DRAFT Working Paper – PLEASE DO NOT REDISTRIBUTE
Warning: This is still a pretty early and raw draft!

Description Logic Programs: Combining Logic Programs with Description Logic

Benjamin N. Grosof
MIT Sloan School of Management
Room E53-317, 50 Memorial Drive, Cambridge, MA 02142, USA
bgrosof@mit.edu ; http://www.mit.edu/~bgrosof/

Ian Horrocks
University of Manchester
Department of Computer Science
Oxford Road, Manchester, M13 9PL, UK
horrocks@cs.man.ac.uk ; http://www.cs.man.ac.uk/~horrocks/

Version 0.7-brief – Oct. 17, 2002

Abstract

We discuss how and why to combine rules with ontologies for the Semantic Web. As an ontology language, we focus on DAML+OIL; this is based on the particular logical knowledge representation (KR) formalism known as Description Logic (DL), and is the point of departure for the newly-formed WebOnt W3C Working Group. As a rule language, we focus on RuleML; this is based on the particular KR formalism known as logic programs (LP), and is the point of departure for the leading current standardization approach to rules for the Semantic Web.

1 Summary

We discuss how and why to combine rules with ontologies for the Semantic Web. As an ontology language, we focus on DAML+OIL; this is based on the particular logical knowledge representation (KR) known as Description Logic (DL), and is the point of departure for the newly-formed WebOnt W3C Working Group. As a rule language,

we focus on RuleML; this is based on the particular logical knowledge representation known as logic programs (LP), and is the point of departure for the leading current standardization approach to rules for the Semantic Web.

2 Preliminaries

2.1 Description logic

DAML+OIL is based on (an extension of) the $\mathcal{SHIQ}$ DL. $\mathcal{SHIQ}$ is built over a signature of distinct sets of concept ($\mathcal{CN}$), role ($\mathcal{RN}$) and individual ($\mathcal{O}$) names. In addition, we distinguish two non-overlapping subsets of $\mathcal{RN}$ ($\mathcal{TRN}$ and $\mathcal{FRN}$) which denote the transitive and the functional roles. The set of all $\mathcal{SHIQ}$ roles is equal to the set of role names $\mathcal{RN}$ union the set of the inverse roles $\{R^- \mid P \in \mathcal{RN}\}$. The set of all $\mathcal{SHIQ}$ concepts is the smallest set such that every concept name in $\mathcal{CN}$ and the symbols $\top$, $\bot$ are concepts, and if C,D are concepts, R is a role, and n an integer, then $\neg C$, $(C \sqcap D)$, $(C \sqcup D)$, $(\forall R.C)$, $(\exists R.C)$, $\geqslant n\, R.C$, and $\leqslant n\, R.C$ are concepts.

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 concepts into subsets of $\Delta^{\mathcal{I}}$, individual names into elements of $\Delta^{\mathcal{I}}$, and role names into subsets of the cartesian product of $\Delta^{\mathcal{I}}$ ($\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$). Concept names are interpreted as subsets of $\Delta^{\mathcal{I}}$, while complex expressions are interpreted according to the following equations (see [**?**])

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \quad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \quad\;\; (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
&\qquad\qquad \neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y (x,y) \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \\
(\geqslant n\, R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\} \\
(\leqslant n\, R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}
\end{aligned}$$

A role 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 and functional restrictions on role names; i.e., for any $R \in \mathcal{TRN}$ if $(x,y) \in R^{\mathcal{I}}$ and $(y,z) \in R^{\mathcal{I}}$, then $(x,z) \in R^{\mathcal{I}}$, and for any $F \in \mathcal{FRN}$ if $(x,y) \in F^{\mathcal{I}}$ and $(x,z) \in F^{\mathcal{I}}$, then $y = z$.

3 Mappings of DL Statements to LP

In this section, we give mappings of DL *statements* to LP.

Preamble:
DL class: is represented in mon-LP as a predicate with arity 1.
DL property: is represented in mon-LP as a predicate with arity 2. "?" prefix indicates a logical variable, in mon-LP syntax below.

Next we give the mapping of several kinds of DL statements (axioms) into LP, for which the mapping is relatively simple and straightforward.

The first four are axioms concerning classes and properties: in the DL setting these are often called Tbox axioms.

1. $C \sqsubseteq D$, i.e., class C is SUBCLASS of class D. This maps to:
   $D(?X) \leftarrow C(?X)$.

2. $dom(P) : C$, i.e., DOMAIN of property P is class C. This maps to:
   $C(?X) \leftarrow P(?X, ?Y)$.

3. $range(P) : D$, i.e., RANGE of property P is class D. This maps to:
   $D(?Y) \leftarrow P(?X, ?Y)$.

4. $Q \sqsubseteq P$, i.e., Q is a SUB-PROPERTY of P. This maps to:
   $P(?X, ?Y) \leftarrow Q(?X, ?Y)$.

The next two kinds of axiom are statements of ground facts concerning individual and pairs of individuals: in the DL setting these are often called Abox axioms.

