Working Paper, version of Nov. 21, 2002

Description Logic Programs: Combining Logic Programs with Description Logic

Benjamin N. Grosof
MIT Sloan School of Management
Cambridge, MA 02142, USA
`http://www.mit.edu/~bgrosof`
bgrosof@mit.edu

Ian Horrocks
University of Manchester
Manchester, UK
`http://www.cs.man.ac.uk/~horrocks`
horrocks@cs.man.ac.uk

ABSTRACT

We show how to interoperate, semantically and inferentially, between the leading Semantic Web approaches to rules (RuleML Logic Programs) and ontologies (OWL/DAML+OIL Description Logic) via analyzing their expressive intersection. To do so, we define a new intermediate knowledge representation (KR) contained within this intersection: *Description Logic Programs (DLP)*, and the closely related *Description Horn Logic (DHL)* which is an expressive fragment of first-order logic (FOL). DLP provides a significant degree of expressiveness, substantially greater than the RDF-Schema fragment of Description Logic.

We show how to perform *DLP-fusion*: the bidirectional translation of premises and inferences (including typical kinds of queries) from the DLP fragment of DL to LP, and vice versa from the DLP fragment of LP to DL. In particular, this translation enables one to "build rules on top of ontologies": it enables the rule KR to have access to DL ontological definitions for vocabulary primitives (e.g., predicates and individual constants) used by the rules. Conversely, the DLP-fusion technique likewise enables one to "build ontologies on top of rules": it enables ontological definitions to be supplemented by rules, or imported into DL from rules. It also enables available efficient LP inferencing algorithms/implementations to be exploited for reasoning over large-scale DL ontologies.

Keywords

Semantic Web, rules, ontologies, logic programs, Description Logic, knowledge representation, XML, RDF, model-theoretic semantics, inferencing, interoperability, translation, information integration

1. INTRODUCTION

The challenge we address in this paper is how and why to combine rules with ontologies for the Semantic Web (SW). This is a large topic, on which doubtless many more papers will be written. In this paper, we focus on meeting a few key requirements, via a few key initial steps of logical knowledge representation (KR). We start from the currently leading draft standards for SW ontologies[1] (OWL) [6] and for SW rules[2] (RuleML).[3] We then develop a semantics for combining such rules with such ontologies, based on a translation between their respective KRs. This semantics exploits correspondences with classical First Order Logic (FOL). We then show how this translation semantics supports several important patterns of KR usage, including querying, and discuss how it bestows a variety of benefits in these. Finally, we point the way to several interesting directions for future work.

In the remainder of this section and the next, we give in more detail the motivation and technical overview.

Layering in the Semantic Web Stack: The Semantic Web can be viewed as largely about "KR meets the Web". Over the last two years or so, a broad consensus has evolved in the Semantic Web community that the vision of the Semantic Web includes, specifically, rules as well as ontologies. A key requirement for the Semantic Web's architecture overall, then, is to be able to layer rules on top of ontologies — in particular to create and reason with rulebases that mention vocabulary specified by ontological knowledge bases — and to do so in a semantically coherent and powerful manner. The current version of the W3C's Semantic Web stack diagram, given in Figure 1, reflects this idea / consensus view.

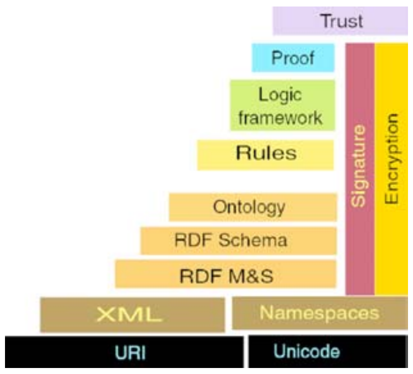

Figure 1: Semantic web stack diagram, from W3C.

[1]An ontology is a formally specified/defined vocabulary.

[2]A fact is a special case of a rule. A relational database, including its queries, can be viewed as a rulebase.

[3]Rule Markup Language Initiative `http://www.ruleml.org` and `http://ebusiness.mit.edu/bgrosof/#RuleML`; see [12] for a relatively recent overview paper.

.

Focus KRs: As an ontology language, we focus on DAML+OIL; this is based on the particular KR known as Description Logic (DL)[4], and is the point of departure for the recently-formed W3C Web-Ontology (WebOnt) Working Group, whose current draft standard is called OWL. OWL is based on a DL very close to DAML+OIL's, and is is very similar to DAML+OIL in many aspects, including in its approach to syntax based on RDF. However, since OWL is still under active development, while DAML+OIL is relatively stable, we focus on the particular DL used in DAML+OIL.

As a rule language, we focus on RuleML; this is based on the particular KR known as (declarative) logic programs (LP), and is the leading current standardization approach to rules for the Semantic Web. RuleML syntax has both an RDF version and a (non-RDF) XML version (these are semantically the same). In tandem with LP, we also focus on the (positive) Horn expressive fragment[5] of FOL, which is closely related to the positive Horn expressive fragment of LP .

Motivation from Semantic Web Services: A task-oriented motivation for combining RuleML LP rules with OWL/DAML+OIL DL ontologies arises from the efforts to design and build *Semantic Web Services (SWS)*. Semantic Web Services are Web Services that make use of Semantic Web techniques to describe (or implement) services in a knowledge-based manner. The knowledge-based service descriptions may be used for a variety of purposes, including: discovery and search; selection, evaluation, negotiation, and contracting; composition and planning; execution; and monitoring. Efforts to develop Semantic Web Services techniques and to explore their application scenarios include: the DAML-Services effort (DAML-S)[6], the Web Service Modeling Framework effort (WSMF)[7], SweetDeal e-contracting [13, 19, 14], and ECOIN financial knowledge integration [9, 10]. Both the DAML-S and SweetDeal efforts have specifically identified combining rules with ontologies as an important requirement. DAML-S began with DAML+OIL as its main tool for describing services. DAML-S then identified LP rules as desirable in addition. Interestingly, DAML-S has identified LP rules as desirable even to specify ontologies, partly because LP rules are more familiar to mainstream software engineers than DL is.

2. OVERVIEW OF THE APPROACH

In this section, we give an overview of our approach, and outline the rest of the paper.

We start with the goal of understanding the relationship between the two logic based KR formalisms (so as to be able to combine knowledge taken from both): Description Logics (decidable fragments of FOL closely related to propositional modal and dynamic logics [20]), and Logic Programs (see, e.g., [2] for review) which in turn is closely related to the Horn fragment of FOL. We further focus on def-Horn (a large fragment of Horn FOL), and then go on to show how both DL and LP are related to def-Horn. Highly efficient LP reasoning engines can be used to provide reasoning services for def-Horn.

Our approach is driven by the insight that understanding the expressive *intersection* of these two KRs will be crucial to understanding the expressive *combination/union* of the two KRs. We define a new intermediate KR called *Description Horn Logic (DHL)*, which is contained within this intersection (and so is also a fragment of FOL), and the closely related *Description Logic Programs (DLP)*, which can be viewed as DHL with a moderate weakening of what kind of conclusions can be drawn.

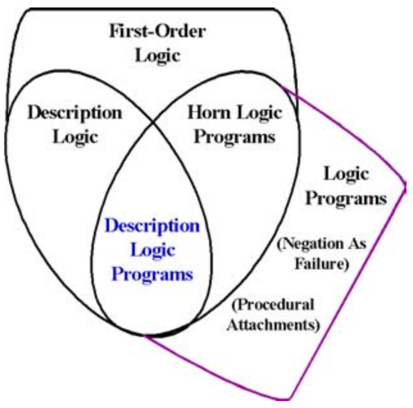

Figure 2: Expressive overlap of DL with LP.

Figure 2 illustrates the relationship between the various KRs and their expressive classes. DL and Horn are strict (decidable) subsets of FOL. LP, on the other hand, intersects with FOL but neither includes nor is fully included by FOL. FOL can express (positive) disjunctions, which are inexpressible in LP. There are, however, expressive features of LP, frequently used in practical rule-based applications, that are inexpressible in FOL. One is negation-as-failure, a basic kind of logical non-monotonicity. Another is procedural attachments, e.g., the association of action-performing procedural invocations with the drawing of conclusions about particular predicates.

Description Logic Programs, our newly defined intermediate KR, is contained within the intersection of DL and LP. "Full" LP, including non-monotonicity and procedural attachments, can thus be viewed as including an "ontology sub-language", namely the DLP subset of DL.

