2(x) \leftarrow D(x)$
$\mathrm{C1} \cup \mathrm{C2} \subseteq \mathrm{D}$	$D(x) \leftarrow C_1(x)$
	$D(x) \leftarrow C_2(x)$
$C \subseteq \forall R.D$	$D(y) \leftarrow C(x) \wedge R(x,y)$
$\exists R.C \subseteq D$	$D(x) \leftarrow R(x,y) \wedge C(y)$

Figure 10. Mapping from DL statements to Horn rules

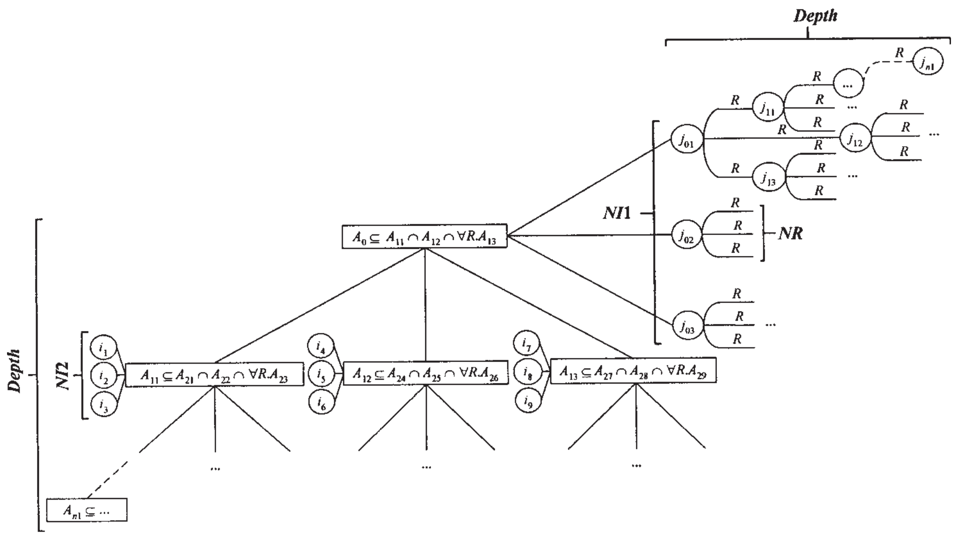
Depth
R
j_{n1}
j_{11}
j_{01}
j_{12}
j_{13}
NI1
j_{02}
NR
j_{03}
$A_0 \subseteq A_{11} \cap A_{12} \cap \forall R.A_{13}$
Depth
NI2
i_1 i_2 i_3
$A_{11} \subseteq A_{21} \cap A_{22} \cap \forall R.A_{23}$
i_4 i_5 i_6
$A_{12} \subseteq A_{24} \cap A_{25} \cap \forall R.A_{26}$
i_7 i_8 i_9
$A_{13} \subseteq A_{27} \cap A_{28} \cap \forall R.A_{29}$
$A_{n1} \subseteq \dots$

Figure 10. Mapping from DL statements to Horn rules

Table 1. OWL class constructors

Constructor	DL Syntax	Example
intersectionOf	$C_1 \cap \dots \cap C_n$	Human ∩ Male
unionOf	$C_1 \cup \dots \cup C_n$	Doctor ∪ Lawyer
complementOf	$\neg C$	¬Male
allValuesFrom	∀R.C	∀hasChild.Lawyer
someValuesFrom	∃R.C	∃hasChild.Doctor
minCardinality	$\geq n\ R.C$	≥2 hasChild.Male
maxCardinality	$\leq n\ R.C$	≤1 hasChild.Female

Table 2. OWL axioms

Axiom	DL Syntax	Example
rdfs:subClassOf	$C_1 \subseteq C_2$	Human ⊆ Animal ∩ Biped
owl:equivalentClass	$C_1 \equiv C_2$	Man ≡ Human ∩ Male
owl:disjointWith	$C_1 \subseteq \neg C_2$	Male ⊆ ¬ Female

Table 3. Parameters used in the second experiment

Parameter	**Group**			
	B1	**B2**	**B3**	**B4**
*NI*1	3	3	3	3
*NI*2	0	3	0	3
NR	1	1	3	3
Depth	4,5	4,5	4,5	4,5

Table 4. Results of using traditional backward chaining

Group	**Depth**	**#Concept**	**#Instance**	**#Property**	**Q1 (sec.)**	**Q2 (sec.)**
B1	4	40	3	12	0.046	0.343
	5	121	3	15	0.140	2.781
B2	4	40	120	12	0.046	0.453
	5	121	363	15	0.140	3.234
B3	4	40	3	36	0.031	0.906
	5	121	3	45	0.140	8.312
B4	4	40	120	36	0.046	1.109
	5	121	363	45	0.203	8.500

References

Antoniou, G., and van Harmelen, F. (2004). Web ontology language: OWL. In: Handbook on Ontologies. Staab, S., and Studer, R., (eds.). Springer, p. 67-92.

