

THE DATALOG^{DL} COMBINATION OF DEDUCTION RULES AND DESCRIPTION LOGICS

JING MEI¹, HAROLD BOLEY^{2,3}, JIE LI^{2,3}, VIRENDRAKUMAR C. BHAVSAR², ZUOQUAN LIN¹

¹ *Department of Information Science, Peking University, Beijing 100871, China*
{mayyam, lz} AT is.pku.edu.cn

² *Faculty of Computer Science, University of New Brunswick, Fredericton, NB, E3B 5A3, Canada*
{Jie.Li, bhavsar} AT unb.ca

³ *National Research Council of Canada, Fredericton, NB, E3B 9W4, Canada*
{Harold.Boley, Jie.Li} AT nrc.gc.ca

Uniting ontologies and rules has become a central topic in the Semantic Web. Bridging the discrepancy between these two knowledge representations, this paper introduces Datalog^{DL} as a family of hybrid languages, where Datalog rules are parameterized by various DL (description logic) languages ranging from *ALC* to *SHIQ*. Making Datalog^{DL} a decidable system with complexity of EXPTIME, we propose independent properties in the DL body as the restriction to hybrid rules, and weaken the safeness condition to balance the trade-off between expressivity and reasoning power. Building on existing well-developed techniques, we present a principled approach to enrich (RuleML) rules with information from (OWL) ontologies, and develop a prototype system combining a rule engine (OO jDREW) with a DL reasoner (RACER).

Keywords: Hybrid Rules, Description Logic, Datalog, Tableau Algorithms, SLD-resolution.

1. INTRODUCTION

Alternative architectures for the Semantic Web were proposed by several groups at the W3C Workshop on Rule Languages for Interoperability, and follow-up discussions helped to establish the Rule Interchange Format Working Group (RIF 2005). Whether, in the Semantic Web's layered structure, there should be only one homogeneous hierarchy for ontologies and rules (Horrocks *et al.* 2005b), or these should stand heterogeneously (hybridly) side by side under a logic framework (Kifer *et al.* 2005), the combination of ontologies and rules, within a practical and feasible framework, is an interesting topic deserving more investigation.

Description logics (DLs) (Baader *et al.* 2003) have been recognized as the logical foundation of ontologies in the Semantic Web, and the Web Ontology Language (OWL 2004) has two species, OWL-Lite and OWL-DL, that are closely related to DL languages, *SHIQ(D)* and *SHOIN(D)*, respectively. On the other hand, Datalog is a wide-spread rule-based language, even popular in the industry. That is, both of these knowledge representations have reached a certain level of maturity, which make them suitable candidates for combination.

Among frameworks for uniting rules and DLs (see Table 1), homogeneous approaches – like DLP (Grosz *et al.* 2003), SWRL (Horrocks *et al.* 2005a), KAON2 (Motik *et al.* 2005), and $\mathcal{ALC}_{\mathbb{P}}^u$ (Mei *et al.* 2007) – can be distinguished, cf. the double-bar in Table 1, from hybrid approaches, like AL-log (Donini *et al.* 1998), CARIN (Levy and Rousset 1998), HEX-programs (Eiter *et al.* 2006) originating from dl-programs (Eiter *et al.* 2004), $\mathcal{DL}+\log$ (Rosati 2006) originating from r-hybrid KBs (Rosati 2005), and Datalog^{DL}. We call an approach *homogeneous* if rule predicates are shared with DL classes

and DL properties, and if interpretations are defined in a uniform manner for the integrated language. Conversely, we call an approach *hybrid* if rule and DL predicates are kept separate, and if interpretations are partitioned for the combined language. Whichever approach is employed for uniting rules and DLs, there exists the usual trade-off between the expressivity of languages and the complexity of their reasoning services.

Table 1: Comparison of frameworks for uniting rules and DLs

	Datalog Safeness	$\mathcal{DL}+\log$ Safeness	DL Safeness	AL-log Safeness	Uni-direction Information Flow	Bi-direction Information Flow	Implementation Strategy
DLP	✓					✓	Reduction
SWRL	✓					✓	Reduction
KAON2			✓			✓	Reduction
$\mathcal{ALC}_{\mathbb{P}}^u$		✓				✓	Tableaux-based
AL-log				✓	✓		SLD-resolution
CARIN	✓				✓		Entailment
HEX-programs		✓				✓	Fixpoint Iteration
$\mathcal{DL}+\log$		✓				✓	Fixpoint Iteration
Datalog ^{DL}	✓				✓		SLD-resolution

AL-log, an early and simple hybrid language, combines standard Datalog rule inference procedures with intermediate \mathcal{ALC} DL satisfiability checking. It adopts backward chaining (based on SLD-resolution), first collecting the disjunction of the obtained DL queries, and then using classical DL tableaux algorithms to check the consistency of those DL atoms. As a result, AL-log is a complete and sound system, whose complexity is EXPTIME stemming from the complexity of \mathcal{ALC} and Datalog. However, binary predicates (i.e., DL properties) are not considered in AL-log, and it requires that each variable appearing in the DL component also appears in the Datalog component (we call this the *AL-log safeness* condition, whose formal definition will be presented below), s.t. only atoms with unary predicates and no variables (i.e., ground class atoms) will be submitted to the DL tableaux reasoner.

Increasing generality, CARIN is a family of languages, each of which combines (a sublanguage of) $\mathcal{ALCN}\mathcal{R}$ DL and Datalog rules. Unlike AL-log, CARIN first computes the entailments of the DL component based on DL tableaux algorithms, and one step of the standard forward chaining is then done for each augmented rule component, using the added DL assertions as new facts. Moreover, CARIN allows ground as well as open DL queries with unary and binary predicates. The variables appearing in the head of a rule are required to also appear in the body, although not necessarily in the DL body (we call this the *Datalog safeness* condition, whose formal definition will be presented below) – this is a general safeness condition for rule-based languages, weaker than that of AL-log. For *non-recursive* CARIN- $\mathcal{ALCN}\mathcal{R}$, a sound and complete inference procedure has been established, while reasoning in *recursive* CARIN- $\mathcal{ALCN}\mathcal{R}$ is undecidable, although there are two ways of restricting expressivity to regain decidability: one is to remove some DL constructors and allow an acyclic terminology only, while the other is to make the safeness condition strong, concerning only role-safe rules. There, role-safe rules, as defined in (Levy and Rousset 1998), require that in every DL binary atom at least one variable that appears in the atom also appears in an atom of a base predicate (i.e., a predicate that does not appear in the rule head and is not a DL class or a DL property).

It should be pointed out that bi-directional information flows are not permitted in the above two systems, and the predicate symbols in the head of hybrid rules are disjoint from those in the DL

component. Two other well-known hybrid systems, dl-programs (upgraded to HEX-programs) and r-hybrid KBs (upgraded to $\mathcal{DL}+\log$), are less restricted, and the stable model semantics performs well for both systems; also, they each provide a decidable strategy. In these systems, negation as failure is investigated as an important feature, which is beyond this scope of the current paper. Interestingly, (Rosati 2006) also proposes a safeness condition, which requires each variable appearing in the rule head also appears in the Datalog component (we call this the $\mathcal{DL}+\log$ safeness condition, whose formal definition will be presented below).

Being homogeneous approaches, DLP and SWRL share all of the predicate symbols between the rule component and the DL component. However, DLP has more expressivity restrictions, while SWRL is undecidable. KAON2 seems a novelty as to reasoning support for both OWL-DL and rules, reducing the DL knowledge bases to disjunctive programs. But such reduction pushes the task of DL reasoning completely into rule engines, not gaining the benefits from the existing tableaux DL reasoners. Besides, a *DL safeness condition* is required by KAON2, which requires that each variable appearing in the DL body also appears in the Datalog body (again, the formal definition will be presented below), although this restriction covers some of the common usages of DL expressivity.