Rather than working from the intersection as we do in this paper, one may instead directly address the expressive union of DL and LP by studying the expressive union of DL and Horn within the overall framework of FOL. This is certainly an interesting thing to do. However, to our knowledge, this has not yet been well characterized theoretically, e.g., it is unclear how, if at all, such a union differs from full FOL.

Full FOL has some significant practical and expressive drawbacks as a KR in which to combine DL and rules. First, full FOL has severe computational complexity: it is undecidable in the general case, and intractable even under the Datalog restriction. Second, it is not understood even at a basic research level how to expressively extend full FOL to provide non-monotonicity and procedural attachments; yet these are crucial expressive features in many (perhaps most) practical usages of rules. Third, full FOL and its inferencing techniques are unfamiliar to the great majority of mainstream software engineers, whereas rules (e.g., in the form of SQL-type queries, or Prolog) are familiar conceptually to many of them. The approach we take in this paper avoids these drawbacks by avoiding directly tackling the union (of DL and Horn) in FOL.

DLP provides a significant degree of expressiveness. It is a large fragment of the intersection of DL and LP/Horn, and includes the RDF-Schema (RDFS) [4] fragment of DL.

The RDFS fragment of DL permits: stating that a class D is a

[4]Actually, "Description Logic" is often used to mean more broadly a family of variant KRs, of which the one used in DAML+OIL is just one member. We will define later the precise DL KR upon which we focus.

[5]expressive "fragment" means expressive subset or special case

[6]`http://www.daml.org/services`

[7]`http://informatik.uibk.ac.at/users/c70385/wese/index.html`

Subclass of a class E; stating that the *Domain* of a property P is a class C; stating that the *Range* of a property P is a class C; stating that a property P is a *Subproperty* of a property Q; stating that an individual b is an *Instance* of a class C; and stating that a pair of individuals (a,b) is an *Instance* of a property P.

Additional DLP expressively permits (within DL): using the *Intersection* connective (conjunction) within *class* descriptions (i.e., in C, D, or E above); using the *Union* connective (disjunction) within *subclass* descriptions (i.e., in D above); using (a restricted form of) *Universal* quantification within *superclass* descriptions (i.e., in E above); using (a restricted form of) *Existential* quantification within *subclass* descriptions (i.e., in D above); stating that a property P is *Transitive*; stating that a property P is *Symmetric*; and stating that a property P is the *Inverse* of a property Q. In RDFS, in contrast, the classes (i.e., C, D, E above) are atomic primitives—they may not have connectives or quantifiers appearing within them.

Via the DLP KR, we give a new technique to combine DL and LP. We show how to perform *DLP-fusion*: the bidirectional mapping of premises and inferences (including typical kinds of queries) from the DLP fragment of DL to LP, and vice versa from the DLP fragment of LP to DL. We call it "DLP-fusion" because it fuses the two logical KRs—DL and LP—-so that information from each can be used in the other. The DLP-fusion technique promises several benefits. We say "promises" because we present in this paper mainly a theoretical basis; development of detailed algorithms and implementations remain for future work.

In particular, DLP-fusion enables one to "build rules on top of ontologies": it enables the rule KR to have access to DL ontological definitions for vocabulary primitives (e.g., predicates and individual constants) used by the rules. Conversely, the technique enables one to "build ontologies on top of rules": it enables ontological definitions to be supplemented by rules, or imported into DL from rules. It also enables efficient LP inferencing algorithms/implementations, e.g., rule or relational DBMS[8] engines, to be exploited for reasoning over large-scale DL ontologies.

Organization of Rest of paper: In Section 3, we give formal preliminaries about DL, Horn, and LP, including typical kinds of queries in each. In Section 4, we give a semantic mapping, i.e., a translation from a portion of DL into def-Horn. In Section 5, we define the resulting fragment of FOL to be DHL, and define DLP as the corresponding fragment of LP; it is slightly weaker than DHL. In Section 6, we give some more discussion of the process of translating between DL, DHL, DLP, and LP. In Section 7, we show how to combine/inter-operate DL and LP knowledge via a correspondence between the standard inference problems in each KR language. Finally, in Section 8, we summarise what has been achieved so far and discuss directions for future work.

3. PRELIMINARIES

In this section we will introduce Horn Logic, Description Logic (DL) and the DL based ontology language DAML+OIL. In particular, we will describe their syntax and formalise their meaning in terms of classical First Order Logic (FOL).

3.1 DAML+OIL and Description Logic

DAML+OIL is an ontology language designed for use on the (semantic) web. Although DAML+OIL is syntactically "layered" on top of RDFS, semantically it is layered on a subset of RDFS. This subset does *not* include RDFS's recursive meta model (i.e., the unrestricted use of the type relation), but instead treats RDFS as a very simple DL supporting only atomic class names. Like other DLs, this "DAML+OIL subset" of RDFS corresponds to a fragment of classical FOL, making it much easier to develop mappings to rule languages as well as to DLs. From now on, when we talk about RDFS, we will be referring to the DAML+OIL subset of RDFS.

[8]Data Base Management Systems, e.g., SQL query answering systems

Constructor	DL Syntax	Example
`intersectionOf`	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
`complementOf`	$\neg C$	$\neg$Male
`oneOf`	$\{i_1 \ldots i_n\}$	$\{$john, mary$\}$
`hasClass`	$\exists P.C$	$\exists$hasChild.Lawyer
`toClass`	$\forall P.C$	$\forall$hasChild.Doctor
`hasValue`	$\exists P.\{i\}$	$\exists$citizenOf.$\{$USA$\}$
`minCardinalityQ`	$\geqslant n\, P.C$	$\geqslant 2$ hasChild.Lawyer
`maxCardinalityQ`	$\leqslant n\, P.C$	$\leqslant 1$ hasChild.Male
`cardinalityQ`	$= n\, P.C$	$= 1$ hasParent.Female

Figure 4: DAML+OIL class constructors

DAML+OIL is equivalent to a very expressive DL—in fact it is equivalent to the $\mathcal{SHOIQ}(\mathbf{D})$ DL [17, 15]. In addition to "abstract" classes and individuals, DAML+OIL also supports the use of "concrete" datatypes and data values (the $(\mathbf{D})$ in $\mathcal{SHOIQ}(\mathbf{D})$). In this paper, however, we will restrict our attention to the abstract part of the language, which corresponds to the $\mathcal{SHOIQ}$ DL.

Figure 3 (on page 4) shows how DAML+OIL statements correspond to $\mathcal{SHOIQ}$ axioms, where C (possibly subscripted) is a class, P (possibly subscripted) is a property, P^- is the inverse of P, P^+ is the transitive closure of P, i (possibly subscripted) is an individual and $\top$ is an abbreviation for $A \sqcup \neg A$ for some class A (i.e., the most general class, called "Thing" in DAML+OIL).

It can be seen that all DAML+OIL statements can be reduced to class/property inclusion axioms and ground facts (asserted class-instance and instance-property-instance relationships).[9] In the case of `transitiveProperty`, however, the axiom $P^+ \sqsubseteq P$ is taken to be equivalent to asserting that P is a transitive property (like DAML+OIL, $\mathcal{SHOIQ}$ does not include the transitive closure operator).

As in any DL, DAML+OIL classes can be names (URIs) or *expressions*, and a variety of *constructors* are provided for building class expressions. Figure 4 summarises the available constructors and their correspondence with $\mathcal{SHOIQ}$ class expressions.

Formally, $\mathcal{SHOIQ}$ is built over a signature of distinct sets of class ($\mathcal{CN}$), property ($\mathcal{RN}$) and individual ($\mathcal{O}$) names.[10] The set of all $\mathcal{SHOIQ}$ properties is equal to the set of property names $\mathcal{RN}$ union the set of the inverse properties $\{R^- \mid P \in \mathcal{RN}\}$. In addition, we distinguish *simple* properties, where a simple property is a $\mathcal{SHOIQ}$ property that is neither transitive, nor has any transitive sub-properties, and whose inverse is also a simple property.

The set of all $\mathcal{SHOIQ}$ classes is the smallest set such that every class name in $\mathcal{CN}$ and the symbols $\top$, $\bot$ are classes, and if C,D are classes, i is an individual name from $\mathcal{O}$, R is a property, S is a simple property and n an integer, then $\neg C$, $\{i\}$, $(C \sqcap D)$, $(C \sqcup D)$, $(\forall R.C)$, $(\exists R.C)$, $\geqslant n\, S.C$, and $\leqslant n\, S.C$ are classes.

