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

Description Rules: 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 0.5-brief – Aug. 23, 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.

We give a new technique to combine DL and LP: via *Description Rules (DR)*, a new intermediate KR that is contained within the expressive intersection of DL and LP. We show how to perform *DR-fusion*: bidirectional translation of premises and inferences from the DR fragment of DL to LP, and vice versa from the DR 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 the DL ontological definitions for the vocabulary primitives (e.g., predicate, individual, and function constants) used by the rules. Conversely, the DR-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.

1 About this document

Former Title up through version 0.4 was: "Combining Rules with Ontologies for the Semantic Web: A proposal for DAML-Rules, combining DAML+OIL with RuleML (extended abstract)"

version 0.1: 7/18/01 (hand-written notes by BG)
version 0.2: 8/08/01 (hand-written notes by BG)
version 0.3: 8/17/01 (first circulated, within Joint Committee only)
version 0.4: 11/30/01
version 0.5-private: 8/15/02
version 0.5-brief: 8/23/02

2 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.

More precisely, any particular KR includes not only a syntactic language but also a semantics that specifies for each set of premises what is associated set of sanctioned conclusions that can be drawn from those premises. Expressions in a KR include: 1) premises; 2) inferencing tasks, e.g., queries; and 3) the results of inferencing tasks, i.e., conclusions or answer/binding sets (in short, inferences).

We give a new technique to combine DL and LP: via *Description Rules (DR)*, a new intermediate KR that is contained within the expressive intersection of DL and LP. (DR is a large expressive fragment of this intersection.) We show how to perform *DR-fusion*: bidirectional translation of premises and inferences from the DR fragment of DL to LP, and vice versa from the DR 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 the DL ontological definitions for the vocabulary primitives (e.g., predicate, individual, and function constants) used by the rules. Conversely, the DR-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.

We call it "DR-fusion" because it fuses the two logical KR's – DL and LP – so that information from each can be used in the other.

The DR-fusion technique is potentially quite useful and powerful, but it is also challenging to develop fully. As an initial step, we give a partial theory (with declarative KR semantics) for translation and integration of DL to and from LP. More specifically, we focus on logically-monotonic LP (mon-LP), beginning with Horn LP, which is close (but not identical) to the Horn-clauses subclass of first-order-logic (Horn-FOL). (We say "close" because the basic kind of LP uses the Unique Names Assumption, which is not assumed always in Horn-FOL or DL.) Part of our theory includes defining the expressive intersection of DL and mon-LP/Horn-FOL. (Neither DL nor mon-LP/Horn-FOL expressively subsumes the other.) Our theory enables one to view a KB as either a set of DL axioms (e.g., premises) or as a set of mon-LP/Horn-FOL rules, and to translate them back and forth between the two representations. Interestingly, the axioms might also be conclusions (e.g., ground facts) drawn from more expressively-general DL or LP/Horn-FOL beyond the intersection. The DR-fusion technique enables one to have a choice of either DL inferencing tools or mon-LP/Horn-FOL inferencing tools, for information in the expressive intersection.

The DR-fusion technique also lays the groundwork for a possible new integral yet hybrid KR that combines some DL expressiveness with some mon-LP/Horn-FOL expressiveness and goes beyond the expressive intersection. Of course, general FOL is a candidate for such a hybrid KR; however, general FOL is often impractical due to its computational complexity/semi-decidability. To be truly valuable, a new such hybrid KR would have usefully increased expressiveness along with efficient algorithms for important tasks. But finding that new KR remains for further exploration.

In the near term, a prime challenge for the Semantic Web is to make real the intuition expressed by rules/logic being depicted as sitting on top of ontologies in Tim Berners-Lee's famous wedding-cake diagram (this in turn has long roots in the philosophical logic and AI KR literature). We largely accomplish this vision in our overall approach by using the DR-fusion technique. In particular, we import info into RuleML from an expressive fragment of DAML+OIL (i.e., "a compliance level" of DAML+OIL corresponding to an expressive subclass of the intersection between DL and LP). Elegantly, our techniques and theory (especially about the intersection of DL with mon-LP/Horn-FOL) also enable the converse direction as well: info can be imported into DAML+OIL from RuleML to enable "ontologies on top of rules" in a manner analogous to Ontolingua utilizing KIF.

3 DR-fusion technique: overview

4 Preliminaries

IAN: *We need to introduce some basics about DL and LP*

4.1 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 $y = z$.