Since the homogeneous-vs.-hybrid issue is still open, we have been pursuing both approaches with the homogeneous $\mathcal{ALCC}_{\mathbb{P}}^u$ (Mei *et al.* 2007) and the hybrid Datalog^{DL} (Mei *et al.* 2006). An $\mathcal{ALCC}_{\mathbb{P}}^u$ KB consists of a TBox of subsumptions, an ABox of assertions, and a novel PBox \mathbb{P} of general rules that share predicates with DL classes and DL properties, making a *unique names assumption* in open answer set semantics. While (Mei *et al.* 2007) attempts to extend tableaux-based DL algorithms into integrated decision procedures for the $\mathcal{ALCC}_{\mathbb{P}}^u$ KB satisfiability and the query entailment problems, Datalog^{DL} strives for a decidable combination of existing syntaxes, semantics, and algorithms, i.e. DL reasoners and rule engines, thus reusing and extending earlier theoretical results as well as developing hybrid tools and facilitating practical use cases.

In this paper, our objective is to generalize the framework of AL-log, combining (any sublanguage of) a decidable DL system with Datalog, and provide less restricted hybrid rules with DL queries to both classes and properties. Although CARIN is similar in this respect, it requires some built-in coding into a DL reasoner, to obtain a complete entailment for hybrid rules; otherwise, anonymous individuals (e.g., introduced by existential restrictions) and uncertain assertions (e.g., derived from disjunctive descriptions) in the DL component will just be kept inside of the primitive DL reasoner, with no access to rule engines. Aiming at developing a feasible strategy for the Semantic Web community by employing existing techniques as much as possible, we attempt to balance the trade-off of expressivity and reasoning power, and consider *SHIQ* DL as our bottom line, for which practical and efficient tools are available, such as RACER (Haarslev and Möller 2001). Here, we adopt the most general **Datalog safeness** condition, and the problems introduced by *pure-DL variables* in DL queries will be handled cautiously, provided that those expressive statements would be eliminated by the bottom line of *SHIQ* DL. By defining *independent properties*, we characterize our reasoning services: hybrid rules with DL queries to classes and independent properties obeying the Datalog safeness condition are fully supported.

Consequently, this paper presents Datalog^{DL} as a family of hybrid representation languages, where Datalog rules are parameterized by some DL language \mathcal{L} such that in Datalog ^{\mathcal{L}} the \mathcal{L} ranges from \mathcal{ALC} , \mathcal{ALC}_{R^+} (namely \mathcal{S}), \mathcal{SI} , \mathcal{SHI} to \mathcal{SHIQ} . On the theoretical side, we show a sound and complete algorithm for reasoning in Datalog^{DL}, with the complexity of EXPTIME for any DL language parameter \mathcal{L} in its range. On the practical side, a typical rule engine, e.g. OO jDREW (Ball *et al.* 2005), will be

extended to incorporate hybrid rules, where the collection of DL queries, after a so-called constrained SLD-resolution for hybrid rules, will be submitted to an unchanged external DL reasoner, e.g. RACER.

Next, Datalog^{DL} is introduced in Section 2 with its syntax and semantics, while its reasoning is described in Section 3 together with proofs of soundness and completeness. Section 4 illustrates the expressivity and reasoning power of Datalog^{DL} with a suite of examples and use cases. Section 5 clarifies technical issues of (un)decidability when uniting DL and rules, and finally conclusions are drawn in Section 6.

2. THE DATALOD^{DL} LANGUAGE

The matured languages, Datalog and DL, will be combined in a hybrid approach: Datalog^{DL} is a family of languages, each of which parameterizes Datalog with a variety of DL queries.

Consider the main layers of the DL family bottom-up (Baader *et al.* 2003; Horrocks *et al.* 1999), \mathcal{ALC} is a basic and simple language, permitting class descriptions via $C \sqcap D, C \sqcup D, \neg C, \forall P.C$, and $\exists P.C$ where C, D are classes and P is a property. Augmented by transitive properties, \mathcal{ALC} becomes \mathcal{ALC}_{R^+} in the following denoted by \mathcal{S} . \mathcal{SI} is an extension to \mathcal{S} with inverse properties, followed by \mathcal{SHI} with property hierarchies. It becomes \mathcal{SHIF} if extended by functional restrictions, \mathcal{SHIN} if extended by cardinality restrictions, and \mathcal{SHIQ} if extended by qualified number restrictions. Support for datatype predicates (e.g., string, integer) leads to the concrete domain of \mathbf{D} , and using nominals \mathcal{O} allows to construct classes from singleton sets.

Assuming the usual definitions of DLs and rules are familiar to readers, cf. (Baader *et al.* 2003; Horrocks *et al.* 1999) and (Lloyd 1987), we introduce the syntax and semantics for Datalog^{DL} with no need for preliminaries. Besides, compatible with OWL and DL, the so-called unique names assumption (UNA) is not imposed for the whole Datalog^{DL} language, but interpreting hybrid rules relies on Herbrand interpretations where UNA applies.

2.1. Syntax

In order to preserve decidability, we fix the rule language to Datalog, so that terms must be variables or constants (viz. named individuals). Undecidable extensions, such as Horn logic, where terms can also be function applications, have been considered as well, but are beyond the scope of this paper. Alternatively, (decidable) built-ins, such as arithmetic operators and aggregate functions, might be considered as a workaround for datatype predicates (e.g., string, integer) concerning the concrete domain \mathbf{D} of $\mathcal{SHIQ}(\mathbf{D})$.

Given a decidable DL language \mathcal{L} (here, it ranges from \mathcal{ALC} to \mathcal{SHIQ}), we denote by Datalog ^{\mathcal{L}} a subset of the function-free first-order Horn logic language over an alphabet of predicates $\mathcal{N} = \mathcal{N}_{\mathcal{T}} \cup \mathcal{N}_{\mathcal{P}}$, with $\mathcal{N}_{\mathcal{T}} \cap \mathcal{N}_{\mathcal{P}} = \emptyset$, and an alphabet of constants \mathcal{C} . Note that, the predicates in $\mathcal{N}_{\mathcal{P}}$ can be of arbitrary arity, while those of $\mathcal{N}_{\mathcal{T}}$ should be either unary (also called class in DL) or binary (also called property in DL).

Definition 1. A Datalog ^{\mathcal{L}} knowledge base \mathcal{K} is a pair (Σ, Π) , where: Σ is a \mathcal{L} -based description logic knowledge base with predicates in $\mathcal{N}_{\mathcal{T}}$; Π is a Datalog program with DL queries to Σ , s.t. each hybrid rule r in Π is

$$h(\vec{u}) : - b_1(\vec{v}_1), \dots, b_m(\vec{v}_m) \ \& \ q_1(\vec{w}_1), \dots, q_n(\vec{w}_n)$$

where $h(\vec{u})$ and $b_i(\vec{v}_i)$ are Datalog atoms with $h \in \mathcal{N}_{\mathcal{P}}$ and $b_i \in \mathcal{N}_{\mathcal{P}}$, also \vec{u} and \vec{v}_i are term sequences of arbitrary arity, while each $q_j(\vec{w}_j)$ is a DL query with $q_j \in \mathcal{N}_{\mathcal{T}}$ and \vec{w}_j is a unary/binary term sequence, for all $1 \leq i \leq m$ and $1 \leq j \leq n$.

Four safeness conditions are introduced for hybrid rules:

Datalog safeness: A variable occurring in \vec{u} must occur in one of the $\vec{v}_i|\vec{w}_j$'s.