The semantics of $\mathcal{SHOIQ}$ is given by interpretations, where an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty domain $\Delta^{\mathcal{I}}$ and a interpretation function $\cdot^{\mathcal{I}}$. The interpretation function maps classes into subsets of $\Delta^{\mathcal{I}}$, individual names into elements of $\Delta^{\mathcal{I}}$, and property names into subsets of the cartesian product of $\Delta^{\mathcal{I}}$

[9]Equivalence axioms can be reduced to a symmetrical pair of inclusion axioms.

[10]What DAML+OIL calls classes and properties are normally called concepts and roles in a DL, but to avoid confusion we will use class and property throughout this paper.

Axiom	DL Syntax	Example
`subClassOf`	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
`sameClassAs`	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
`subPropertyOf`	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
`samePropertyAs`	$P_1 \equiv P_2$	cost $\equiv$ price
`disjointWith`	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
`sameIndividualAs`	$\{i_1\} \equiv \{i_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
`differentIndividualFrom`	$\{i_1\} \sqsubseteq \neg\{i_2\}$	{john} $\sqsubseteq \neg${peter}
`inverseOf`	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
`transitiveProperty`	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
`uniqueProperty`	$\top \sqsubseteq \leqslant 1\, P.\top$	$\top \sqsubseteq \leqslant 1$ hasMother.$\top$
`unambiguousProperty`	$\top \sqsubseteq \leqslant 1\, P^-.\top$	$\top \sqsubseteq \leqslant 1$ isMotherOf$^-$.$\top$
`range`	$\top \sqsubseteq \forall P.C$	$\top \sqsubseteq \forall$hasParent.Human
`domain`	$\top \sqsubseteq \forall P^-.C$	$\top \sqsubseteq \forall$hasParent$^-$.Human
i `type` C	$i : C$	john : Man
$i_1\, P\, i_2$	$\langle i_1, i_2\rangle : P$	$\langle$john, peter$\rangle$: hasParent

Figure 3: DAML+OIL statements and $\mathcal{SHIQ}$ axioms

$(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$. Compound class expressions are interpreted according to the following equations (see [21])

$$\begin{array}{rclrcl}
\top^{\mathcal{I}} &=& \Delta^{\mathcal{I}} & (C \sqcap D)^{\mathcal{I}} &=& C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
\bot^{\mathcal{I}} &=& \emptyset & (C \sqcup D)^{\mathcal{I}} &=& C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
\neg C^{\mathcal{I}} &=& \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & \{i\}^{\mathcal{I}} &=& \{i^{\mathcal{I}}\}
\end{array}$$

$$\begin{array}{rcl}
(\forall R.C)^{\mathcal{I}} &=& \{x \in \Delta^{\mathcal{I}} \mid \forall y(x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &=& \{x \in \Delta^{\mathcal{I}} \mid \exists y(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \\
(\geqslant n\, R.C)^{\mathcal{I}} &=& \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\} \\
(\leqslant n\, R.C)^{\mathcal{I}} &=& \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}
\end{array}$$

A property and its inverse must be interpreted according to the equation

$$(R^-)^{\mathcal{I}} = \{(x,y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (y,x) \in R^{\mathcal{I}}\}.$$

In addition, the interpretation function must satisfy the transitive restriction on property names; i.e., for any $R \in \mathcal{TRN}$ if $(x,y) \in R^{\mathcal{I}}$ and $(y,z) \in R^{\mathcal{I}}$, then $(x,z) \in R^{\mathcal{I}}$.

Axioms $C \sqsubseteq D$, $i : C$, $\langle i_1, i_2\rangle : R$ and $R \sqsubseteq R'$ are satisfied by an interpretation $\mathcal{I}$ iff respectively $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $i^{\mathcal{I}} \in C^{\mathcal{I}}$, $(i_1^{\mathcal{I}}, i_2^{\mathcal{I}}) \in R^{\mathcal{I}}$ and $R^{\mathcal{I}} \sqsubseteq R'^{\mathcal{I}}$; an ontology $\mathcal{O}$ is satisfied by $\mathcal{I}$ iff $\mathcal{I}$ satisfies every axiom in $\mathcal{O}$; $C \sqsubseteq D$, $i : C$ and $\langle i_1, i_2\rangle : R$ w.r.t. $\mathcal{O}$ iff respectively $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $i^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(i_1^{\mathcal{I}}, i_2^{\mathcal{I}}) \in R^{\mathcal{I}}$ in every interpretation $\mathcal{I}$ of $\mathcal{O}$.

The meaning of $\mathcal{SHOIQ}$ can also be seen in terms of a correspondence to FOL, where classes correspond to unary predicates (predicates of arity 1), properties correspond to binary predicates (predicates of arity 2), and subclass/property axioms correspond to implication [7, 3].

To be more precise, individuals are equivalent to FOL constants, classes and class expressions are equivalent to FOL formulae with one free variable, and properties (and property expressions when supported by the DL) are equivalent to FOL formulae with two free variables. Class and property inclusion axioms are equivalent to FOL sentences consisting of an implication between two formulae with the free variables universally quantified at the outer level. E.g., a DL axiom of the form $C \sqsubseteq D$ is equivalent to a FOL sentence of the form $\forall x.C(x) \to D(x)$. DL axioms of the form $a : C$ and $\langle a, b\rangle : P$ correspond to ground atoms $C(a)$ and $P(a,b)$. Finally, DL axioms asserting the transitivity of a property P, the functionality of a property P and that property Q is the inverse of property P are equivalent to FOL sentences of the form $\forall x,y,z.(P(x,y) \wedge P(y,z)) \to P(x,z)$, $\forall x,y,z.(P(x,y) \wedge P(x,z)) \to y = z$ and $\forall x,y.P(x,y) \iff Q(y,x)$ respectively.

DL	FOL
$a : C$	$C(a)$
$\langle a, b\rangle : P$	$P(a,b)$
$C \sqsubseteq D$	$\forall x.C(x) \to D(x)$
$P^+ \sqsubseteq P$	$\forall x,y,z.(P(x,y) \wedge P(y,z)) \to P(x,z)$
$\top \sqsubseteq \leqslant 1\, P$	$\forall x,y,z.(P(x,y) \wedge P(x,z)) \to y = z$
$P \equiv Q^-$	$\forall x,y.P(x,y) \iff Q(y,x)$
$C_1 \sqcap \ldots \sqcap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$
$C_1 \sqcup \ldots \sqcap C_n$	$C_1(x) \vee \ldots \vee C_n(x)$
$\neg C$	$\neg C(x)$
$\{a_1, \ldots, a_n\}$	$x = a_1 \vee \ldots \vee x = a_n$
$\exists P.C$	$\exists y.(P(x,y) \wedge C(y))$
$\forall P.C$	$\forall y..(P(x,y) \to C(y))$
$\geqslant n\, P.C$	$\exists y_1, \ldots, y_n. \bigwedge_{1 \leqslant i \leqslant n}(P(x,y_i) \wedge C(y_i))$
	$\wedge \bigwedge_{1 \leqslant i < n, i < j \leqslant n} y_i \neq y_j$
$\leqslant (n-1)\, P.C$	$\forall y_1, \ldots, y_n.(\bigwedge_{1 \leqslant i \leqslant n}(P(x,y_i) \wedge C(y_i)))$
	$\to (\bigvee_{1 \leqslant i < n, i < j \leqslant n} y_i = y_j)$

Figure 5: DL FOL equivalence

Figure 5 summarises the above equivalences and shows the FOL formulae corresponding to the DL class expressions described in Figure 4, where a, b are constants, and x is the free variable. These formulae can be composed in the obvious way, e.g., $\exists R.(C \sqcap D) \equiv \exists y.(P(x,y) \wedge (C(y) \wedge D(y)))$.

As a notational convention we will, throughout the paper, use a and b for constants and w, x, y and z for variables.

3.2 Logic Programs and Horn FOL

Declarative logic programs (LP)[11] is the KR whose semantics underlies RuleML and, in large part, the four families of rule systems that are currently most commercially important — SQL relational databases, OPS5-heritage production rules, Prolog, and Event-Condition-Action rules. As we mentioned earlier, it is our focus KR for rules.

The commonly used expressiveness of full LP includes features, notably negation-as-failure/priorities and procedural attachments, that are not expressible in FOL — much less in DL. We thus concentrate on only an expressive portion of LP.