Baader, F., Burckert, H.-J., Hollunder, B., Nutt, W., and Siekman, J.H. (1990). Concept Logics. In: Computational Logics, Symposium Proceedings. Lloyd, J.W., (ed.). Springer-Verlag, p. 177-201.

Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., and Patel-Schneider, P.F. (2002). In: Description Logic Handbook. 1st ed. Cambridge University Press, 574 p.

Berners-Lee, T. (2000). Weaving the Web. 1st ed. Harper Business, 226 p.

Borgida, A. (1996). On the relative expressiveness of description logics and predicate logics. Artificial Intelligence, 82:353-367.

Calvanese, D., De Giacomo, G., Lenzerini, M., and Nardi, D. (2001). Reasoning in expressive description logics. In: Handbook of Automated Reasoning. Robinson, A., and Voronkov, A., (eds.). Elsevier Science, p.1,581-1,634.

Decker, S., Melnik, S., van Harmelen, F., Fensel, D., Klein M., Broekstra, J., Erdman, M., and Horrocks, I. (2000). The semantic web: The roles of XML and RDF. IEEE

Internet computing, 4(5):63-74.

Donini, F.M., Lenzerini, N., Nardi, D., and Schaerf, A. (1996). Reasoning in description logics. In: Principles of Knowledge Representation and Reasoning. Brewka, A. (ed). CLSI Publications, p. 193ñ238.

Donini, F.M., Lenzerini, M., Nardi, D., and Schaerf, A. (1998). AL-log: Integrating datalog and description logics. Journal of Intelligent Information Systems, 16: 227-252.

Fensel, D. (2001). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. 1st ed. Springer-Verlag, 138 p.

Grosof, B.N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: combining logic programs with description logic. Proceedings of the 12th International Conference on World Wide Web (WWW-2003); May 20ñ24, 2003; Budapest, Hungary, p. 48-57.

Haarslev, V., and Moller, R. (2001). RACER system description. Lecture Notes in Artificial Intelligence, 2,083:701-705.

Hendler, J. (2001). Agents and the semantic web. IEEE Intelligent Systems, 16(2):30-37.

Horrocks, I. (1998). The FaCT system. Lecture Notes in Artificial Intelligence, 1,397:307-312.

Horrocks, I., and Tessaris, S. (2000). A conjunctive query language for description logic ABoxes. Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000); July 30-August 3, 2000; Austin, Texas, USA, p. 399-404.

Nantajeewarawat, E., and Wuwongse, V. (2001). Defeasible inheritance through specialization. Computational Intelligence, 17(1):62-86.

Patel-Schneider, P.F., and Horrocks, I. (1999). DLP and FaCT. Lecture Notes in Artificial Intelligence, 1,617:19-23.

Russell, S. and Norvig, P. (1995). Artificial Intelligence: A Modern Approach. 1st ed. Prentice Hall, p. 265-294.

Staab, S. and Studer, R. (2004). Handbook on Ontologies. 1st ed. Springer, 660 p.

Tessaris, S. (2001). Questions and answers: reasoning and querying in description logic, [Ph.D. thesis]. Department of Computer Science,University of Manchester, UK, 199 p.

Ullman, J.D. (1989). Principles of database and knowledge base systems. 1st ed. Computer Science Press, 631 p.

Uschold, M., and Gruninger, M. (1996). Ontologies: principles, methods and applications. Knowledge Engineering Review, 11(2): 93-136.

Vardi, M.Y. (1997). Why is model logic so robustly decidable. In: Descriptive Complexity and Finite Models. N. Immerman, N., and Kolaitis, P., (eds.). American Mathematical Society, p. 149-184.

Wuwongse, V., and Nantajeewarawat, E. (2002). Declarative programs with implicit implication. IEEE Transactions on Knowledge and Data Engineering, 14(4): 836-849.