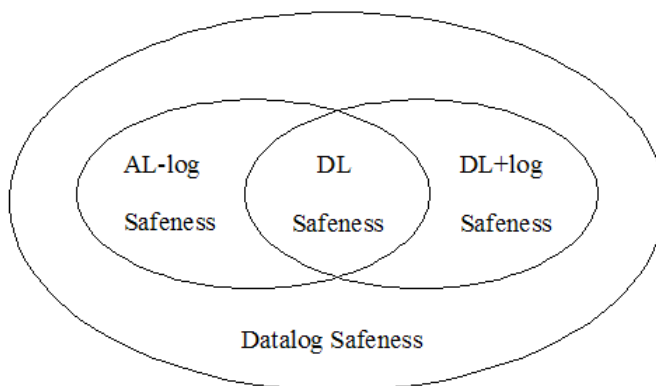
AL-log safeness: A variable occurring in \vec{w}_j 's must occur in one of the $\vec{u}|\vec{v}_i$'s.

$\mathcal{DL}+\log$ safeness: A variable occurring in \vec{u} must occur in one of the \vec{v}_i 's.

DL safeness: A variable occurring in r must occur in one of the \vec{v}_i 's.

By $[\vec{u}]$, we denote the set of those variables appearing in the term sequence \vec{u} , and we can rewrite safeness conditions more formally.

- Datalog safeness
 $[\vec{u}] \subseteq \bigcup_{i=1}^m [\vec{v}_i] \cup \bigcup_{j=1}^n [\vec{w}_j]$
- AL-log safeness
 $\bigcup_{j=1}^n [\vec{w}_j] \subseteq [\vec{u}] \cup \bigcup_{i=1}^m [\vec{v}_i]$
- $\mathcal{DL}+\log$ safeness
 $[\vec{u}] \subseteq \bigcup_{i=1}^m [\vec{v}_i]$
- DL safeness
 $[\vec{u}] \cup \bigcup_{j=1}^n [\vec{w}_j] \subseteq \bigcup_{i=1}^m [\vec{v}_i]$



We remark that DL safe rules are AL-log safe, and also are $\mathcal{DL}+\log$ safe; Datalog safeness is the most general condition; see the diagram for the complete inclusion hierarchy. For simplicity, in the rest of the paper “rule” means “hybrid rule”, while “Datalog rule” refers to a hybrid rule after deletion of the DL body. Besides, making rules strongly DL safe has been introduced in (Motik *et al.* 2005), that is: (1) For each rule r whose variable x does not occur in any of the \vec{v}_i 's, we add an atom $\mathcal{O}(x)$ to the Datalog body of r , where \mathcal{O} is a special predicate symbol and $\mathcal{O} \in \mathcal{N}_P$; (2) For each named individual $c \in \mathcal{C}$ occurring in $\mathcal{K} = (\Sigma, \Pi)$, we add a fact $\mathcal{O}(c)$ to Π .

As mentioned in Section 1, we prefer to the Datalog safeness condition rather than the AL-log, $\mathcal{DL}+\log$ or DL ones. Below, *pure-DL* variables are defined.

Definition 2. A *pure-DL* variable x in a rule r is a variable that only occurs in one of the \vec{w}_j 's, i.e., $x \in \bigcup_{j=1}^n [\vec{w}_j] \setminus \bigcup_{i=1}^m [\vec{v}_i] \setminus [\vec{u}]$

The presence of pure-DL variables, to which $\mathcal{DL}+\log$ safeness pays little attention, leads to a violation of the AL-log safeness condition (also the DL safeness) in cases where the Datalog safeness condition is obeyed. Note that, disregarding pure-DL variables, our system turns to be a straightforward extension to AL-log, namely, a combination of Datalog rules with conjunctive DL queries where named individuals are ready for being semantically assigned to all rule variables.

According to the classical SLD-resolution with rules, non-pure-DL variables in (the DL body of) r will be bound to ground values, still leaving pure-DL variables free in the DL body. This situation is similar to conjunctive query answering in DL containing both constants and variables (Horrocks and Tessaris 2002). Instantiation (Is an individual an instance of a class?) can be reduced to KB unsatisfiability by transforming the query into a negated assertion. However, queries involving properties and variables are nontrivial given that the negation of properties is not supported by most DLs. Hence, a candidate technique is folding, called rolling-up in (Horrocks and Tessaris 2002), whose objective is to eliminate properties from DL queries.

Following this route, we encounter another problem: the simple procedure of folding cannot be applied to parts of the query that contain cycles, or where more than one arc enters a node that corresponds to a variable (e.g., $P(u, x) \wedge Q(v, x)$). Requirement of tree-shaped DL queries might be a solution to this problem by exploiting the tree model property of the DL (Horrocks and Tessaris 2002); however, adding rules to DLs possibly causes the loss of any form of tree model property (Motik *et al.* 2005). Hence, the DL safeness is introduced for DL-safe rules (Motik *et al.* 2005) and other approaches (Eiter *et al.* 2004; Rosati 2005), while we define *independent properties*, which address the trade-off as mentioned above.

Definition 3. *A property P is said to be independent in a rule r , if no P occurrence shares any pure-DL variable with other property occurrences (including other P occurrences).*

Now, suppose r is a hybrid rule violating the AL-log safeness condition, γ being its head, α being its Datalog body, and β being its DL body. Specifically, it has the form $\gamma : -\alpha \ \& \ \beta$, where β contains a pure-DL variable x having a class description C (C can be the DL top class \top). We classify the possibilities for β into four cases:

1. If x does not participate (as the first or second argument) in any property, then the DL query of $C(x)$ is reduced to checking whether C is nonempty.
2. If there exists exactly one property occurrence of P relating x with a term u , then the DL query of $P(u, x) \wedge C(x)$ or $P(x, u) \wedge C(x)$ becomes its *folding* result $\exists P.C(u)$ or $\exists P^-.C(u)$, respectively.
3. If there exists exactly two property occurrences of P and Q relating x with terms u and v , respectively, where P and Q , u and v can be identical, then the DL queries become the results of following *foldings* (chaining can start with either u or v):
 - (a) $P(u, x) \wedge Q(v, x) \wedge C(x)$ becomes $\exists P.(\exists Q^-. \{v\} \sqcap C)(u)$ or $\exists Q.(\exists P^-. \{u\} \sqcap C)(v)$
 - (b) $P(u, x) \wedge Q(x, v) \wedge C(x)$ becomes $\exists P.(\exists Q. \{v\} \sqcap C)(u)$ or $\exists Q^-.(\exists P^-. \{u\} \sqcap C)(v)$
 - (c) $P(x, u) \wedge Q(v, x) \wedge C(x)$ becomes $\exists P^-.(\exists Q^-. \{v\} \sqcap C)(u)$ or $\exists Q.(\exists P. \{u\} \sqcap C)(v)$
 - (d) $P(x, u) \wedge Q(x, v) \wedge C(x)$ becomes $\exists P^-.(\exists Q. \{v\} \sqcap C)(u)$ or $\exists Q^-.(\exists P. \{u\} \sqcap C)(v)$
4. If there exists three or more property occurrences, nested foldings might be employed by iterating case 3 chainings.

Case 3 requires support by using nominals \mathcal{O} (i.e., classes with a singleton extension), as known from the DL literature, whose interaction with cardinality restrictions \mathcal{N} and inverse properties \mathcal{I} makes the complexity jump from EXPTIME (for *SHIN*) to NEXPTIME (for *SHOIN*). Despite that the operator of $\{u\}$ can be ‘simulated’ by its *representative* class C_u (Horrocks and Tessaris 2002), we still focus on cases 1 and 2 in this paper, not introducing different fresh class names for different individuals. The consideration is also following the requirement of independent properties in a hybrid rule r , which is fulfilled by cases 1 and 2, excluding cases 3 and 4 where the pure-DL variable x is a variable shared among properties in r .