[11] see, e.g., [2] for a review

An *ordinary* (a.k.a. "normal"[12]) LP is a set of *rules* each having the form:

$H \;\leftarrow\; B_1 \wedge \ldots \wedge B_m \wedge \sim B_{m+1} \wedge \ldots \wedge \sim Bn$

where H, B_i are atoms (atomic formulae), and $n \geq m \geq 0$. Note that no restriction is placed on the arity of the predicates appearing in these atoms. Logical variables, and logical functions (with any arity), may appear unrestrictedly in these atoms.

H is called the *head* (a.k.a. *consequent*) of the rule;

$B_1 \wedge \ldots \wedge B_m \wedge \sim B_{m+1} \wedge \ldots \wedge \sim Bn$

is called the *body* (a.k.a. *antecedent*) of the rule. $\leftarrow$ is to be read as "if", so that the overall rule should be read as "[head] if [body]", i.e., "if [body] then [head]". If $n = 0$, then the body is empty, i.e., $True$, and notationally the " $\leftarrow$ " is often omitted. A *fact* is a rule whose body is empty and whose head is a ground atom. $\sim$ stands for negation-as-failure, a logically non-monotonic form of negation whose semantics differs, in general, significantly from the semantics of classical negation ($\neg$). Intuitively, $\sim B_i$ means "B_i is not believed" (i.e., is unknown or false), whereas $\neg$ means "B_i is false". Intuitively, each rule can be viewed as universally quantified at the outer level. More precisely, each rule can be viewed as standing for the set of all its ground instantiations.

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

$H \;\leftarrow\; B_1 \wedge \ldots \wedge B_m$

where H, B_i are atoms, and $m \geq 0$.

Definite LP is closely related syntactically and semantically to the Horn fragment of FOL, a.k.a. Horn-clause logic. A clause in FOL has the form:

$L_1 \vee \ldots \vee L_k$

where each L_i is a (classical) literal. A literal L has either the form (1) A or (2) $\neg A$, where A is an atom. The literal is said to be *positive* in case (1), or to be *negative* in case (2). A clause is said to be *Horn* when *at most one* of its literals is positive. A Horn clause is said to be *definite* when *exactly one* of its literals is positive. A definite Horn clause is also known as a *Horn rule*. A definite Horn clause, a.k.a. *Horn rule*, can thus be written in the form:

$H \;\leftarrow\; B_1 \wedge \ldots \wedge B_m$

where H, B_i are atoms, and $m \geq 0$. We say that this Horn rule *corresponds* to the definite LP rule that has the same syntactic form, and vice versa. Likewise, we say that a Horn ruleset $\mathcal{RH}$ and a definite LP ruleset $\mathcal{RP}$ correspond to each other when their rules do (isomorphically). We then also say that $\mathcal{RP}$ is the *LP-correspondent* of $\mathcal{RH}$, and conversely that $\mathcal{RH}$ is the *Horn-correspondent* of $\mathcal{RP}$.

As mentioned above, it is implicit in this notation that all logical variables are universally quantified at the outer level, i.e., over the scope of the whole clause. E.g., the rule

$man(x) \;\leftarrow\; human(x) \wedge male(x)$

can be written equivalently as:

$\forall x.\, man(x) \;\leftarrow\; human(x) \wedge male(x).$

Note the similarity with the FOL equivalent of a DL inclusion (subClassOf) axiom given in Figure 5.

An LP rule or Horn clause is said to be *equality-free* when the equality predicate does not appear in it. Likewise, each is said to be *Datalog* when no logical functions (of arity greater than zero) appear in it.

The semantics of an ordinary LP is defined to be a *conclusion set*, where each conclusion is a ground atom[13], i.e., fact, entailed by the LP. Note that we (cf. RuleML) adopt the *well-founded semantics*[14] for LP. Formally, the semantics of a definite LP $\mathcal{R}$ is defined as follows. Let HB stand for the Herbrand base of $\mathcal{R}$. The conclusion set $\mathcal{C}$ is the smallest (w.r.t. set inclusion) subset $\mathcal{S}$ of HB such that for any rule

$H \;\leftarrow\; B_1 \wedge \ldots \wedge B_m,$

if $B_1 \wedge \ldots \wedge B_m \in \mathcal{S}$ then $H \in \mathcal{S}$.

The relationship of LP semantics to FOL semantics is relatively simple to describe for the case of definite equality-free Datalog LP, which we call *def-LP*. The syntactically corresponding fragment of FOL is definite equality-free Datalog Horn FOL, which we call *def-Horn*. Let $\mathcal{RP}$ be a def-LP. Let $\mathcal{RH}$ stand for the corresponding def-Horn ruleset. The conclusion set of $\mathcal{RP}$ then coincides with the smallest (again, w.r.t. set inclusion) Herbrand model of $\mathcal{RH}$.

Next, we discuss this relationship. The def-LP and the def-Horn ruleset entail exactly the same set of facts. Every conclusion of the def-LP is also a conclusion of the def-Horn ruleset. Relative to the def-Horn ruleset, the def-LP is thus sound; moreover, it is complete for fact-form conclusions, i.e., for queries whose answers amount to conjunctions of facts. However, the def-LP is a mildly *weaker* version of the def-Horn ruleset, in the following sense. Every conclusion of the def-LP must have the form of a fact. By contrast, the entailments, i.e., conclusions, of the def-Horn ruleset are not restricted to be facts. E.g., suppose $\mathcal{RH}$ consists of the two rules

$kiteDay(Tues) \;\leftarrow\; sunny(Tues) \wedge windy(Tues)$

and

$sunny(Tues).$

Then it entails

$kiteDay(Tues) \;\leftarrow\; windy(Tues)$

(a non-unit derived clause) whereas $\mathcal{RP}$ does not. In practical applications, however, quite often only the fact-form conclusions are desired — e.g., an application might be interested above only in whether or not $kiteDay(Tues)$ is entailed. The def-LP has the virtue of conceptual and computational simplicity. To use an analogy, like a hard-boiled detective from a mid-century cop story, it says "give me the facts, ma'am, just the facts". Thinking in terms of expressive classes, we will view def-LP as an *expressive subset* of def-Horn— we will call it the expressive *f-subset*. def-LP is a mild weakening of def-Horn along the dimension of entailment power, permitting only fact-form conclusions — we will call this *f-weakening*.

In return for this f-weakening, def-LP has some quite attractive computational characteristics (as well as being expressively extensible in directions that FOL is not, as discussed earlier). For the propositional case of def-LP, exhaustive inferencing is $O(n)$ where $n = |\mathcal{RP}|$ — i.e., worst-case linear time [8]. For the general case with logical variables, the entire conclusion set of a def-LP $\mathcal{RP}$ can be computed in time $O(n^{v+1})$, when there is a constant bound v on the number of logical variables per rule (this restriction, which we will call *VB*, is typically met in practice). Inferencing in def-LP is thus tractable (worst-case polynomial time) given VB. In contrast, DLs are generally not tractable (typically ExpTime or even NExpTime complexity for key inference problems), and full FOL is not decidable.

4. MAPPING DL TO def-Horn

In this section we will discuss how DL languages (e.g., DAML+OIL) can be mapped to def-Horn, and vice versa.

4.1 Expressive Restrictions

[12][2] call this "general"; however, there are actually a number of frequently used extensions!

[13]in the LP literature, this conclusion set is often called its "model"

[14]There are several other proposed semantics for LP (e.g., the stable semantics) which differ for cases where negation-as-failure interacts complexly with recursive (cyclic) dependencies (through the rules themselves) among predicates/atoms. The well-founded semantics is the most conceptually popular LP semantics in the basic research community, is increasingly prevalent in both academic and commercial implementation, and has the virtue of preserving computational tractability. For definite LP, the stable semantics is equivalent to the well-founded semantics.

We will first discuss the expressive restrictions of DL and def-Horn as these will constrain the subset of DL and def-Horn for which a complete mapping can be defined.

DLs are decidable subsets of FOL where the decidability is due in large part to their having (a form of) the tree model property [22].[15] This property says that a DL class C has a model (an interpretation $\mathcal{I}$ in which $C^{\mathcal{I}}$ in non-empty) iff C has a tree-shaped model, i.e., one in which the interpretation of properties defines a tree shaped directed graph.