An Abox consists of a set of axioms of the form $a{:}C$ and $\langle a, b\rangle{:}R$, where a, b are individuals, C is a class, and R is a role. $a{:}C$ asserts that a is an instance of class C, and $\langle a, b\rangle{:}R$ asserts that $\langle a, b\rangle$ is an instance of R.

Each Abox axiom is mapped to an LP ground fact.

1. a is an instance of class C (where a is ground):
   $C(a)$.

2. (a, b) is an instance of property P (where a,b are ground):
   $P(a, b)$.

The above six kinds of statements are essentially **the RDFS subset of DL**.

Next, we give three more kinds of Tbox statements.

1. $trans(P)$, i.e., property P is TRANSITIVE. This maps to:
   $P(?X, ?Z) \leftarrow P(?X, ?Y) \wedge P(?Y, ?Z)$.

2. $symm(P)$, i.e., property P is SYMMETRIC. This maps to:
   $P(?Y, ?X) \leftarrow P(?X, ?Y)$.

3. $fctnal(P)$, i.e., property P is FUNCTIONAL.

   The semantics of this (in English) is:
   "For every $?X$, there *exists at most one* $?Y$ such that $P(?X, ?Y)$."

   In FOL, we can represent the (partial-)functionality condition as:
   $\forall ?X.\ \neg\exists ?Y, ?Z.\ P(?X, ?Y) \wedge P(?X, ?Z) \wedge (?Z \neq ?Y)$
   which is equivalent to:
   $\forall ?X, ?Y, ?Z.\ P(?X, ?Y) \wedge P(?X, ?Z)\ \supset\ (?Z = ?Y)$

   In LP, we can thus represent the partial-functionality condition as:
   $(?Z = ?Y)\ \leftarrow\ P(?X, ?Y) \wedge P(?X, ?Z).$
   Note that this requires the LP language to permit ***explicit equality*** to appear in the head. Note also that this in a sense ***requires modification of UNA*** in general, since it may imply that two distinctly named individuals/ground-terms are indeed equal. (UNA stands for the Unique Names Assumption/Axiom in LP.)

   Note that, technically, this is partial-functionality which differs from functionality in that it omits the existence of Y, i.e.:
   "For every $?X$, there exists at least one $?Y$ such that $P(?X, ?Y)$."
   which can be formulated in FOL as:
   $\forall ?X.\ \exists ?Y.\ P(?X, ?Y)$

   Intuitively, the reason why LP cannot represent (at least directly) this existence is that it cannot represent an existential quantifier in the head/consequent of a rule. Such a head existential is essentially disjunctive. That is, $\exists ?Y.\ P(?X, ?Y)$ can be viewed as a kind of possibly-infinitary disjunction $P(?X, a1) \vee P(?X, a2) \vee \ldots$ where $a1, a2, \ldots$ are the members of the Herbrand universe, i.e., constitute the domain of quantification.

   Remember that LP cannot represent disjunction in the head of a rule.

4 Mapping DL Constructors to LP

In the next section, we give mapping of DL constructors to LP.

The DL constructors can be used syntactically to define complex class expressions that may appear in place of classes in the various kinds of DL statements. **In the RDFS subset of DL, only simple classes** (not complex class expressions) can appear.

In the notation below:
C,D each stands for a class.
P,Q each stands for a property.
k stands for an integer.

1. $\sqcap$, i.e., DL conjunction.

   Outline:

   Usage is: $C \sqcap D$, which results in a class.

   Intuitively, this poses no problem to represent in LP, when the LP (via the usual Lloyd-Topor expressive extension) permits head conjunction and conjunction of expressions in body and head. Remember that we can rewrite the head conjunction in LP (e.g., a rule with two head conjuncts gets rewritten into two rules with the same body, each having one of the conjuncts as its head).

2. $\forall$, i.e., DL universal quantifier, which is relativized.

   This quantifier is relativized in the following sense. It's used only in expressions of the form:
   $\forall P.C$
   which means that the property P is always of type C. Here, P must be a single primitive property, but C may be compound.

   Intuitively, this is OK "in head" but not "in body". There is a need to rewrite in LP. E.g.,
   $D \sqsubseteq \forall P.C$
   gets rewritten in LP as:
   $(C(?Y) \leftarrow P(?X, ?Y)) \leftarrow D(?X)$
   and then further rewritten in LP as:
   $C(?Y) \leftarrow P(?X, ?Y) \wedge D(?X)$
   We need to restrict what kind of DL constructors may appear in C – they must be ones which are OK to appear "in head", e.g., $\sqcap$ and $\forall$.

3. $\sqcup$, i.e., DL disjunction.

   Usage is: $C \sqcup D$, which results in a class.

   Intuitively this is OK "in body" but not "in head". There is a need to rewrite in LP, in the usual manner for body disjunction there. E.g.,
   $C \sqcup D \sqsubseteq A$
   gets rewritten in LP as two rules:
   $A(?X) \leftarrow C(?X)$
   $A(?X) \leftarrow D(?X)$
   We need to restrict what kind of DL constructors may appear in C and D – they must be ones which are OK to appear "in body".

4. $\exists$, i.e., DL existential quantifier, which is relativized.