Proposition 1. *For hybrid rules with independent properties according to case 2, the folding results are equivalent to the original DL queries.*

Proof. For a set of closed formulas S and a closed formula F of a first order language, F is a logical consequence of S iff $S \cup \{\neg F\}$ is unsatisfiable. Applied to logic programming, consider a Datalog

program Π with a goal G of the form $\leftarrow G_1 \wedge \dots \wedge G_n$ with variables y_1, \dots, y_m . Showing that the set of clauses $\Pi \cup \{G\}$ is unsatisfiable is exactly the same as showing that $\exists y_1 \dots \exists y_m (G_1 \wedge \dots \wedge G_n)$ is a logical consequence of Π . Note that DL languages are variable-free, where any free variables are hidden within \forall, \exists , etc., such as $u \in \exists P.C$ meaning $u \in \{x \mid \exists y. P(x, y) \wedge C(y)\}$. So, the folding results, e.g., $\exists P.C(u)$, are equivalent to the original DL queries, e.g., $\leftarrow P(u, x) \wedge C(x)$ with an independent property of P . \square

2.2. Semantics

The semantics of Datalog^{DL} derives in a natural way from the semantics of its component languages. We direct readers to the Description Logic Handbook (Baader *et al.* 2003) and the Foundations of Logic Programming (Lloyd 1987) for relevant definitions, and below gives sketches for DL and Herbrand interpretations.

A DL interpretation $\mathcal{I} = (\Delta_{\mathcal{I}}, \bullet^{\mathcal{I}})$ for a description logic language \mathcal{L} consists of a nonempty domain $\Delta_{\mathcal{I}}$ and a function $\bullet^{\mathcal{I}}$ mapping (1) every named individuals $c \in \mathcal{C}$ to an element in $\Delta_{\mathcal{I}}$; (2) every class $C \in \mathcal{N}_{\mathcal{I}}$ to a subset of $\Delta_{\mathcal{I}}$; (3) every property $P \in \mathcal{N}_{\mathcal{I}}$ to a subset of $\Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$. Also certain DL semantic conditions are satisfied, cf. (Baader *et al.* 2003).

Let \mathcal{R} be a set of Datalog rules. Its Herbrand universe $HU_{\mathcal{R}}$ consists of all named individuals in \mathcal{R} , and its Herbrand base $HB_{\mathcal{R}}$ is the set of all ground atoms of the form $Pr(o_1, \dots, o_n)$ where $Pr \in \mathcal{N}_P$ is a n -ary predicate in \mathcal{R} and for each $1 \leq i \leq n$, $o_i \in HU_{\mathcal{R}}$. A Herbrand interpretation \mathcal{H} for \mathcal{R} is a subset of the Herbrand base $HB_{\mathcal{R}}$.

To make it comparable to DL interpretations, a Herbrand interpretation \mathcal{H} for \mathcal{R} can also be reformulated as $\mathcal{H} = (\Delta_{\mathcal{H}}, \bullet^{\mathcal{H}})$, where $\Delta_{\mathcal{H}}$ is exactly the Herbrand universe $HU_{\mathcal{R}}$, and the mapping $\bullet^{\mathcal{H}}$ is an identical function for named individuals s.t. $o^{\mathcal{H}} = o \in \Delta_{\mathcal{H}} \subseteq \mathcal{C}$, while for every n -ary predicate Pr , $Pr^{\mathcal{H}}$ is a subset of $\Delta_{\mathcal{H}}^n$.

Definition 4. Suppose $\mathcal{K} = (\Sigma, \Pi)$ be a knowledge base of the language Datalog^L. An interpretation $\mathcal{J} = (\mathcal{I}, \mathcal{H})$ for \mathcal{K} consists of a DL interpretation \mathcal{I} for Σ and a Herbrand interpretation \mathcal{H} for $\Pi_{\mathcal{R}}$, where $\Pi_{\mathcal{R}}$ is the set of Datalog rules obtained from hybrid rules in Π by deleting the DL atoms from every hybrid rule.

Given an interpretation \mathcal{J} for \mathcal{K} , ground atoms α are satisfied in \mathcal{J} , denoted by $\mathcal{J} \models \alpha$, if

- $o^{\mathcal{J}} \in C^{\mathcal{J}}$ for $\alpha = C(o)$ and $C \in \mathcal{N}_{\mathcal{I}}$ is a DL class
- $(o_1^{\mathcal{J}}, o_2^{\mathcal{J}}) \in P^{\mathcal{J}}$ for $\alpha = P(o_1, o_2)$ and $P \in \mathcal{N}_{\mathcal{I}}$ is a DL property
- $(o_1^{\mathcal{J}}, \dots, o_n^{\mathcal{J}}) \in Pr^{\mathcal{J}}$ for $\alpha = Pr(o_1, \dots, o_n)$ and $Pr \in \mathcal{N}_P$ is an n -ary predicate

Following the LP approach, the satisfiability of rules needs to address variables first. But here, the variable assignment to pure-DL variables is arbitrary, as in DL, while the remaining variables are bound into the Herbrand universe, as in LP.

Definition 5. Suppose $\mathcal{J} = (\mathcal{I}, \mathcal{H})$ be an interpretation for \mathcal{K} and r be a hybrid rule.

- A variable assignment V_r w.r.t. \mathcal{J} is an assignment to each pure-DL variable in r of an element in the domain of \mathcal{I} , and to any other variable in r , of an element in the domain of \mathcal{H} .
- A term assignment T_r w.r.t. \mathcal{J} is defined thus: (1) Each variable is given its assignment according to V_r w.r.t. \mathcal{J} ; (2) Each named individual is given its assignment according to \mathcal{J} .

A hybrid rule r of the form $h(\vec{u}) : -b_1(\vec{v}_1), \dots, b_m(\vec{v}_m) \ \&\& \ q_1(\vec{w}_1), \dots, q_n(\vec{w}_n)$ is satisfied in \mathcal{J} , denoted by $\mathcal{J} \models r$, if $T_r(\vec{u}) \in h^{\mathcal{J}}$ whenever $T_r(\vec{v}_i) \in b_i^{\mathcal{J}}$ and $T_r(\vec{w}_j) \in q_j^{\mathcal{J}}$, given $1 \leq i \leq m$ and $1 \leq j \leq n$, for every possible term assignment T_r w.r.t. \mathcal{J} .

Definition 6. Let $\mathcal{J} = (\mathcal{I}, \mathcal{H})$ be an interpretation for a knowledge base $\mathcal{K} = (\Sigma, \Pi)$ of the language *Datalog^L*. We call \mathcal{J} a model of \mathcal{K} , if \mathcal{I} is a model of Σ , and $\mathcal{J} \models r$ for each hybrid rule r in Π .

3. REASONING IN DATALOG^{DL}

Deviating from AL-log, the algorithm in CARIN is meant to test DL entailment but not satisfiability, resulting in forward chaining being employed as the strategy for the rule component. On the other hand, not concerned with the internals of DL's tableaux calculus, our *Datalog^{DL}* family is in the tradition of AL-log, making use of the constrained SLD-resolution, and backward chaining plays the role of our principal reasoning strategy.

3.1. Algorithm

Below is the specification of an algorithm, in pseudo-code, for reasoning in *Datalog^L*, where \mathcal{L} is a DL language ranging from *ALC* to *SHIQ*, restricted to independent properties in the DL body of hybrid rules under the *Datalog* safeness condition.

Input: *Datalog^L* KB $\mathcal{K} = (\Sigma, \Pi)$ and a query q .

Output: TRUE if q is entailed by \mathcal{K} , FALSE otherwise.

BEGIN

1. Apply SLD-resolution for q with *Datalog* rules. Use the resulting substitution to ground the hybrid rules (no assignment can be made to pure-DL variables). If there is no such grounded version, then return FALSE. Otherwise, collect the disjunction of the obtained DL queries, after folding in step-2 for each rule r having pure-DL variables left.
2. For each pure-DL variable x in the rule r , where C is the class description of x , and P is an independent property relating x with a term u , output the folding results of $\exists P.C(u)$ from $P(u, x) \wedge C(x)$, and of $\exists P^-.C(u)$ from $P(x, u) \wedge C(x)$.
3. Apply the DL tableaux algorithm to (the step-2 folding results of) the DL queries from step-1. We build a disjunctive DL class $D_1 \sqcup \dots \sqcup D_m$ such that its class description D_i is collected from the involved hybrid rule r_i , where $1 \leq i \leq m$. For an individual a , the separate DL query $D_i(a)$ will be replaced by a single new one, $D_1 \sqcup \dots \sqcup D_m(a)$. If the DL query $D_1 \sqcup \dots \sqcup D_m(a)$ in addition to at least one of the remaining disjuncts are satisfiable in every model, then return TRUE, else return FALSE.