This requirement severely restricts the way variables and quantifiers can be used. In particular, quantifiers must be *relativized* via atomic formulae (as in the guarded fragment of FOL [11]), i.e., the quantified variable must occur in a property predicate along with the free variable (recall that DL classes correspond to formulae with one free variable). For example, the DL class $\exists P.C$ corresponds to the FOL formula $\exists y.(P(x,y) \wedge C(y))$, where the property predicate P acts as a guard. One obvious consequence of this restriction is that it is impossible to describe classes whose instances are related to another anonymous individual via different property paths. For example, it is impossible to assert that individuals who live and work at the same location are "HomeWorkers". This is easy with a Horn rule, e.g.:

$$\text{HomeWorker}(x) \ \leftarrow\ \text{work}(x,y) \wedge \text{live}(x,z) \wedge \text{loc}(y,w) \wedge \text{loc}(z,w)$$

Another restriction in DLs is that only unary and binary predicates can usually be captured.[16] This is a less onerous restriction, however, as techniques for reifying higher arity predicates are well known [16].

Definite Horn FOL requires that all variables are universally quantified (at the outer level of the rule), and restricts logical connectives in certain ways. One obvious consequence of the restriction on quantifiers is that it is impossible to assert the existence of individuals whose identity might not be known. For example, it is impossible to assert that all persons have a father (known or unknown). This is easy with a DL axiom, e.g.:

$$\text{Person} \sqsubseteq \exists \text{father}.\top.$$

No negation may appear within the body of a rule, nor within the head. No existentials may appear within the head. Thus it is impossible to assert, e.g., that all persons are either men or women (but not both). This would also be easy using DL axioms, e.g.:

$$\begin{array}{rcl}\text{Person} & \sqsubseteq & \text{Man} \sqcup \text{Woman} \\ \text{Man} & \sqsubseteq & \neg\text{Woman}.\end{array}$$

The Datalog restriction of def-Horn is not an issue for mapping DL into it, since DL also has the Datalog restriction. Finally, the equality-free restriction of def-Horn is a significant restriction in that it prevents representing (partial-)functionality of a property and also prevents representing maximum cardinality. The prohibition against existentials in the head prevents representing minimum cardinality.

4.2 Mapping Statements

In this section, we show how (some of) the *statements* (axioms) of DL and DL based languages (such as DAML+OIL and OWL) correspond to def-Horn statements (rules).

[15] Expressive features such as transitive properties and the `oneOf` constructor compromise the tree model property to some extent, e.g., transitive properties can cause "short-cuts" down branches of the tree.

[16] This is not an inherent restriction, and n-ary DLs are known, e.g., $\mathcal{DLR}$ [5].

4.2.1 RDFS Statements

RDFS provides a subset of the DL statements described in Section 3.1: subclass, subproperty, range, and domain statements (which in a DL setting are often called Tbox axioms); and asserted class-instance (type) and instance-property-instance relationships (which in a DL setting are often called Abox axioms).

As we saw in Section 3.1, a DL inclusion axiom corresponds to an FOL implication. This leads to a straightforward mapping from class and property inclusion axioms to def-Horn rules as follows:

> $C \sqsubseteq D$, i.e., class C is subclass of class D, maps to:
> $D(x) \ \leftarrow\ C(x)$
>
> $Q \sqsubseteq P$, i.e., Q is a subproperty of P, maps to:
> $P(x,y) \ \leftarrow\ Q(x,y)$

As shown in Figure 3, RDFS range and domain statements correspond to DL axioms of the form $\top \sqsubseteq \forall P.C$ (range of P is C) and $\top \sqsubseteq \forall P^{-}.C$ (domain of P is C). From Figure 5, we can see that these are equivalent to the FOL sentences $\forall x.\ true \rightarrow (\forall y.\ P(x,y) \rightarrow C(y))$ and $\forall x.\ true \rightarrow (\forall y.\ P(y,x) \rightarrow C(y))$, which can be simplified to $\forall x,y.\ P(x,y) \rightarrow C(y)$ and $\forall x,y.\ P(y,x) \rightarrow C(y)$ respectively. These FOL sentences are already in def-Horn form, which gives us the following mappings for range and domain:

> $\top \sqsubseteq \forall P.C$, i.e., the range of property P is class C, maps to:
> $C(y) \ \leftarrow\ P(x,y)$
>
> $\top \sqsubseteq \forall P^{-}.C$, i.e., the domain of property P is class C, maps to:
> $C(y) \ \leftarrow\ P(y,x)$

Finally, asserted class-instance (type) and instance-property-instance relationships, which correspond to DL axioms of the form $a : C$ and $\langle a,b \rangle : P$ respectively (Abox axioms), are equivalent to FOL sentences of the form $C(a)$ and $P(a,b)$, where a and b are constants. These are already in def-Horn form: they are simply rules with empty bodies (which are normally omitted):

> $a : C$, i.e., the individual a is an instance of the class C, maps to:
> $C(a)$
>
> $\langle a,b \rangle : P$, i.e., the individual a is related to the individual b via the property P, maps to:
> $P(a,b)$

Note that in these rules a and b are ground (constants).

4.2.2 DAML+OIL statements

DAML+OIL extends RDF with additional statements about classes and properties (Tbox axioms). In particular, it adds explicit statements about class, property and individual equality and inequality, as well as statements asserting property inverses, transitivity, functionality (unique) and inverse functionality (unambiguous).

As discussed in Section 3.1, class and property equivalence axioms can be replaced with a symmetrical pair of inclusion axioms, so they can be mapped to a symmetrical pair of def-Horn rules as follows:

> $C \equiv D$, i.e., the class C is equivalent to (has the same extension as) the class D, maps to:
> $$\begin{array}{rcl}D(x) & \leftarrow & C(x) \\ C(x) & \leftarrow & D(x)\end{array}$$
>
> $P \equiv Q$, i.e., the property P is equivalent to (has the same extension as) the property Q, maps to:
> $$\begin{array}{rcl}Q(x,y) & \leftarrow & P(x,y) \\ P(x,y) & \leftarrow & Q(x,y)\end{array}$$

As we saw in Section 3.1, the semantics of inverse axioms of the form $P \equiv Q^-$ are captured by FOL sentences of the form $\forall x, y.P(x, y) \iff Q(x, y)$, and the semantics of transitivity axioms of the form $P^+ \sqsubseteq P$ are captured by FOL sentences of the form $\forall x, y, z.P(x, y) \wedge P(y, z) \rightarrow P(x, z)$. This leads to a direct mapping into def-Horn as follows:

$P \equiv Q^-$, i.e., the property Q is the inverse of the property P, maps to:

$$\begin{array}{rcl} Q(y,x) & \leftarrow & P(x,y) \\ P(x,y) & \leftarrow & Q(y,x) \end{array}$$

$P^+ \sqsubseteq P$, i.e., the property P is transitive, maps to:

$$P(x,z) \leftarrow P(x,y) \wedge P(y,z)$$

As we saw in Section 3.1, DL axioms asserting the functionality of properties correspond to FOL sentences with equality. E.g., a DL axiom $\top \sqsubseteq \leqslant 1\, P$ (P is a functional property) corresponds to the FOL sentence $\forall x, y, z.P(x, y) \wedge P(x, z) \rightarrow y = z$.[17] This kind of axiom cannot be dealt with in our current framework (see Section 4.1) as it would require def-Horn rules with equality in the head, i.e., rules of the form $(x = y) \leftarrow P(x, y) \wedge P(x, z)$.

4.3 Mapping Constructors

In the previous section we showed how DL axioms correspond with def-Horn rules, and how these can be used to make statements about classes and properties. In DLs, the classes appearing in such axioms need not be atomic, but can be complex compound expressions built up from atomic classes and properties using a variety of constructors. A great deal of the power of DLs derives from this feature, and in particular from the set of constructors provided.[18] In the following section we will show how these DL expressions correspond to expressions in the body of def-Horn rules.

In the following we will, as usual, use C, D to denote classes, P, Q to denote properties and n to denote an integer.