   This quantifier is relativized in the following sense. It's used only in expressions of the form:
   $\exists P.C$

which means that there is some value the property P that is of type C. Here, P must be a single primitive property, but C may be compound.

Intuitively, this is OK "in body" but not "in head". There is a need to rewrite in LP. E.g.,
$\exists P.C \sqsubseteq D$
gets rewritten in LP as:
$D(?X) \leftarrow P(?X, ?Y) \wedge C(?Y)$
We need to restrict what kind of DL constructors may appear in C – they must be ones which are OK to appear "in body".

5. $\neg$, i.e., DL negation.

   Usage: applies to classes.

   Intuitively: in (monotonic/definite) LP one cannot represent this – there essentially is no negation.

4.1 Recursive DL to LP Mapping Function

Section 4 naturally leads to the definition of two DL languages, concepts from which can be translated into the head on 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 can be defined as follows. In both languages an atomic name A is a concept, and if C and D are concepts, then $C \sqcap D$ is also a concept. In $\mathcal{L}_h$, if C is a concept and R is a role, then $\forall R.C$ is also a concept, while in $\mathcal{L}_b$, if D, C are concepts and R is a role, then $C \sqcup D$ and $\exists R.C$ are also concepts.

Using the translations from Section 4, we can now define a recursive mapping function T which takes a DL axiom of the form $C \sqsubseteq D$, where C is an $\mathcal{L}_b$-concept and D is an $\mathcal{L}_h$-concept, and translates it into an LP rule of the form $A \leftarrow B$. The translation is defined as follows:

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

where A is an atomic concept name, C and D are concepts, R is a role and $?x, ?y$ are variables, with y being a "fresh" variable, i.e., one that has not previously been used.

Note that rules of the form $(H \wedge H') \leftarrow B$ can be rewritten as two rules $H \leftarrow B$ and $H' \leftarrow B$; rules of the form $(H \leftarrow H') \leftarrow B$ can be rewritten

as $H \leftarrow (B \wedge H')$; and rules of the form $H \leftarrow (B \vee B')$ can be rewritten as two rules $H \leftarrow B$ and $H \leftarrow B'$.

For example, the above mapping would translate 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 can be simplified to the pair of rules

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

If $\mathcal{L}$ is the intersection of $\mathcal{L}_h$ and $\mathcal{L}_b$, i.e., the language where an atomic name A is a concept, and if C and D are concepts, then $C \sqcap D$ is a concept, then we can also use T translate axioms of the form $C \equiv D$, where C and D are both $\mathcal{L}$-concepts. This is because $C \equiv D$ is equivalent to a pair of inclusion axioms $C \sqsubseteq D$ and $D \sqsubseteq C$.

At first sight, the DL that we can deal with via the above translation may not appear to be a very natural or useful one, but it includes the RDFS subset of the DAML+OIL and OWL web ontology languages, as well as a simple DL corresponding to standard frame based KR languages. Moreover, the translation can easily be extended to deal with transitive roles, symmetrical roles and inverse roles, all features of DAML+OIL and OWL.

4.2 Translating RDFS to LP

DAML+OIL and OWL are web ontology languages that are layered on top of *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 concept names. RDFS also supports axioms of the form $A \sqsubseteq B$, $R \sqsubseteq S$, $x : A$, $\langle x, y \rangle : R$, $\top \sqsubseteq \forall R.A$ and $\top \sqsubseteq \forall R^-.A$, where A, B are atomic concept names, R, S are role names, x, y are individual names, $\top$ is the "top" concept (equivalent to $C \sqcup \neg C$ for any concept C) and R^- is the inverse (converse) of the role R.

Concept inclusion axioms in this DL are trivially translated to LP using T, i.e., $A \sqsubseteq B$ can be translated as $B(?x) \leftarrow A(?x)$. Similarly, a role inclusion axiom $R \sqsubseteq S$ can be translated as $S(?x, ?y) \leftarrow R(?x, ?y)$. Ground facts of the form $x : A$ and $\langle x, y \rangle : R$ can also be easily translated into LP as $A(x) \leftarrow$ and $R(x, y) \leftarrow$ respectively.

The final two forms of axiom, $\top \sqsubseteq \forall R.A$ and $\top \sqsubseteq \forall R^-.A$, correspond to RDFS range and domain constraints. Again, these can easily be translated into LP by translating $\top$ as an empty rule body and using T to translate $\forall R.A$. The inverse role R^- can then be dealt with simply be reversing the order of the variables in the LP predicate, i.e., $\top \sqsubseteq \forall R^-.A$ is translated as $C(?y) \leftarrow R(?y, ?x) \leftarrow$.

4.3 Extending the translation

Simple frame language typically support (at least) 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, this language can be extended with role axioms corresponding to those supported in DAML+OIL and OWL. In particular, for roles R, S, asserting that R is transitive corresponds to the LP rule $R(?x, ?z) \leftarrow R(?x, ?y) \wedge R(?y, ?z)$, asserting that R is the inverse of S corresponds to the LP rule $R(?x, ?y) \leftarrow S(?y, ?x)$, and asserting that R is symmetrical corresponds to the LP rule $R(?x, ?y) \leftarrow R(?y, ?x)$ (a special case of inverse).

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.