Datalog^{DL}: Datalog Rules Parameterized by Description Logics

Jing Mei[1], Harold Boley[3], Jie Li[2,3], Virendrakumar C. Bhavsar[2], Zuoquan Lin[1]

[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]Institute for Information Technology - e-Business

National Research Council of Canada

Fredericton, NB, E3B 9W4, Canada

{Harold.Boley, Jie.Li} AT nrc.gc.ca

Abstract

Combining ontologies with 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 $\mathcal{ALC}$ to $\mathcal{SHIQ}$. 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 integrating a rule engine (OO jDREW) with a DL reasoner (RACER).

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 [20]. Whether, in the Semantic Web's layered structure, there should be only one homogeneous hierarchy for ontologies and rules [13], or these should stand heterogeneously (hybridly) side by side under a logic framework [10], the combination of ontologies and rules, within a practical and feasible framework, is an interesting topic deserving more investigation.

Description logics (DLs) have been recognized as the logical foundation of ontologies in the Semantic Web, and the Web Ontology Language, namely OWL [18], has two species: OWL-Lite and OWL-DL, closely related to the DL languages $\mathcal{SHIQ}(\mathrm{D})$ and $\mathcal{SHOIN}(\mathrm{D})$, respectively. On the other hand, Datalog is a wide-spread rule-based language, even popular in the industry. That is, both of these two knowledge representations have reached a certain level of maturity, which make them suitable candidates for combination.

Among the integration frameworks for combining rules and DLs (see Table 1), one is the homogeneous approach (like DLP [8], SWRL [11], and KAON2 [17]), while the other is the hybrid approach (like AL-log [6], CARIN [14], dl-programs [7], and r-hybrid KBs [19]). However, there exists the usual trade-off between the expressivity of languages and the complexity of their reasoning services.

Table 1. Comparison of Approaches

		Safeness Condition		Information Flow		Strategy
		Strong	Weak	Uni-directional	Bi-directional	
Homogeneous Approach	DLP		X		X	Reduction
	SWRL		X		X	–
	KAON2	X			X	Reduction
Hybrid Approach	AL-log	X		X		SLD-resolution
	CARIN		X	X		Entailment
	dl-programs	X			X	Fixpoint iteration
	r-hybrid KBs	X			X	–
	Datalog^{DL}		X	X		SLD-resolution

AL-log, the earlier and simpler case, integrates 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 those of $\mathcal{ALC}$ and Datalog. But, the binary predicates (i.e., 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 a *strong* safeness condition, and a formal definition is presented below), s.t. only unary predicates without variables (i.e., ground classes) will be submitted to the DL tableaux reasoner.

More generally, CARIN is a family of languages, each of which combines (a sublanguage of) $\mathcal{ALCNR}$ 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. Besides, CARIN allows ground or open DL-queries with unary and binary predicates, and the variables appearing in the head of a rule are required also to appear in the body but not necessary of being in the DL body (we call this a *weak* safeness condition, and a formal definition is presented below) -- this is a general safeness condition for rule-based languages, weaker than that of AL-log. As to non-recursive CARIN-$\mathcal{ALCNR}$, a sound and complete inference procedure has been established, while reasoning in recursive CARIN-$\mathcal{ALCNR}$ is un-decidable, and there are two ways of restricting expressivity to regain soundness and completeness: one is to remove some DL constructors and allow an acyclic terminology only, and the other is to make the safeness condition strong.

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 and r-hybrid KBs, 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.

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. Also, a strong safeness condition, similar to the one in AL-log and r-hybrid KBs, is required by KAON2, where this restriction covers some of the common usages of DL expressivity.

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-query 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 existence restrictions) and uncertain assertions (e.g., derived from disjunction 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 reasonable Semantic Web community by employing existing techniques as much as possible, we attempt to balance the trade-off of the expressivity and the reasoning power, and consider $\mathcal{SHIQ}$ as our bottom line, whose practical and efficient tools are available (such as RACER [9]). Here, we adopt the weak safeness condition, and the problems introduced by the *pure-DL* variables in DL-queries, beyond the strong safeness condition, will be handled cautiously, provided that those expressive statements would be kicked out by the bottom line of $\mathcal{SHIQ}$ DL. By defining *independent* properties, we clarify our current reasoning services: hybrid rules with DL-query to classes and independent properties in weak safeness condition are fully supported.