END.

The hybrid rules from the *Datalog^L* KB \mathcal{K} input obey the restriction of only having independent properties, as imposed by our definition of \mathcal{K} , s.t. step-2 produces ground rules under the *Datalog* safeness condition. For rules fulfilling the AL-log safeness condition, step-2 will be skipped due to the non-appearance of pure-DL variables. That is, our algorithm introduces a method to eliminate

all pure-DL variables, while a collection of ground DL queries will be submitted to a DL reasoner for satisfiability checking.

Instead of processing the rule bodies separately, step-3 evaluates them as a single disjunction. As a simple example consider a DL TBox with one axiom $\top \sqsubseteq A \sqcup B$ as well as two hybrid rules that $C(x) : \neg \& A(x)$. and $C(x) : \neg \& B(x)$. Additionally, there is a named individual a asserted. Given a query $C(a)$, neither $A(a)$ nor $B(a)$ holds, while step-3 allows to finalize this query via $A \sqcup B(a)$ to which the DL reasoner replies ‘True’.

3.2. Query Answering

In general, a substitution θ is a finite set of the form $\{x_1/t_1, \dots, x_n/t_n\}$, where x_i is a variable, t_i is a term, and $x_i \neq x_j$ for $i \neq j$. A ground substitution is a substitution where t_i is a constant for every $i \in \{1, \dots, n\}$. Below are the technical details for query answering, using the notions inherited from AL-log but with extensions to DL properties.

Definition 7 (Constrained SLD-resolution). *Let \mathcal{L} be a specific DL language, $\mathcal{K} = (\Sigma, \Pi)$ be a Datalog ^{\mathcal{L}} knowledge base, $q = \alpha_1, \dots, \alpha_s \& \beta_1, \dots, \beta_t$ be a query to \mathcal{K} where α_i is a Datalog atom and β_j is a DL atom ($1 \leq i \leq s, 1 \leq j \leq t$), and r be a hybrid rule of the form $\alpha' : \neg \alpha'_1, \dots, \alpha'_m \& \beta'_1, \dots, \beta'_n$. Suppose θ is the most general substitution such that $\alpha'\theta = \alpha_k\theta$, where α_k is one of $\{\alpha_1, \dots, \alpha_s\}$. The resolvent of q and r with substitution θ is the query $q' = \mu \& \nu$, where $\mu = (\alpha_1, \dots, \alpha_{k-1}, \alpha'_1, \dots, \alpha'_m, \alpha_{k+1}, \dots, \alpha_s)\theta$ and $\nu = (\beta_1, \dots, \beta_t, \beta'_1, \dots, \beta'_n)\theta$ with simplification: if there are two DL atoms of the form $C(t)$ and $D(t)$, they are replaced by the equivalent DL atom $C \sqcap D(t)$.*

Definition 8 (Constrained SLD-derivation). *A constrained SLD-derivation for a query q_0 in \mathcal{K} is a derivation constituted by:*

1. A sequence of queries q_0, q_1, \dots, q_n
2. A sequence of hybrid rules r_1, \dots, r_n
3. A sequence of substitutions $\theta_1, \dots, \theta_n$

such that for each $i \in \{0, 1, \dots, n-1\}$, q_{i+1} is the resolvent of q_i and r_{i+1} with substitution θ_{i+1} . We call n the length of the derivation.

A derivation may terminate with the last query of the form $q_{\text{DL}} = \emptyset \& \beta_1, \dots, \beta_t$, which is called *constrained empty clause*. For the AL-log safeness conditions, the constrained empty clause should have not any variable, while for the Datalog safeness conditions, pure-DL variables appear as being existentially quantified in some of “ β_1, \dots, β_t ”. In this sense, we currently only consider independent properties in hybrid rules, with folding results fully supported by existing DL reasoners.

Proposition 2. *Let q_0, q_1, \dots, q_n be a constrained SLD-derivation for q_0 in \mathcal{K} . If \mathcal{J} is a model of \mathcal{K} such that $\mathcal{J} \models q_{i+1}$, then $\mathcal{J} \models q_i$ for $i = 0, 1, \dots, n-1$*

Proof. It follows from the soundness of SLD-resolution as well as the fact that the simplification of constraints preserves validity. In particular, Proposition 1 states the folding results are equivalent to the original DL queries, also applying to the last query q_n , i.e., the constrained empty clause q_{DL} with pure-DL variables. Together with DL classical tableaux algorithms, it holds that

$\mathcal{K} \vdash \emptyset \& C(x)$ iff $C^{\mathcal{J}}$ is nonempty, where \mathcal{J} is the model of \mathcal{K}

$\mathcal{K} \vdash \emptyset \& P(u, x) \wedge C(x)$ iff $\mathcal{K} \models \exists P.C(u)$

$\mathcal{K} \vdash \emptyset \& P(x, u) \wedge C(x)$ iff $\mathcal{K} \models \exists P^-.C(u)$ □

Definition 9 (Constrained SLD-refutation). *A constrained SLD-refutation for a query q in \mathcal{K} is a finite set of constrained SLD-derivations d_1, \dots, d_m for q in \mathcal{K} such that, denoting as $q_0^i, \dots, q_{n_i}^i$ the sequence of queries of the i -th derivation d_i , the following conditions hold:*

1. *For each i , $q_{n_i}^i$ is one of the form " $\emptyset \ \& \ \beta_1^i, \dots, \beta_{l_i}^i$ ", i.e., the last query of each derivation is a constrained empty clause.*
2. *For each $q_{n_i}^i$ with pure-DL variables, obtain the folding results of $q_{n_i}^i$.*
3. *For each model \mathcal{J} of \mathcal{K} , there exists at least one $i \in \{1, \dots, m\}$ such that $\mathcal{J} \models q_{n_i}^i$; we write this condition $\mathcal{K} \models \text{disj}(q_{n_1}^1, \dots, q_{n_m}^m)$*

We write $\mathcal{K} \vdash q$, if there is a constrained SLD-refutation for q in \mathcal{K} .

Lemma 1. *Let q be a ground query to a Datalog^L knowledge base $\mathcal{K} = (\Sigma, \Pi)$. $\mathcal{K} \vdash q$ if and only if $\mathcal{K} \models q$.*

Proof. With restriction to independent properties in hybrid rules, we present our proof based on the correctness and completeness of SLD-resolution and DL tableaux algorithms, similar as AL-log does.

\Rightarrow : Suppose $\mathcal{K} \vdash q$, i.e., the ground query q has a constrained SLD-refutation. Then, for each derivation, if \mathcal{J} is a model of \mathcal{K} that satisfies the constrained empty clause q_{DL} then it satisfies q (by repeated application of Proposition 2 with q_{DL} as q_n and q as q_0); moreover, each model \mathcal{J} of \mathcal{K} satisfies at least one of the constrained empty clauses. Consequently, each model of \mathcal{K} satisfies q , that is $\mathcal{K} \models q$.

\Leftarrow : Suppose $\mathcal{K} \not\models q$, we have no constrained SLD-refutation for q in \mathcal{K} , resulting from three possibilities according to Definition 9.