5 Outline of Essential Expressive Restrictions in DL and LP

```
o LP's essential restrictions on premises, e.g.,
- no head disjunction (or existential)
- conjunction OK in head, as extension; can rewrite
- disjunction OK in body, as extension; can rewrite
- conjunction of expressions OK in head or body, as extension; can rewrite
- disjunction of expressions OK in body, as extension; can rewrite
```

- no negation, for montonic (‘‘mon’’) LP -- except as syntactic sugar,
which is OK as extension;
negation as failure for nonmonotonic (‘‘nonmon’’) LP
- for basic kind of LP, treat equality and the domain aspect of models
differently from Horn/FOL in that one considers only Herbrand models;
this implies the unique names assumption (UNA). However, LP can be extended
to treat equality in a manner similar to Horn/FOL; this is fairly
straightforward semantically but more complicated algorithmically.

o DL's essential restrictions on premises, e.g.,
- arity of class is 1
- arity of property is 2
- can't directly represent bigger arity predicates

o LP's essential restrictions on queries and inferencing, e.g.,
- conclusions are ground facts/literals
- queries are literals (may have variables/complex-terms however)
- ... or might define queries more complicatedly if can reduce to that form
plus some new added premises, e.g., to derive rules such as subsumption

o DL's essential restrictions on queries and inferencing, e.g., typicall focus
on subsumption and satisfiability. Example inferencing tasks are:
- subclass subsumption query
- type(class-membership) query for an individual
- property values queries
- which are the individuals that are instances of a given class
(can always be done by enumerating the finite set of named individuals
and query for each, since only named ones could be answers to a query)
- is this class consistent (is it subsumed by the inconsistent class);
note this probably cannot be directly pertinent to monotonic LP which
is always consistent
- weak query languages (retrieval and instantiation)
- least common subsumers
- ...

6 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 into LP, for which the mapping is relatively simple and straightforward.

The first four are T-box statements.

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 DL statements are A-box statements.

An A-box 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 A-box 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 T-box 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.

```
Still Todo:  More kinds of DL statements cf. OWL:

- equality and inequality between individuals
- equivalence and disjointness between classes or properties
- class defined extensionally, i.e., oneOf
- inverse functionality of a property
- universal local range restriction
- existential local range restriction
- required value of a property
```

7 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.

   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.

6. $\geq_k$, i.e., minimum cardinality.

   Usage: cardinality applies to a property, and results in a class. I.e., $\geq_k P.C$ means that property P has at least k distinct values of type C.

   Intuitively, this is OK "in body" but not "in head". This is because this is essentially existental in nature. One can write this in LP in the usual way as

   $\exists ?Y1, ?Y2, \ldots, ?Yk.\ P(?X, ?Y1) \wedge P(?X, ?Y2) \wedge \ldots P(?X, Yk) \wedge ((?Y1 \neq Y2) \wedge \ldots \wedge (?Y1 \neq Yk) \wedge (?Y2 \neq Yk) \wedge \ldots) \wedge (C(?Y1) \wedge C(?Y2) \wedge \ldots \wedge C(?Yk))$

   (Note this requires explicit equality in the LP language.) This can thus be reduced (?) to a class definition.

   We need to restrict what kind of DL constructors may appear in C – they must be ones which are OK to appear "in body".

7. $\leq_k$, i.e., maximum cardinality.

   Usage: cardinality applies to a property, and results in a class. I.e., $\leq_k P.C$ means that property P has at most k distinct values of type C.

   Intuitively, this is OK "in head" but not "in body" IF the LP language permits ***explicit equality*** in the head (as with $fctnal(P)$). Also as with $fctnal(P)$, this in a sense ***requires modification of the UNA*** in general, since it may imply that two distinctly named individuals/ground-terms are equal.

   E.g.,

   $\leq_1 P.C$

   can be rewritten as:

$(?Y1{=}?Y2) \leftarrow P(?X,?Y2) \wedge C(?Y2) \wedge P(?X,?Y1) \wedge C(?Y1)$
(let us call this whole expression $Expr1$) and
$\leq_2 P.C$
can be rewritten as:
$Expr1 \leftarrow (?Y3 \neq ?Y1) \wedge (?Y3 \neq ?Y2) \wedge P(?X,?Y3) \wedge C(?Y3)$
Then one can play this same game recursively for $k = 3$ up to arbitrary k for which the overall expression size will scale $O(k^2)$.

We need to restrict what kind of DL constructors may appear in C – they must be ones which are OK to appear "in head".

8 More Sections, including on ...

`Equality, and Horn vs. LP`

`Recursive DL2LP Mapping Function`

`Extension to exploit Negation As Failure in LP`

`LP2DL direction`

`DL Union LP`

`Typical Queries and Inferencing Tasks in DL and LP`

`Architecture for usage, algorithm sketches, and implementation/performance`

`Discussion, including implications and things that can be based on this approach`

Acknowledgements

Thanks to Stefan Decker, Peter Patel-Schneider, Michael Dean, Patrick Hayes, Jim Hendler, Richard Fikes, Tim Berners-Lee, Deborah McGuinness, and Harold Boley for helpful discussions.