As a result, this paper presents Datalog^{DL} as a family of hybrid representation languages, where Datalog rules are parameterized by a specific DL language L, namely Datalog^{L}, where L ranges from $\mathcal{ALC}$ to $\mathcal{SHIQ}$. On the theoretical side, we show a sound and complete algorithm for reasoning in Datalog^{DL}, with the complexity of ExpTime in any case of its parameterized DL language L. On the practical side, while keeping a DL reasoner unchanged, a typical rule engine (e.g., OO jDREW [4]) 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 external DL reasoner (e.g., RACER).

Next, Datalog^{DL} will be introduced in section 2 with its syntax and semantics, while its reasoning will be described in section 3 together with proofs of soundness and completeness. Section 4 is meant to clarify technical problems of decidability underlying in hybrid rules, and finally conclusions are drawn in section 5.

2. The Datalog^{DL} Languages

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 some variety of DL-query.

Consider the main layers of the DL family bottom-up [3], $\mathcal{ALC}$ is a basic and simple language, permitting class descriptions via $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, and $\exists R.D$ where C, D are classes and R is a property. Augmented by transitive properties, $\mathcal{ALC}$ becomes $\mathcal{ALC}_{\mathcal{R}^+}$ 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 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, we introduce the syntax and semantics for Datalog^{DL} with no need for preliminaries. However, we adopt the so-called *unique named assumption* (UNA), a convention of Datalog not normally used by DLs.

2.1 Syntax

In order to preserve decidability, we fix the rule language to Datalog, so that terms must be variables or constants. Undecidable extensions to Horn logic, where terms can also be function applications, have been considered as well, but are beyond the scope of this paper.

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

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

$$h(X) :- b_1(Y_1), \cdots, b_m(Y_m) \,\&\, q_1(Z_1), \cdots, q_n(Z_n)$$

where X, Y_1, ..., Y_m are n-ary sequences of terms while Z_1, ..., Z_n are unary/binary sequences of terms, h(X) and $b_i(Y_i)$ ($1 \leqslant i \leqslant m$) are Datalog atoms with predicates in A_P while each $q_j(Z_j)$ ($1 \leqslant j \leqslant n$) is a DL-query with a predicate in A_T.

Two safeness conditions are introduced for hybrid rules:
Weak safeness: a variable occurring in X must occur in one of the $Y_i|Z_j$'s.
Strong safeness: a variable occurring in r must occur in one of the Y_i's.

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 safe has been introduced in [17], that is: (1) For each rule r whose variable w does not occur in any of the Y_i's, we add an atom O(w) to the Datalog body of r, where $\mathcal{O}$ is a special predicate symbol, $O \in A_P$; (2) For each constant c occurring in K = (Σ,Π), we add a fact O(c) to Π.

As mentioned in section 1, we prefer to the weak safeness condition rather than the strong one. Below, *pure-DL* variables are defined.

Definition 2. A pure-DL variable in a rule r is a variable that only occurs in one of the Z_j's.

Pure-DL variables lead to the violation of the strong safeness condition in cases where the weak safeness condition is obeyed. Note that, without the presence of pure-DL variables (i.e., under the strong safeness condition), our system appears to be Datalog extended with ground DL-queries, which is a simple and straightforward extension to AL-log.