1. If there is no constrained empty clause, then from the completeness of SLD-resolution, we have $\mathcal{K} \not\models q$.
2. If there is no folding results of the constrained empty clause, then this query q is beyond our consideration, having a natural conflict with $\mathcal{K} \models q$.
3. If there is a model $\mathcal{J} = (\mathcal{I}, \mathcal{H})$ of $\mathcal{K} = (\Sigma, \Pi)$, then for any derivation of q whose last query is a constrained empty clause, written as $q_{n_i}^i = \emptyset \ \& \ \beta_1^i, \dots, \beta_{n_i}^i$, it holds that $\mathcal{J} \not\models q_{n_i}^i$. That is, there is a model \mathcal{I} of Σ such that $\mathcal{I} \not\models \beta_1^i, \dots, \beta_{n_i}^i$.

By Definition 4, \mathcal{H} is a Herbrand interpretation for $\Pi_{\mathcal{R}}$, where $\Pi_{\mathcal{R}}$ is the set of Datalog rules obtained from hybrid rules in Π by deleting the DL atoms from every hybrid rule. Suppose HB be the Herbrand base of $\Pi_{\mathcal{R}}$. We define a mapping $M_{\Pi, \mathcal{I}} : 2^{HB} \rightarrow 2^{HB}$ such that, given a set $S \subseteq HB$ of ground atoms, $M_{\Pi, \mathcal{I}}(S) = \{h(T_r(\vec{u})) | h(\vec{u}) : - b_1(\vec{v}_1), \dots, b_m(\vec{v}_m) \ \& \ q_1(\vec{w}_1), \dots, q_n(\vec{w}_n)$ in Π , $\mathcal{I} \models q_j(T_r(\vec{w}_j))$ and $b_i(T_r(\vec{v}_i)) \in S$, where T_r is a term assignment, $1 \leq i \leq m$ and $1 \leq j \leq n\} \subseteq HB$.

Being a monotone mapping, $M_{\Pi, \mathcal{I}}$ has its least fixpoint $\text{lfp}(M_{\Pi, \mathcal{I}})$ obtained by iterating $M_{\Pi, \mathcal{I}}$ a finite number of times, starting from \emptyset (Donini *et al.* 1998). Consequently, we can construct $\mathcal{J}' = (\mathcal{I}, \text{lfp}(M_{\Pi, \mathcal{I}}))$ as another model of \mathcal{K} , and by induction on the construction of $\text{lfp}(M_{\Pi, \mathcal{I}})$, it can be shown that $\mathcal{J}' \not\models q$, and therefore $\mathcal{K} \not\models q$.

□

Referring to AL-log, Datalog ^{\mathcal{L}} also provides a decidable procedure. Note that satisfiability of an \mathcal{ALC} class (without any TBox) is PSPACE-complete; while the same problem is EXPTIME-complete, if a TBox with general inclusion axioms is present (Baader *et al.* 2003). For the rule component, Datalog is data complete for PTIME while program complete for EXPTIME (Dantsin *et al.* 2001). As a result, the computational complexity of Datalog ^{\mathcal{L}} is EXPTIME, where \mathcal{L} ranges from \mathcal{ALC} to \mathcal{SHIQ} .

Theorem 1. *Query answering in Datalog ^{\mathcal{L}} is a decidable problem in EXPTIME.*

4. DATALOG^{DL} TEST SUITE

The expressivity and reasoning power of Datalog^{DL} is illustrated in this section with a suite of examples from AL-log, CARIN, and DL-safe rules, followed by our use case of RuleML FOAF. This test suite covers much of the expressiveness currently discussed for hybrid rules, e.g., in the W3C RIF WG (RIF 2005). The entire suite is implemented in our hybrid rule engine (Mei 2005) coupling OO jDREW with RACER.

4.1. Examples from AL-log

In Table 2, the DL component specifies that: 1. full professors (FP) are faculty members (FM); 2. non-teaching full professors (NFP) are defined as full professors that do not teach any course; 3. the set of courses (Co) is partitioned into basic courses (BC) and advanced courses (AC).

Table 2: From AL-log (AL-log safeness condition)

\mathcal{ALC} DL	$FP \sqsubseteq FM$ $NFP = FP \sqcap \neg \exists TC.Co$ $AC \sqcup BC = Co$ $AC \sqcap BC \sqsubseteq \perp$	$FP \sqcap \neg \forall TC.AC(mary)$, $FP(john)$, $TC(john, ai)$, $St(paul)$, $AC(ai)$, $Tp(kr)$, $Tp(lp)$.
Datalog	$curr(?X, ?Z) :- exam(?X, ?Y), subject(?Y, ?Z)$ $\quad \& St(?X), Co(?Y), Tp(?Z).$ $mayDoThesis(?X, ?Y) :- curr(?X, ?Z), expert(?Y, ?Z)$ $\quad \& St(?X), Tp(?Z), FM \sqcap \exists TC.AC(?Y).$ $mayDoThesis(?X, ?Y) :- \& St(?X), NFP(?Y).$	$exam(paul, ai)$, $subject(ai, kr)$, $subject(ai, lp)$, $expert(john, kr)$, $expert(mary, lp)$.

The query that *paul* may do the thesis with *mary* is derivable, and our algorithm performs the folding results, e.g., $(FM \sqcap \exists TC.AC) \sqcup NFP(mary)$, collecting the disjunction of DL queries from two hybrid rules for *mayDoThesis*.

We remark that “ $mayDoThesis(?X, ?Y) :- \& St(?X), NFP(?Y).$ ”, lacking a Datalog body, does not strictly obey the DL safeness condition, but also has no pure-DL variables. Actually, making AL-log rules strongly DL safe would be possible. For instance, the extended rule is “ $mayDoThesis(?X, ?Y) :- \mathcal{O}(?X), \mathcal{O}(?Y) \& St(?X), NFP(?Y).$ ” where \mathcal{O} is a special Datalog predicate, and for each named individual c a fact $\mathcal{O}(c)$ is added.

4.2. Examples from CARIN

Here (Table 3), in the second hybrid rule of “price”, the variable $?Z$ is a pure-DL variable, such that $associate(?Y, ?Z)$ and $american(?Z)$ have a folding result of $\exists associate.american(?Y)$. Under the Datalog safeness condition, we still obtain the desirable answer to “price(a, usa, high)”, because of $no-fellow-company \sqcup \exists associate.american(b)$.

TransitiveProperty(sameGroup). However, it requires to extend $\mathcal{ALCN}\mathcal{R}$ DL with property hierarchies \mathcal{H} and transitive properties R_+ , i.e., $\mathcal{ALC}_{R+}\mathcal{HN}$ also called as \mathcal{SHN} , and it runs well in our system whose bottom line is \mathcal{SHIQ} covering \mathcal{SHN} . As a result, “TaxLaw(c3, USA, Domestic)” is derivable in this revised \mathcal{SHN} DL without recursive Datalog rules. Actually, this query includes a folding result for checking $c3 \in (\exists \text{associate}^- . \text{american-associate}) \sqcup (\exists \text{sameGroup}^- . (\text{foreign-associate} \sqcap \text{conglomerate}))$, where sameGroup is transitive and $\text{associate} \sqsubseteq \text{sameGroup}$.

4.3. Examples DL-safe Rules

In Table 5, the hybrid rule of “BadChild” has a variable ?Y as the pure-DL variable, and the binary DL queries of $\text{parent}(?X, ?Y)$ and $\text{parent}(?Z, ?Y)$ are chained by this ?Y, conflicting with our definition of independent properties. That means, our system fails to infer “BadChild(Cain)” under the Datalog safeness condition, lacking of expressions for nominals such as $\exists \text{parent} . \exists \text{parent}^- . \{\text{Abel}\}(\text{Cain})$.