Conjunction (DL $\sqcap$)

A DL class can be formed by conjoining existing classes, e.g., $C \sqcap D$. From Figure 5 it can be seen that this corresponds to a conjunction of unary predicates. Conjunction can be directly expressed in the body of a def-Horn rule. E.g., when a conjunction occurs on the l.h.s. of a subclass axiom, it simply becomes conjunction in the body of the corresponding rule

$$C_1 \sqcap C_2 \sqsubseteq D \quad \equiv \quad D(x) \leftarrow C_1(x) \wedge C_2(x)$$

Similarly, when a conjunction occurs on the r.h.s. of a subclass axiom, it becomes conjunction in the head of the corresponding rule:

$$C \sqsubseteq D_1 \sqcap D_2 \quad \equiv \quad D_1(x) \wedge D_2(x) \leftarrow C(x),$$

This is then easily transformed (via the Lloyd-Topor transformations [18]) into a pair of def-Horn rules:

$$\begin{array}{rcl} D_1(x) & \leftarrow & C(x) \\ D_2(x) & \leftarrow & C(x) \end{array}$$

Disjunction (DL $\sqcup$)

A DL class can be formed from a disjunction of existing classes, e.g., $C \sqcup D$. From Figure 5 it can be seen that this corresponds to a disjunction of unary predicates. When a disjunction occurs on the l.h.s. of a subclass axiom it simply becomes disjunction in the body of the corresponding rule:

$$C_1 \sqcup C_2 \sqsubseteq D \quad \equiv \quad D(x) \leftarrow C_1(x) \vee C_2(x)$$

This is easily transformed (again by Lloyd-Topor) into a pair of def-Horn rules:

$$\begin{array}{rcl} D(x) & \leftarrow & C_1(x) \\ D(x) & \leftarrow & C_2(x) \end{array}$$

When a disjunction occurs on the r.h.s. of a subclass axiom it becomes a disjunction in the head of the corresponding rule, and this cannot be handled within the def-Horn framework.

Universal Restriction (DL $\forall$)

In a DL the universal quantifier can only be used in *restrictions*—expressions of the form $\forall P.C$ (see Section 4.1). This is equivalent to an FOL clause of the form $\forall y.P(x, y) \rightarrow C(y)$ (see Figure 5). P must be a single primitive property, but C may be a compound expression. Therefore, when a universal restriction occurs on the r.h.s. of a subclass axiom it becomes an implication in the head of the corresponding rule:

$$C \sqsubseteq \forall P.D \quad \equiv \quad (D(y) \leftarrow P(x,y)) \leftarrow C(x),$$

which is easily transformed into the standard def-Horn rule:

$$D(y) \leftarrow C(x) \wedge P(x,y).$$

When a universal restriction occurs on the l.h.s. of a subclass axiom it becomes an implication in the body of the corresponding rule. This cannot, in general, be mapped into def-Horn as it would require negation in a rule body.

Existential Restriction (DL $\exists$)

In a DL, the existential quantifier (like the universal quantifier) can only be used in restrictions of the form $\exists P.C$. This is equivalent to an FOL clause of the form $\exists y.P(x, y) \wedge C(y)$ (see Figure 5). P must be a single primitive property, but C may be a compound expression.

When an existential restriction occurs on the l.h.s. of a subclass axiom, it becomes a conjunction in the body of a standard def-Horn rule:

$$\exists P.C \sqsubseteq D \quad \equiv \quad D(x) \leftarrow P(x,y) \wedge C(y).$$

When an existential restriction occurs on the r.h.s. of a subclass axiom, it becomes a conjunction in the head of the corresponding rule, with a variable that is existentially quantified. This cannot be handled within the def-Horn framework.

Negation and Cardinality Restrictions (DL $\neg$, $\geqslant$ and $\leqslant$)

These constructors cannot, in general, be mapped into def-Horn. The case of negation is obvious as negation is not allowed in either the head or body of a def-Horn rule. As can be seen in Figure 5, cardinality restrictions correspond to assertions of variable equality and inequality in FOL, and this is again outside of the def-Horn framework.

In some cases, however, it would be possible to simplify the DL expression using the usual rewriting tautologies of FOL in order to eliminate the offending operator(s). For example, negation can always be pushed inwards by using a combination of De Morgan's laws and equivalences such as $\neg\exists P.C \equiv \forall P.\neg C$ and $\neg \geqslant n\, P \equiv\ \leqslant (n-1)\, P$ [1]. Further simplifications are also possible, e.g., using the equivalences $C \sqcup \neg C \equiv \top$, and $\forall P.\top \equiv \top$. For the sake of simplicity, however, we will assume that DL expressions are in a canonical form where all relevant simplifications have been carried out.

[17] Note that, technically, this is partial-functionality as for any given x there is no requirement that there exist a y such that $P(x, y)$.

[18] Note that this feature is not supported in the RDFS subset of DLs.

4.4 Defining DHL via a Recursive Mapping from DL to def-Horn

As we saw in Section 4.3, some DL constructors (conjunction and universal restriction) can be mapped to the heads of rules whenever they occur on the r.h.s. of an inclusion axiom, while some DL constructors (conjunction, disjunction and existential restriction) can be mapped to the bodies of rules whenever they occur on the l.h.s. of an inclusion axiom. This naturally leads to the definition of two DL languages, classes from which can be mapped into the head or body of LP rules; we will refer to these two languages as $\mathcal{L}_h$ and $\mathcal{L}_b$ respectively.

The syntax of the two languages is defined as follows. In both languages an atomic name A is a class, and if C and D are classes, then $C \sqcap D$ is also a class. In $\mathcal{L}_h$, if C is a class and R is a property, then $\forall R.C$ is also a class, while in $\mathcal{L}_b$, if D, C are classes and R is a property, then $C \sqcup D$ and $\exists R.C$ are also classes.

Using the mappings from Section 4.3, we can now define a recursive mapping function $\mathcal{T}$ which takes a DL axiom of the form $C \sqsubseteq D$, where C is an $\mathcal{L}_b$-class and D is an $\mathcal{L}_h$-class, and maps it into an LP rule of the form $A \leftarrow B$. The mapping is defined as follows:

$$\begin{array}{lll}
\mathcal{T}(C \sqsubseteq D) & \longrightarrow & Th(D,y) \ \leftarrow \ Tb(C,y) \\
Th(A,x) & \longrightarrow & A(x) \\
Th((C \sqcap D),x) & \longrightarrow & Th(C,x) \wedge Th(D,x) \\
Th((\forall R.C),x) & \longrightarrow & Th(C,y) \ \leftarrow \ R(x,y) \\
Tb(A,x) & \longrightarrow & A(x) \\
Tb((C \sqcap D),x) & \longrightarrow & Tb(C,x) \wedge Tb(D,x) \\
Tb((C \sqcup D),x) & \longrightarrow & Tb(C,x) \vee Tb(D,x) \\
Tb((\exists R.C),x) & \longrightarrow & R(x,y) \wedge Tb(C,y)
\end{array}$$

where A is an atomic class name, C and D are classes, R is a property and x, y are variables, with y being a "fresh" variable, i.e., one that has not previously been used.

As we saw in Section 4.3, rules of the form $(H \wedge H') \leftarrow B$ are rewritten as two rules $H \leftarrow B$ and $H' \leftarrow B$; rules of the form $(H \leftarrow H') \leftarrow B$ are rewritten as $H \leftarrow (B \wedge H')$; and rules of the form $H \leftarrow (B \vee B')$ are rewritten as two rules $H \leftarrow B$ and $H \leftarrow B'$.

For example, $\mathcal{T}$ would map the DL axiom

$$A \sqcap \exists R.C \sqsubseteq B \sqcap \forall P.D$$

into the LP rule

$$B(x) \wedge (D(z) {\leftarrow} P(x,z)) \ \leftarrow \ A(x) \wedge R(x,y) \wedge C(x)$$

which is rewritten as the pair of rules

$$\begin{array}{lll}
B(x) & \leftarrow & A(x) \wedge R(x,y) \wedge C(x) \\
D(z) & \leftarrow & A(x) \wedge R(x,y) \wedge C(x) \wedge P(x,z).
\end{array}$$

We call $\mathcal{L}$ the intersection of $\mathcal{L}_h$ and $\mathcal{L}_b$, i.e., the language where an atomic name A is a class, and if C and D are classes, then $C \sqcap D$ is also a class. We then extend $\mathcal{T}$ to deal with axioms of the form $C \equiv D$, where C and D are both $\mathcal{L}$-classes:

$$\mathcal{T}(C \equiv D) \quad \longrightarrow \quad \begin{cases} \mathcal{T}(C \sqsubseteq D) \\ \mathcal{T}(D \sqsubseteq C) \end{cases}$$

As we saw in Section 4.2.1, range and domain axioms $\top \sqsubseteq \forall P.D$ and $\top \sqsubseteq \forall P^-.D$ are mapped into def-Horn rules of the form $D(y) \leftarrow P(x,y)$ and $D(x) \leftarrow P(x,y)$ respectively. Moreover, class-instance and instance-property-instance axioms $a : D$ and $\langle a,b \rangle : P$ are mapped into def-Horn facts (i.e., rules with empty bodies) of the form $D(a)$ and $P(a,b)$ respectively. We therefore extend $\mathcal{T}$ to deal with these axioms in the case that D is an $\mathcal{L}_h$-class:

$$\begin{array}{lll}
\mathcal{T}(\top \sqsubseteq \forall P.D) & \longrightarrow & Th(D,y) \ \leftarrow \ P(x,y) \\
\mathcal{T}(\top \sqsubseteq \forall P^-.D) & \longrightarrow & Th(D,x) \ \leftarrow \ P(x,y) \\
\mathcal{T}(a : D) & \longrightarrow & Th(D,a) \\
\mathcal{T}(\langle a,b \rangle : P) & \longrightarrow & P(a,b)
\end{array}$$

where x, y are variables and a, b are constants.