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 [12]. 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 non-trivial given that the negation of properties is not supported by most DLs. Hence, a candidate technique is *folding* (called *rolling-up* in [12]), whose objective is to eliminate properties from 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. $P(u, x) \wedge Q(v, x)$). Tree-shaped DL queries appear to be a solution to this problem by exploiting the tree model property of the DL [12]; however, the undecidability of an unrestricted combination of DLs with rules is exactly due to the fact that adding rules to DLs causes the loss of any form of tree model property [17]. Hence, strong safeness is imposed by DL-safe rules [17] and other approaches [6][7][19], 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 strong safeness condition, γ being its head, α being its Datalog body, and β being its DL body. Specifically, it has the form γ:-α&β, where β contains a pure-DL variable x having a class description C (C can be the DL top class). 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)∧C(x) or P(x, u)∧C(x) becomes its *folding* result ∃P.C(u) or ∃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)∧Q(v, x)∧C(x) becomes
   $\exists P.(\exists Q^{-}.\{v\} \sqcap C)(u)$ or $\exists Q.(\exists P^{-}.\{u\} \sqcap C)(v)$
   (b) P(u, x)∧Q(x, v)∧C(x) becomes
   $\exists P.(\exists Q.\{v\} \sqcap C)(u)$ or $\exists Q^{-}.(\exists P^{-}.\{u\} \sqcap C)(v)$
   (c) P(x, u)∧Q(v, x)∧C(x) becomes
   $\exists P^{-}.(\exists Q^{-}.\{v\} \sqcap C)(u)$ or $\exists Q.(\exists P.\{u\} \sqcap C)(v)$
   (d) P(x, u)∧Q(x, v)∧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 N and inverse properties I makes the complexity jump from ExpTime (for $\mathcal{SHIN}$) to NExpTime (for $\mathcal{SHOIN}$). Although the operator {u} could be 'simulated' by its *representative* concept C_u [12], we still focus on cases 1 and 2 in this paper, not introducing different fresh concept names for different individuals. Another consideration is 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 $\leftarrow G_1 \wedge ... \wedge G_n$ with variables $y_1,...,y_m$. Showing that the set of clauses $\Pi \cup \{G\}$ is unsatisfiable is exactly the same as showing that $\exists y_1 ... \exists y_m (G_1 \wedge ... \wedge G_n)$ 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.

2.2 Semantics

The semantics of Datalog^{DL} derives in a natural way from the semantics of its component languages, based on the first-order semantics. As follows, we define an interpretation and a model of our language Datalog^{L}, including the satisfying conditions for ground Datalog atoms, ground DL-queries, and hybrid rules. We direct readers to the description logic handbook [3] and the foundations of logic programming [16] for those related definitions.

Definition 4. An interpretation $I = (\triangle, \bullet^I)$ of a language Datalog^{L} consists of the following: (1) A nonempty domain $\triangle$; (2) For each constant a in C, the assignment of an element in $\triangle$, i.e.,$a^I \in \triangle$; (3) For each n-ary predicate p in the alphabet of predicates $A = A_T \cup A_P$, the assignment of a relation of arity n over the domain $\triangle$, i.e., a relation on $\triangle^n$.

Definition 5. Let I be an interpretation for a language Datalog^{L}, and for a given hybrid rule r,

A variable assignment V_r w.r.t I is an assignment to each variable in r of an element in the domain of I.

A term assignment T_r w.r.t I is defined: (1) Each variable is given its assignment according to V_r; (2) Each constant is given its assignment according to I.

Definition 6. Let I be an interpretation for a language Datalog^{L}. (1) A ground Datalog atom $\alpha = p(C)$, $p \in A_P$, is satisfied by I if $C^I \in p^I$, written as $I \models \alpha$. (2) A ground DL-query $\beta = q(C)$, $q \in A_T$, is satisfied by I if $C^I \in q^I$, written as $I \models \beta$. (3) A hybrid rule r that $h(X) :- b_1(Y_1), \cdots, b_m(Y_m) \,\&\, q_1(Z_1), \cdots, q_n(Z_n)$ is satisfied by I if, whenever T_r is a term assignment w.r.t I, such that $T_r(Y_i) \in b_i^I$ and $T_r(Z_j) \in q_j^I$ ($1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$) for every atom in the body of r, then $T_r(X) \in h^I$ for the head of r, written as $I \models r$.

Definition 7. Let I be an interpretation for a language Datalog^{L}. I is a model of the Datalog^{L} knowledge base (Σ,Π), consisting of a Datalog program Π with DL-queries to Σ, if I satisfies each hybrid rule in Π and I is a model of Σ according to the description logic L.

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, so that backward chaining plays the role of our principal reasoning strategy.