Table 5: From DL-safe rules (Datalog safeness condition)

$\mathcal{SHOIQ}(\mathbf{D})$ DL	Person $\sqsubseteq \exists$ father.Person $\exists \text{father} . (\exists \text{father} . \text{Person}) \sqsubseteq \text{Grandchild}$ father \sqsubseteq parent	Person(Cain) father(Cain, Adam) father(Abel, Adam)
Datalog	BadChild(?X) :- hates(?X, ?Z) & Grandchild(?X), parent(?X, ?Y), parent(?Z, ?Y).	hates(Cain, Abel)

To make this hybrid rule strongly DL safe, we rewrite it as follows
BadChild(?X) :- hates(?X, ?Z), $\mathcal{O}(?Y)$

& Grandchild(?X), parent(?X, ?Y), parent(?Z, ?Y).

together with three facts, $\mathcal{O}(\text{Adam})$, $\mathcal{O}(\text{Abel})$, and $\mathcal{O}(\text{Cain})$, added to the initial Datalog. Now, “BadChild(Cain)” is derivable, since “Adam” has been explicitly defined as the father of both “Cain” and “Abel”. However, for another DL assertion such as $\exists \text{father} . \exists \text{father}^- . \{\text{Remus}\}(\text{Romulus})$, this revised strong DL-safe version does not work, due to a failure of matching to that unknown father of “Remus” and “Romulus”.

4.4. Use Case of RuleML FOAF

This use case extends the factual FOAF (Friend-Of-A-Friend) vocabulary with person-centric rules (Li *et al.* 2006a,b). Table 6 shows a Datalog^{DL} KB, whose DL component replies on a \mathcal{SHIQ} DL. It specifies in the DL component that: (1) Every FOAF person knows another FOAF person; (2) A FOAF ‘fan’ is a FOAF person and all of those who he/she knows well are FOAF persons; (3) A FOAF ‘star’ is a FOAF person and is known by at least two FOAF persons; (4) The property *isKnownBy* is the inverse of *knows*; (5) *knowsWell* is a subproperty of *knows*. In the rule component, we have: (1) A person is ‘close’ to the FOAF community if he/she knows a FOAF person; (2) A person ‘possibly knows’ another person if the former is close to the FOAF community and the latter is a FOAF star.

This didactic example, adapted from the RuleML FOAF use case (Li *et al.* 2006b), attempts to employ the expressivity of \mathcal{SHIQ} DL as much as possible, e.g., inverse properties and qualified number restrictions. However, the DL property *knows* cannot be asserted as transitive, because expressive DL like \mathcal{SHIQ} only permits the simple property (i.e., neither itself nor its subproperty is transitive) in qualified number restrictions. In our case, *isKnownBy*, being inverse of *knows*, has a qualified number restriction to two FOAF persons. As a result, it infers that “possiblyKnows(Laura, Ben)”, which equals to the conjunctive queries of $\exists \text{knows} . \text{FOAFPerson}(\text{Laura})$ and $\text{FOAFStar}(\text{Ben})$. In fact, from the DL

Table 6: From RuleML FOAF (Datalog safeness condition)

<i>SHIQ</i> DL	FOAFPerson $\sqsubseteq \exists$ knows.FOAFPerson FOAFFan = FOAFPerson $\sqcap \forall$ knowsWell.FOAFPerson FOAFStar = FOAFPerson $\sqcap \geq 2$ isKnownBy.FOAFPerson isKnownBy = knows ⁻ knowsWell \sqsubseteq knows knowsWell(Barbara, Vivian), isKnownBy(Vivian, Laura), knowsWell(Barbara, Ben), knows(Vivian, Ben), FOAFFan(Barbara).
Datalog	close2FOAF(?X) :- & knows(?X, ?Y), FOAFPerson(?Y). possiblyKnows(?X, ?Y) :- close2FOAF(?X) & FOAFStar(?Y).

component, Barbara is a FOAF fan who knows well Vivian and Ben, s.t. Vivian and Ben are FOAF persons. So, Laura is close to the FOAF community, because she knows a FOAF person, namely Vivian. And another FOAF person, Ben, is a FOAF star, since Ben is known by Barbara and Vivian, both of whom are FOAF persons. Consequently, Laura possibly knows Ben.

5. RE-OBTAINING DECIDABILITY

As pointed in CARIN, the problem of determining whether $\mathcal{K} \models q$ is undecidable, where \mathcal{K} is an unrestricted Datalog ^{\mathcal{L}} KB with recursive Datalog rules, and its \mathcal{L} -based DL component allows arbitrary inclusion statements while \mathcal{L} itself includes only the constructor $\exists P.C$. In short, the recursive Datalog rules extended with cyclic TBox including only one DL constructor of $\exists P.C$ will destroy decidability, while $\exists P.C$ is the most basic DL constructor, introduced first by the simpler \mathcal{ALC} DL. This theorem has been proved in (Levy and Rousset 1998), by reducing the halting problem of a Turing machine to the entailment problem of \mathcal{K} , and the following statements are abstracted from the proof.

DL ABox: integer(1)

DL TBox: integer $\sqsubseteq \exists$ succ.integer

rule-primitive: lessThan(x, y) :- & succ(x, y).

rule-recursive: lessThan(x, y) :- lessThan(z, y) & succ(x, z).

Below, we identify two ways of restricting the expressivity in the KB as to re-obtain a decision procedure, where the first one is in the view of DL and the second is of rules.

(1) To remove some DL constructors: Not obtaining the benefits from the current mature DL techniques as much as possible, we backtrack to the systems of nearly 10 years ago – actually, CARIN has a (maximal) decidable sublanguage, namely CARIN-MARC, which includes the constructors $\sqcap, \sqcup, (\geq nR), \exists R.C$ and negation on primitive classes, with the terminology consisting of acyclic class definitions (i.e., no inclusions or property definitions). DLP has another solution: it requires that the existential DL constructor of $\exists P.C$ can only occur on the left hand side of an inclusion axiom, that is, it allows the form of being $\exists P.C \sqsubseteq D$ but disallows that of $D \sqsubseteq \exists P.C$.

(2) To enforce stronger safeness conditions: Generally speaking, rules are required to be safe, i.e., a variable that appears in the head must also appear in the body – we call it as the Datalog safeness condition in this paper, and the above undecidable encoding is such a case. As mentioned in Table

1, CARIN, DLP and SWRL obey this Datalog safeness, but either CARIN or DLP has its respective restrictions under other considerations as to obtain decidability, while SWRL admits itself undecidable. For the other systems, DL safeness condition has to be emphasized, such as r-hybrid KBs and KAON2 (demanding that “x” must occur in “lessThan(z, y)” for the rule-recursive statement, given our above KB example); moreover, AL-log only permits DL queries to classes without admission to DL properties. Regarding our proposal of Datalog^{DL}, the Datalog safeness conditions are fine, but the above rules will obtain such DL queries as “succ(x, z), succ(z, y)” provided by “lessThan(x, y)” with length of two steps. Here, no independent properties are guaranteed, due to sharing the pure-DL variable of “z”, such that a folding result like $\exists \text{succ. } \exists \text{succ. } \{y\}(x)$ will be submitted to a DL reasoner. Considering that it lacks full provision to the nominals \mathcal{O} in existing DL systems, and our framework conforms to the available techniques, we exclude the above hybrid rules with requirement of independent properties. Thus, we also define some expressivity restrictions to avoid undecidability, driven by considerations to existing DL reasoners rather than other stronger safeness conditions. Actually, for simplicity, we deal little with the recursive rules in our prototype system (Mei 2005), but having been scoped in our ongoing work, this aspect will be paid more attention.

6. CONCLUSION

AL-log has combined Datalog with \mathcal{ALC} , which is obtained as Datalog ^{\mathcal{ALC}} in our proposal. To provide an efficient tool in practice and a sound as well as complete system in theory, our Datalog^{DL} permits any sublanguage \mathcal{L} of \mathcal{SHIQ} as its parameter, obtaining the general Datalog ^{\mathcal{L}} . Existing practical SLD-resolution and DL-tableaux algorithms work well in our combined framework, extending what AL-log has begun. As in CARIN, both class and property predicates are allowed in DL queries, with Datalog safeness conditions instead of AL-log, \mathcal{DL} +log or DL ones. Unique to Datalog ^{\mathcal{L}} is the admission of independent properties in hybrid rules, which enables reasoning support by existing DL reasoners. Unlike CARIN, which prefers forward chaining for modeling an entailment completion, our prototype system (Mei 2005) performs query answering via backward chaining as an extension to a rule engine (e.g., OO jDREW). This makes the hybrid rules processable while keeping the DL reasoner (e.g., RACER) unchanged, acting as an external service. We found that such an architecture is more intuitive to typical users since the non-trivial DL algorithms are just called as a black box.