Finally, we extend $\mathcal{T}$ to deal with the property axioms discussed in Section 4.2:

$$\begin{array}{lll}
\mathcal{T}(P \sqsubseteq Q) & \longrightarrow & Q(x,y) \ \leftarrow \ P(x,y) \\
\mathcal{T}(P \equiv Q) & \longrightarrow & \begin{cases} Q(x,y) \ \leftarrow \ P(x,y) \\ P(x,y) \ \leftarrow \ Q(x,y) \end{cases} \\
\mathcal{T}(P \equiv Q^-) & \longrightarrow & \begin{cases} Q(x,y) \ \leftarrow \ P(y,x) \\ P(y,x) \ \leftarrow \ Q(x,y) \end{cases} \\
\mathcal{T}(P^+ \sqsubseteq P) & \longrightarrow & P(x,z) \ \leftarrow \ P(x,y) \wedge P(y,z)
\end{array}$$

Definition 1 (Description Horn Logic) *A* Description Horn Logic (DHL) *ontology is a set of DHL axioms of the form $C \sqsubseteq D$, $A \equiv B$, $\top \sqsubseteq \forall P.D$, $\top \sqsubseteq \forall P^-.D$, $P \sqsubseteq Q$, $P \equiv Q$, $P \equiv Q^-$, $P^+ \sqsubseteq P$, $a : D$ and $\langle a,b \rangle : P$, where C is an $\mathcal{L}_b$-class, D is an $\mathcal{L}_h$-class, A, B are $\mathcal{L}$-classes, P, Q are properties and a, b are individuals.*

Using the relationships of (full) DL to FOL discussed in Section 3.1, especially Figure 5, it is straightforward to show the following.

Theorem 1 (Translation Semantics) *The mapping $\mathcal{T}$ preserves semantic equivalence. Let $\mathcal{K}$ be a DHL ontology and $\mathcal{H}$ be the def-Horn ruleset that results from applying the mapping $\mathcal{T}$ to all the axioms in $\mathcal{K}$. Then $\mathcal{H}$ is logically equivalent to $\mathcal{K}$ w.r.t. the semantics of FOL — $\mathcal{H}$ has the same set of models and entailed conclusions as $\mathcal{K}$.*

DHL can, therefore, be viewed alternatively and precisely as an expressive fragment of def-Horn— i.e., as the *range* of $\mathcal{T}$(DHL).

4.5 Expressive Power of DHL

Although the asymmetry of DHL (w.r.t. classes on different sides of axioms) makes it rather unusual by DL standards, it is easy to see that it includes (the DAML+OIL subset of) RDFS, as well as that part of DAML+OIL which corresponds to a simple frame language.

As far as RDFS is concerned, we saw in Section 4.2.1 that RDFS statements are equivalent to DL axioms of the form $C \sqsubseteq D$, $\top \sqsubseteq \forall P.C$, $\top \sqsubseteq \forall P^-.C$, $P \sqsubseteq Q$, $a : D$ and $\langle a,b \rangle : P$, where C, D are classes, P, Q are properties and a, b are individuals. Given that all RDFS classes are $\mathcal{L}$-classes (they are atomic class names), a set of DL axioms corresponding to RDFS statements would clearly satisfy the above definition of a DHL ontology.

DHL also includes the subset of DAML+OIL corresponding to simple frame language axioms, i.e., axioms defining a primitive hierarchy of classes, where each class is defined by a frame. A frame specifies the set of subsuming classes and a set of slot constraints. This corresponds very neatly to a set of DL axioms of the form $A \sqsubseteq \mathcal{L}_h$.

Moreover, DHL supports the extension of this language to include equivalence of conjunctions of atomic classes, and axioms corresponding to DAML+OIL transitive property, and inverse property statements.

5. DEFINING DLP

Definition 2 (Description Logic Programs) *We say that a def-LP $\mathcal{RP}$ is a* Description Logic Program (DLP) *when it is the LP-correspondent of some DHL ruleset $\mathcal{RH}$.*

A DLP is directly defined as the LP-correspondent of a def-Horn ruleset that results from applying the mapping $\mathcal{T}$. Semantically, a DLP is thus the f-weakening of that DHL ruleset (recall subsection 3.2). The DLP expressive class is thus the expressive f-subset of DHL. By Theorem 1, DLP can, therefore, be viewed alternatively and precisely as an expressive subset of DL, not just of def-Horn.

In summary, expressively DLP is contained in DHL which in turn is contained in the expressive intersection of DL and Horn.

6. MORE ABOUT TRANSLATING

As our discussion of expressive relationships has made clear, there is a bi-directional semantic equivalence of (1) the DHL fragment of DL and (2) the DHL fragment of def-Horn. Likewise, there is a bi-directional semantic equivalence of the DLP fragment of DL and the DLP fragment of def-Horn. So far, however, we have mostly concentrated on only one direction of *syntactic* mapping: from DL syntax to def-Horn syntax (and to the corresponding def-LP), rather than from def-Horn (or def-LP) to DL. Next, we elucidate our reasons for this emphasis.

First, a prime immediate goal for the Semantic Web is to enable rules (in LP / Horn) on top of ontologies (in DL) — more than vice versa to enable DL ontologies on top of LP or Horn rules. Second, it is desirable to exploit the relatively numerous, mature, efficient, scalable algorithms and implementations (i.e., engines) already available for LP inferencing so as to perform some fragment of DL inferencing — more than vice versa to perform LP via the fewer available DL engines, which are designed to handle more expressive languages (than DLP) and may not be optimized for DLP ontologies. Third, as compared to def-Horn, DL has a relatively detailed set of quite specific syntactic expressive constructs; it was easier to go through these one by one to define a translation mapping than to do so in the reverse direction where one has to invent more structure/forms.

We do not have space here to give detailed algorithms and computational complexity analyses of the syntactic translations. We will limit ourselves to some relatively high-level observations; these are straightforward to show. The $\mathcal{T}$ mapping, from DL syntax to def-Horn/def-LP syntax, corresponds immediately to an algorithm whose computational complexity is tractable. This mapping is invertible (e.g., in the usual manner of parsers) from def-Horn/def-LP syntax to DL syntax, again, tractably.

7. INFERENCING

As discussed in the previous section, one of the prime goals of this work is to enable some fragment of DL inferencing to be performed by LP engines. In this section we will discuss the kinds of inference typically of interest in DL and LP, and how they can be represented in each other, i.e., in LP and DL respectively. Although the emphasis is on performing DL inferencing, via our mapping translation, using an LP reasoning engine, the reverse mapping can be used in order to perform LP inferencing using a DL reasoning engine. In particular, we will show how inferencing in (the DHL fragment of) DL can be reduced, via our translation, to inferencing in LP; and how vice versa, inferencing in (the DLP fragment of) LP can be reduced to inferencing in DL.

In a DL reasoning system, several different kinds of query are typically supported w.r.t. a knowledge base $\mathcal{K}$. These include queries about classes:

1. class-instance membership queries: given a class C,
   (a) ground: determine whether a given individual a is an instance of C;
   (b) open: determine all the individuals in $\mathcal{K}$ that are instances of C;
   (c) "all-classes": given an individual a, determine all the (named) classes in $\mathcal{K}$ that a is an instance of;
2. class subsumption queries: i.e., given classes C and D, determine if C is a subclass of D w.r.t. $\mathcal{K}$;
3. class hierarchy queries: i.e., given a class C return all/most-specific (named) superclasses of C in $\mathcal{K}$ and/or all/most-general (named) subclasses of C in $\mathcal{K}$;
4. class satisfiability queries, i.e., given a class C, determine if C is satisfiable (consistent) w.r.t. $\mathcal{K}$.