3.1 Algorithm

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

Input: Datalog^{L} KB K=(Σ,Π) and a query q.
Output: TRUE if q is satisfied by 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 $D_1 \sqcup \cdots \sqcup D_m$ such that its class descriptions D_i are collected from the involved hybrid rules r_i, where $1 \leqslant i \leqslant m$. For an individual a, the separate DL-queries $D_i(a)$ will be replaced by a single new one, $D_1 \sqcup \cdots \sqcup D_m(a)$. If the DL-query $D_1 \sqcup \cdots \sqcup D_m(a)$ 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 K input obey the restriction of only having independent properties, as imposed by our definition of K, s.t. step-2 produces ground rules under the weak safeness condition. For rules fulfilling the strong safeness condition, step-2 will be skipped due to the non-appearance of pure-DL variables. That is, our algorithm introduces a

method to re-establish strong safeness by eliminating 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) :- & A(x). and C(x) :- & B(x). In addition, there is an individual a in the DL top class $\top$. 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 $\{X_1/t_1, ..., X_n/t_n\}$, 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 $i \in \{1, ..., n\}$. Below is the technical details for query answering, using the notions inherited from AL-log but with extensions to DL properties.

Definition 8 [Constrained SLD-resolution]. Let L be a specific DL language, $K=(\Sigma,\Pi)$ be a Datalog^L knowledge base, $q = \alpha_1,\ldots,\alpha_s \,\&\, \beta_1,\ldots, \beta_t$ be a query to K where α_i is a Datalog atom and β_j is a DL atom, and r be a hybrid rule of the form $\alpha' :- \alpha'_1,\ldots,\alpha'_m \,\&\, \beta'_1,\ldots, \beta'_n$. Suppose θ is the most general substitution such that $\alpha'\theta = \alpha_k\theta$, where α_k is one of $\{\alpha_1,\ldots,\alpha_s\}$. The resolvent of q and r with substitution θ is the query $q'=\mu\&\nu$, where $\mu=(\alpha_1,\ldots,\alpha_{k-1}, \alpha'_1,\ldots,\alpha'_m, \alpha_{k+1},\ldots,\alpha_s)\theta$ and $\nu=(\beta_1,\ldots, \beta_t, \beta'_1,\ldots, \beta'_n)\theta$ with simplification: if there are two constraints of the form t:C, t:D, they are replaced by the equivalent constraint t: $C \sqcap D$.

Definition 9 [Constrained SLD-derivation]. A constrained SLD-derivation for a query q_0 in K is a derivation constituted by:

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

such that for each $i \in \{0, 1, ..., n-1\}$, 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 $q_{DL} = \varnothing \,\&\, \beta_1,\ldots,\beta_l$, which is called *constrained empty clause*. For strong safeness conditions, the constrained empty clause should have not any variable, while for weak safeness conditions, pure-DL variables appear as being existentially quantified in some of "$\beta_1,\ldots,\beta_l$". In this sense, we currently only consider independent properties in hybrid rules, with folding results fully supported by existing DL reasoners.

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

K |- ∅ & C(x) iff C^I is nonempty, where I is the model of K

K |- ∅ & P(u, x) ∧ C(x) iff K |= ∃P.C(u)

K |- ∅ & P(x, u) ∧ C(x) iff K |= $\exists P^-$.C(u)

Definition 10 [Constrained SLD-refutation]. A constrained SLD-refutation for a query q in K is a finite set of constrained SLD-derivations $d_1,...,d_m$ for q in K such that, denoting as $q_0^i,...,q_{ni}^i$ the sequence of queries of the i^{th} derivation d_i, the following conditions hold:

1. For each i, q_{ni}^i is one of the form "∅ & $\beta_1^i,\ldots,\beta_{li}^i$", i.e., the last query of each derivation is a constrained empty clause.
2. For each q_{ni}^i with pure-DL variables, obtain the folding results of q_{ni}^i.
3. For each model I of K, there exists at least one i∈{1,...,m} s.t. I |= q_{ni}^i; we write this condition K |= disj(q_{n1}^1, ..., q_{nm}^m).

We write K |- q, if there is a constrained SLD-refutation for q in K.

Lemma 1. Let q be a ground query to a $Datalog^L$ knowledge base K = (Σ,Π). K |- q if and only if K |= 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.

=>: Suppose K |- q, i.e., the ground query q has a constrained SLD-refutation. Then, for each derivation, if I is a model of 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 I of K satisfies at least one of the constrained empty clauses. Then each model of K satisfies q, that is K |= q.