Our folding technique is also related to ‘rolling-up’ in (Horrocks and Tessaris 2002). Through rolling-up, (conjunctive) queries to the ABox of a DL KB, perhaps containing variables in DL classes or DL properties, can be rewritten s.t. query answering is reduced to the problem of knowledge base satisfiability. Our folding employs the same kind of technique to bridge the gap between query answering in hybrid rules and satisfiability testing in the DL component. Furthermore, the usage of our “independent properties” to some extent corresponds to a particular case of tree-shaped (or acyclic) DL queries as described in (Horrocks and Tessaris 2002).

Besides, we have investigated DL query languages in support of hybrid rules on the practical level. The expressivity and reasoning power of Datalog^{DL} were explored with a suite of previous examples from AL-log, CARIN, DL-safe rules, and our use case RuleML FOAF (Li *et al.* 2006a,b). This suite covers much of the expressiveness discussed for hybrid rules, e.g., in connection with the RIF-OWL compatibility requirement of the W3C Rule Interchange Format (Boley and Kifer 2007). The entire suite is available as implemented test cases in our hybrid rule engine (Mei 2005) coupling OO jDREW with RACER. Although RIF started as a Horn rule language, it may need a Datalog sublanguage which would enable “decidable OWL Compatibility” as studied with Datalog^{DL}.

For the serialization of Datalog^{DL} hybrid rules, the RuleML and RIF <Implies> element with its <then> role for $h(\vec{u})$ and its <if> role for the $b_i(\vec{v}_i)$ conjunction can be extended by a <call> role for the $q_j(\vec{w}_j)$ conjunction. The <call> part of a rule may also be generally used to query other (non-DL) external decidable provers.

However, in this paper, we enriched rules with information from ontologies, but not vice versa. Sharing common predicates between both components is an attractive alternative the challenges of which, such as decidability, were recently taken up in our $\mathcal{ALC}_{\mathbb{P}}^u$ (Mei *et al.* 2007) research. Also, Datalog^{-V} was investigated in HEX-programs and $\mathcal{DL}+\log$ systems as a more expressive rule component; such rules with disjunction and negation are also considered in our future work.

References

- BAADER, F., CALVANESE, D., MCGUINNESS, D., NARDI, D., and PATEL-SCHNEIDER, P. 2003. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press.
- BALL, M., BOLEY, H., HIRTLE, D., MEI, J., and SPENCER, B. 2005. The OO jDREW Reference Implementation of RuleML. In Proceedings of the 1st International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML), LNCS 3791, 218–223. Springer.
- BOLEY, H. and KIFER, M. 2007. RIF Core Design. W3C Working Draft 30 March 2007. <http://www.w3.org/TR/rif-core/>
- DANTSIN, E., EITER, T., GOTTLOB, G., and VORONKOV, A. 2001. Complexity and Expressive Power of Logic Programming. ACM Computing Surveys, 33(3), 374–425.
- DONINI, F. M., LENZERINI, M., NARDI, D., and SCHAERF, A. 1998. AL-log: Integrating Datalog and Description Logics. Journal of Intelligent Information Systems (JIIS), 10(3), 227–252.
- EITER, T., LUKASIEWICZ, T., SCHINDLAUER, R., and TOMPITS, H. 2004. Combining Answer Set Programming with Description Logics for the Semantic Web. In Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR), 141–151. AAAI Press.
- EITER, T., IANNI, G., SCHINDLAUER, R., and TOMPITS, H. 2006. Effective Integration of Declarative Rules with External Evaluations for Semantic-Web Reasoning. In Proceedings of 3rd European Semantic Web Conference, LNCS 4011, 273–287. Springer.
- GROSOFF, B. N., HORROCKS, I., VOLZ, R., and DECKER, S. 2003. Description Logic Programs: Combining Logic Programs with Description Logic. In Proceedings of the 12th International World Wide Web Conference (WWW), 48–57. ACM.
- HAARSLEV, V. and MOLLER, R. 2001. RACER System Description. In Proceedings of the 1st International Joint Conference on Automated Reasoning (IJCAR), LNCS 2083, 701–706. Springer.
- HORROCKS, I. and TESSARIS, S. 2002. Querying the Semantic Web: A Formal Approach. In Proceedings of the 1st International Semantic Web Conference (ISWC), LNCS 2342, 177–191. Springer.
- HORROCKS, I., SATTLER, U., and TOBIES, S. 1999. A Description Logic with Transitive and Converse Roles, Role Hierarchies and Qualifying Number Restrictions. LTCS-Report 99-08, RWTH Aachen, Germany.
- HORROCKS, I., PATEL-SCHNEIDER, P. F., BECHHOFFER, S., and TSARKOV, D. 2005a. OWL Rules: A Proposal and Prototype Implementation. Journal of Web Semantics, 3(1), 23–40.

- HORROCKS, I., PARSIA, B., PATEL-SCHNEIDER, P. F., and HENDLER, J. A. 2005b. Semantic Web Architecture: Stack or Two Towers? In Proceedings of the 3rd International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR), LNCS 3703, 37–41. Springer.
- KIFER, M., DE BRUIJN, J., BOLEY, H., and FENSEL., D. 2005. A Realistic Architecture for the Semantic Web. In Proceedings of the 1st International Conference on Rules and Rule Markup Languages for the Semantic Web (RuleML), LNCS 3791, 17–29. Springer.
- LEVY, A. Y. and ROUSSET, M.-C. 1998. Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence*, 104(1-2), 165–209.
- LI, J., BOLEY, H., BHAVSAR, V. C., and MEI, J. 2006a. Expert Finding for eCollaboration Using FOAF with RuleML Rules. In Proceedings of the Montreal Conference on eTechnologies, 53–65.
- LI, J., BOLEY, H., BHAVSAR, V. C., HIRTLE, D., and MEI, J. 2006b. RuleML FOAF. <http://www.ruleml.org/usecases/foaf>.
- LLOYD, J. W. 1987. *Foundations of Logic Programming* (second, extended edition). Springer-Verlag.
- MEI, J. 2005. Hybrid Rules in OO jDREW. <http://www.jdrew.org/ojdrew/extra/hybridrules.html>.
- MEI, J., BOLEY, H., LI, J., BHAVSAR, V. C., and LIN, Z. 2006. Datalog^{DL}: Datalog Rules Parameterized by Description Logics. In *Canadian Semantic Web*, volume 2 of *Semantic Web and Beyond*, 171–188.
- MEI, J., LIN, Z., and BOLEY, H. 2007. $\mathcal{ALC}_{\mathbb{P}}^u$: An Integration of Description Logic and General Rules. In Proceedings of the First International Conference on Web Reasoning and Rule Systems (RR 2007), LNCS 4524. Springer.
- MOTIK, B., SATTLER, U., and STUDER, R. 2005. Query Answering for OWL-DL with rules. *Journal of Web Semantics*, 3(1), 41–60.
- OWL, W3C. 2004. OWL: Web Ontology Language Semantics and Abstract Syntax. <http://www.w3.org/TR/owl-absyn/>.
- RIF, W3C. 2005. Rule Interchange Format Working Group. <http://www.w3.org/2005/rules/wg.html>.
- ROSATI, R. 2005. On the Decidability and Complexity of Integrating Ontologies and Rules. *Journal of Web Semantics*, 3(1), 61–73.
- ROSATI, R. 2006. DL+log: Tight Integration of Description Logics and Disjunctive Datalog. In Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR), 68–78. AAAI Press.