In addition, there are similar queries about properties: property-instance membership, property subsumption, property hierarchy, and property satisfiability. We will call $\mathcal{QDL}$ the language defined by the above kinds of DL queries.

In LP reasoning engines, there is one basic kind of query supported w.r.t. a ruleset $\mathcal{R}$: atom queries. These include:

1. ground: determine whether a ground atom A is entailed;
2. open (ground is actually a special case of this): determine, given an atom A (in which variables may appear), all the tuples of variable bindings (substitutions) for which the atom is entailed.

We call $\mathcal{QLP}$ the language defined by the above kinds of LP queries.

Next, we discuss how to reduce $\mathcal{QDL}$ querying in (the DHL fragment of) DL to $\mathcal{QLP}$ querying in (the DLP fragment of) LP using the mapping $\mathcal{T}$. We will assume that $\mathcal{R}$ is a ruleset derived from a DL knowledge base $\mathcal{K}$ via $\mathcal{T}$, and that all $\mathcal{QDL}$ queries are w.r.t. $\mathcal{K}$.

$\mathcal{QLP}$ (ground or open) atom queries can be used to answer $\mathcal{QDL}$ (ground or open) class-instance membership queries when the class is an $\mathcal{L}_h$-class, i.e., a is an instance of C iff $\mathcal{R}$ entails $\mathcal{T}(a : C)$. When C is an atomic class name, the mapping leads directly to a $\mathcal{QLP}$ atom query. When C is a conjunction, the result is a conjunction of $\mathcal{QLP}$ atom queries, i.e., a is an instance of $C \sqcap D$ iff $\mathcal{R}$ entails $\mathcal{T}(a : C)$ *and* $\mathcal{R}$ entails $\mathcal{T}(a : D)$. When C is a universal restriction, the mapping $\mathcal{T}(a : \forall P.C)$ gives $\mathcal{T}(C, y) \leftarrow P(a, y)$. This can be transformed into a $\mathcal{QLP}$ atom query using a simple kind of skolemization, i.e., y is replaced with a constant b, where b is new in $\mathcal{R}$, and we have a is an instance of $\forall P.C$ iff $\mathcal{R} \cup \{P(a, b)\}$ entails $\mathcal{T}(b : C)$.

The case of property-instance membership queries is trivial as all properties are atomic: $\langle a, b\rangle$ is an instance of P iff $\mathcal{R}$ entails $P(a, b)$.

Complete information about class-instance relationships, to answer open or "all-classes" class-instance queries, can then be obtained via class-instance queries about all possible combinations of individuals and classes in $\mathcal{K}$.[19] (Note that the set of named individuals and classes is known, and its size is worst-case linear in the size of the knowledge/rule base.)

For $\mathcal{L}_h$-classes, $\mathcal{QDL}$ class subsumption queries can be reduced to $\mathcal{QLP}$ using a similar technique to class-instance membership queries, i.e., C is a subclass of D iff $\mathcal{R} \cup \{\mathcal{T}(a : C)\}$ entails $\mathcal{T}(a : D)$, for a new in $\mathcal{R}$. For $\mathcal{QDL}$ property subsumption queries, P is a subproperty of Q iff $\mathcal{R} \cup P(a, b)$ entails $Q(a, b)$, for a, b new in $\mathcal{R}$.

[19]More efficient algorithms would no doubt be used in practice.

Complete information about the class hierarchy can be obtained by computing the partial ordering of classes in $\mathcal{K}$ based on the subsumption relationship.

In the DHL (and DLP) fragment, determining class/property satisfiability is a non-issue as, with the expressive power at our disposal in def-Horn, it is impossible to make a class or a property unsatisfiable.

Now let us consider the reverse direction from $\mathcal{QLP}$ to $\mathcal{QDL}$. In the DLP fragment of LP, every predicate is either unary or binary. Every atom query can thus be viewed as about either a named class or a property. Also, generally in LP, any open atom query is formally reducible to a set of ground atom queries — one for each of its instantiations. Thus $\mathcal{QLP}$ is reducible to class-instance and property-instance membership queries in DL.

To recap, we have shown the following.

Theorem 2 (Inferencing Inter-operability) *For $\mathcal{L}_h$-classes, $\mathcal{QDL}$ querying in (the DHL fragment of) DL is reducible to $\mathcal{QLP}$ querying in (the DLP fragment of) LP, and vice versa.*

8. DISCUSSION

In this paper we have shown how to interoperate, semantically and inferentially, between the leading Semantic Web approaches to rules (RuleML Logic Programs) and ontologies (OWL/DAML+OIL Description Logic). We have begun by studying two new KRs, Description Logic Programs (DLP), which is defined by the expressive intersection of the two approaches, and the closely related Description Horn Logic (DHL).

We have shown that DLP (or DHL) can capture a significant fragment of DAML+OIL, including the whole of the DAML+OIL subset of RDFS, simple frame axioms and more expressive property axioms. Many of the ontologies in the DAML ontology library are inside this fragment of DAML+OIL. An immediate result of this work is that LP engines could be used for reasoning with these ontologies and for reasoning with (possibly very large numbers of) facts, such as web page annotations, that use vocabulary from these ontologies.

This work represents only a first step in realising a more complete interoperability between rules and ontologies, and the layering of rules on top of ontology languages in the Semantic Web “stack”. We believe, however, that our study of the expressive intersection will provide a firm foundation for future investigations of more expressive languages up to and including the expressive union of rules and ontologies.

Acknowledgements

Thanks to Tim Berners-Lee, Harold Boley, Dan Connolly, Michael Dean, Stefan Decker, Richard Fikes, Patrick Hayes, Jim Hendler, Deborah McGuinness, Jos De Roo, Peter Patel-Schneider, Raphael Volz, and other members of the DAML+OIL Joint Committee for helpful comments and discussions.

9. REFERENCES

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2002.

[2] C. Baral and M. Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19/20:73–148, 1994.

[3] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.

[4] D. Brinkley and R. V. Guha. Resource description framework (RDF) schema specification 1.0. W3C Candidate Recommentation, Mar. 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.

[5] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, pages 149–158, 1998.

[6] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language 1.0 reference, July 2002. http://www.w3.org/TR/owl-ref/.

[7] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of KR'91*, pages 151–162, 1991.

[8] W. Dowling and J. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 3:267–284, 1984.

[9] A. Firat, S. Madnick, and B. N. Grosof. Financial information integration in the presence of equational ontological conflicts. In *Proc. of WITS-2002*, 2002.

[10] A. Firat, S. Madnick, and B. N. Grosof. Knowledge integration to overcome ontological heterogeneity: Challenges from financial information systems. In *Proc. of ICIS-2002*, To appear.

[11] E. Grädel. On the restraining power of guards. *J. of Symbolic Logic*, 64:1719–1742, 1999.

[12] B. N. Grosof. Representing e-business rules for the semantic web: Situated courteous logic programs in ruleml. In *Proc. of WITS '01*, 2001.

[13] B. N. Grosof, Y. Labrou, and H. Y. Chan. A declarative approach to business rules in contracts: Courteous logic programs in xml. In *Proc. of EC-99*, 1999.

[14] B. N. Grosof and T. C. Poon. Representing agent contracts with exceptions using xml rules, ontologies, and process descriptions. In *Proc. of International Workshop on Rule Markup Languages for Business Rules on the Semantic Web*, 2002.

[15] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of IJCAI 2001*, pages 199–204, 2001.

[16] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide query containment under constraints using a description logic. In *Proc. of LPAR'2000*, 2000.

[17] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, pages 161–180, 1999.

[18] J. W. Lloyd. *Foundations of logic programming (second, extended edition)*. Springer series in symbolic computation. Springer-Verlag, New York, 1987.

[19] D. M. Reeves, M. P. Wellman, and B. N. Grosof. Automated negotiation from declarative contract descriptions. *Computational Intelligence*, 18(4), 2002.

[20] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI'91*, pages 466–471, 1991.

[21] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[22] M. Y. Vardi. Why is modal logic so robustly decidable? In N. Immerman and P. Kolaitis, editors, *Descriptive Complexity and Finite Models*. American Mathematical Society, 1997.