<=: Suppose K |- q fails, we have no constrained SLD-refutation for q in K, resulting from three possibilities according to Definition 10.

1. If there is no constrained empty clause, then from the completeness of SLD-resolution, we have the failure of K |= 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 K |= q.
3. If there is a model I of K, then for any derivation of q whose last query is a constrained empty clause (written as q_{ni}^i = ∅ & $\beta_1^i,\ldots,\beta_{ni}^i$), it makes I |= q_{ni}^i a failure. That is, there is a model I of Σ such that I |= $\beta_1^i,\ldots,\beta_{ni}^i$ fails. Characterized by I, we can construct another model J, and it can be shown -- by induction on the construction of J -- that J |= q fails, and K |= q fails.

Referring to AL-log, $Datalog^L$ 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 [3]. For the rule component, Datalog is data complete for P while program complete for EXPTIME [5]. As a result,

the computational complexity of Datalog^{L} is ExpTime, where L ranges from $\mathcal{ALC}$ to $\mathcal{SHIQ}$.

Theorem 1. Query answering in Datalog^{L} is a decidable problem in ExpTime.

4. Re-obtaining Decidability

As pointed in CARIN, the problem of determining whether K |= q is undecidable, where K is a Datalog^{L} knowledge base with recursive Datalog rules, and its L-based DL component allows arbitrary inclusion statements while L itself includes only the constructor ∃P.C. In short, the recursive Datalog rules extended with cyclic TBox including only one DL constructor of ∃P.C will destroy decidability, while ∃P.C is the most basic DL constructor, introduced first by the simpler $\mathcal{ALC}$ DL. This theorem has been proved in [14], by reducing the halting problem of a Turing machine to the entailment problem of K. Below, we rewrite them:

- DL ABox: integer(1)
- DL TBox: integer $\sqsubseteq$ ∃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 knowledge base 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 ∃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 weak safeness condition in this paper, and the above undecidable encoding is a case of weakness. As mentioned in Table 1, CARIN, DLP and SWRL obey this weak 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, strong safeness conditions have to be

emphasized, such as r-hybrid KBs and KAON2 (demanding that "x" must occur in "lessThan(z, y)" given our above KB example); moreover, AL-log only permits DL-query to classes without admission to DL properties. Regarding our proposal of Datalog^{DL}, weak 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", s.t. a folding result like ∃succ.∃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 strong safeness conditions. Actually, for simplicity, we deal little with the recursive rules in our prototype system [1], but having been scoped in our ongoing work, this aspect will be paid more attention.

5. Conclusion

AL-log has combined Datalog with $\mathcal{ALC}$, regarded as Datalog^{ALC} in our proposal. To provide an efficient tool in practice and a sound and complete system in theory, our Datalog^{DL} concerns any sublanguage L of $\mathcal{SHIQ}$ as its parameter, namely Datalog^{L}, and the practical SLD-resolution and DL tableaux algorithms act well in an integrated framework, beyond what AL-log has done. Like CARIN, both class and property predicates are allowed in DL-queries, with weak safeness conditions instead of strong ones. And the unique requirement is the admission of independent properties in hybrid rules, which conforms to support for reasoning in existing DL reasoners. Besides, different from CARIN, which prefers to forward chaining for modeling an entailment completion, our prototype system [1] performs query answering in backward chaining with improvements to a rule engine (e.g., OO jDREW), making the hybrid rules processable, while keeping the DL reasoner (e.g., RACER) unchanged to act as an external service. And we assume such adaptation is more straightforward to users that the non-trivial DL algorithms would be regarded as a black box.

It should be pointed out how our folding technique is related to 'rolling-up' in [12]. There, (conjunctive) queries to the ABox of a DL knowledge base, 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. Here, this kind of technique is used to bridge the gap

between query answering in hybrid rules and testing satisfiability 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 [12].

We are currently investigating 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 [15]. This suite covers much of the expressiveness currently discussed for hybrid rules, e.g. in the W3C RIF WG [20]. The entire suite is implemented in our hybrid rule engine [1] coupling OO jDREW with RACER.

For the serialization of hybrid rules, the RuleML <Implies> element with its <head> role for h(X) and its <body> role for the $b_i(Y_i)$ can be extended with a <neck> role for the $q_j(Z_j)$. The *neck* of a rule may also be generally used to query other (non-DL) external decidable provers.

In this paper, we enriched rules with information from ontologies, but not vice versa. Sharing common predicates in both components is attractive, while the problems it causes, such as decidability, are open challenges for the Semantic Web. Also, $\text{Datalog}^{\neg\vee}$ was investigated in dl-programs and r-hybrid systems as a more expressive rule component; such rules with disjunction and negation are also considered in our future work.

Acknowledgements

We would like to thank the anonymous CSWWS2006 reviewers for their helpful suggestions.

References

1. Jing Mei (2005) *Hybrid Rules in OO jDREW*. http://www.jdrew.org/oojdrew/exa /hybridrules.html.
2. Grigoris Antoniou, Carlos Viegas Damasio, Benjamin N. Grosof, Ian Horrocks, Michael Kifer, Jan Maluszynski, and Peter F. Patel-Schneider (2005) *Combining Rules and Ontologies - A survey*. http://rewerse.net/deliverables /m12/i3-d3.pdf.
3. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider (2003) *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.
4. Marcel Ball, Harold Boley, David Hirtle, Jing Mei, and Bruce Spencer (2005) *The OO jDREW Reference Implementation of RuleML*. In: International

Conference on Rules and Rule Markup Languages for the Semantic Web, pp 218--223.
5. Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov (2001) *Complexity and Expressive Power of Logic Programming*. ACM Computing Surveys, pp 374--425.
6. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf (1998) *AL-log: Integrating Datalog and Description Logics*. Journal of Intelligent Information Systems, pp 227-252.
7. Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits (2004) *Combining Answer Set Programming with Description Logics for the Semantic Web*. In: The Ninth International Conference on the Principles of Knowledge Representation and Reasoning, pp 141-151.
8. Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker (2003) *Description Logic Programs: Combining Logic Programs with Description Logic*. In: The Twelfth International World Wide Web Conference, pp 48--57.
9. Volker Haarslev and Ralf Mller (2001) *RACER System Description*. In: International Joint Conference on Automated Reasoning, pp 701--706.
10. Ian Horrocks, Bijan Parsia, Peter F. Patel-Schneider, and James A. Hendler (2005) *Semantic Web Architecture: Stack or Two Towers?* In: Workshop on Principles and Practice of Semantic Web Reasoning, pp 37-41.
11. Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin N. Grosof, and Mike Dean (2004) *Semantic Web Rule Language*. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.
12. Ian Horrocks and Sergio Tessaris (2002) *Querying the Semantic Web: a Formal Approach*. In: Workshop on Principles and Practice of Semantic Web Reasoning, pp 177--191.
13. Michael Kifer, Jos de Bruijn, Harold Boley, and Dieter Fensel (2005) *A Realistic Architecture for the Semantic Web*. In: International Conference on Rules and Rule Markup Languages for the Semantic Web, pp 17-29.
14. Alon Y. Levy and Marie-Christine Rousset (1996) *CARIN: A Representation Language Combining Horn Rules and Description Logics*. In: The Twelfth European Conference on Artificial Intelligence, pp 323-327.
15. Jie Li, Harold Boley, Virendrakumar C. Bhavsar, and Jing Mei (2006) *Expert Finding for eCollaboration Using FOAF with RuleML Rules*. In: The Montreal Conference on eTechnologies, May 2006. To Appear.
16. John W. Lloyd (1987) *Foundations of Logic Programming (second, extended edition)*. Springer series in symbolic computation.
17. Boris Motik, Ulrike Sattler, and Rudi Studer (2005) *Query Answering for OWL-DL with Rules*. Journal of Web Semantics, pp 41-60.
18. Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks (2004) *OWL Web Ontology Language Semantics and Abstract Syntax*. http://www.w3.org/TR/owl-absyn/.
19. Riccardo Rosati (2005) *On the decidability and complexity of integrating ontologies and rules*. Journal of Web Semantics, pp 61-73.
20. W3C (2005) *Rule Interchange Format Working Group*. http://www.w3.org/2005/rules/wg.html.