From: AAAI-96 Proceedings. Copyright © 1996, AAAI (www.aaai.org). All rights reserved.

The Limits on Combining Recursive Horn Rules with Description Logics

Alon Y. Levy
AI Principles Research Department
AT&T Research
levy@research.att.com

Marie-Christine Rousset
L.R.I. U.R.A C.N.R.S
University of Paris-Sud, France
mcr@lri.lri.fr

Abstract

Horn rule languages have formed the basis for many Artificial Intelligence application languages, but are not expressive enough to model domains with a rich hierarchical structure. Description logics have been designed especially to model rich hierarchies. Several applications would significantly benefit from combining the expressive power of both formalisms. This paper focuses on combining recursive function-free Horn rules with the expressive description logic $\mathcal{ALCNR}$, and shows exactly when a hybrid language with decidable inference can be obtained. First, we show that several of the core constructors of description logics lead by themselves to undecidability of inference when combined with recursive function-free Horn rules. We then show that without these constructors we obtain a maximal subset of $\mathcal{ALCNR}$ that yields a decidable hybrid language. Finally, we describe a restriction on the Horn rules that guarantees decidable inference when combined with all of $\mathcal{ALCNR}$, and covers many of the common usages of recursive rules.

Introduction

Horn rule languages have formed the basis for many Artificial Intelligence applications as well as the basis for deductive and active database models. Horn rules are a natural representation language in many application domains, and are attractive because they are a tractable subset of first order logic for which several practically efficient inference procedures have been developed. One of the significant limitations of Horn rules is that they are not expressive enough to model domains with a rich hierarchical structure. In contrast, description logics are a family of representation languages that have been designed especially to model rich hierarchies of classes of objects. The computational and expressive properties of description logics have been extensively studied, and systems based on these formalisms have recently been used in applications (e.g., [Wright *et al.*, 1993; Wahlster *et al.*, 1993]).

A description logic is a subset of first order logic with equality that contains only unary relations, representing sets of objects in the domain (referred to as *concepts*) and binary relations (called *roles*). A concept describes a set of objects in the domain, and is defined by the necessary and sufficient conditions satisfied by objects in the set. A description logic contains a set of *constructors* that determines the grammar in which the conditions can be specified. Much of the research in description logics has concentrated on algorithms for determining subsumption relations between concepts, and for checking membership of an object in a concept.

Horn rules and descriptions logics are two orthogonal subsets of first order logic [Borgida, 1994]. Several applications (e.g., combining information from multiple heterogeneous sources, modeling complex physical devices) significantly benefit from combining the expressive power of both formalisms. Starting from the KRYPTON language [Brachman *et al.*, 1985], several works have considered the design of hybrid representation languages that combine rules with description logics (e.g., [Frisch, 1991; Donini *et al.*, 1991a; MacGregor, 1994]). Recently, the CARIN family of languages has been developed [Levy and Rousset, 1996], which combines Horn rules and description logics in a more tight way than previous formalisms. CARIN knowledge bases contain a set of rules in addition to concept definitions. The rules in CARIN can have concept and role literals in their antecedents in addition to ordinary literals. In [Levy and Rousset, 1996] a sound and complete inference procedure for *non recursive* Horn rules and the relatively expressive description logic $\mathcal{ALCNR}$ is described. The combination of these two formalisms in one language has been useful for two reasons. First, it enables us to express rich constraints of a concept hierarchy while still having the ability to use predicates of arbitrary arity, and the ability to express arbitrary joins between relations. Second, it provides a *query language* in which arbitrary conjunctive queries (and unions thereof) can be ex-

pressed over description logic knowledge bases. Both of these features were key to the design of the Information Manifold system [Levy *et al.*, 1996] that combines information from multiple heterogeneous sources.

An important question is to what extent these two subsets of first order logic can be combined, while still retaining the ability to perform *sound* and *complete* reasoning. The main question that remains open, and is also important for applications, is the case in which the Horn rules may be recursive but function-free. In this paper we show the *exact* limitations on combining recursive function-free Horn rules with the $\mathcal{ALCNR}$ description logic and its subsets. We begin by showing that several description logic constructors in isolation (role restriction and role-filler cardinality upper bounds) already lead to undecidability of inference when combined with recursive function-free Horn rules. These results are significant because these constructors are the *core* of most description logics. We then consider two ways of restricting the formulas in the knowledge base for which we establish sound and complete inference procedures. The first is a restriction on the description logic. Specifically, we present a maximal subset of $\mathcal{ALCNR}$ that can be combined with any set of recursive function-free Horn rules. The second is a restriction on the occurrences of role atoms in the rules. In particular, it requires that in every role atom at least one variable that appears in the atom also appears in an ordinary (i.e., not role or concept) atom. Importantly, the second restriction covers some of the common usages of recursive rules (e.g., expressing graph connectivity).

The CARIN Languages

CARIN is a family of languages, each of which combines a description logic $\mathcal{L}$ with Horn rules. We denote a specific language in CARIN by CARIN-$\mathcal{L}$. A CARIN-$\mathcal{L}$ knowledge base (KB) contains two components, the first is a terminology and the second is a set of rules and ground facts. The terminology is a set of statements in $\mathcal{L}$ about concepts (unary predicates) and roles (binary predicates) in the domain. Concepts and roles from the terminology can also appear in the antecedents of the Horn rules of the KB in addition to other *ordinary* predicates of arbitrary arity. We describe each component below.

Terminological Component in CARIN

The terminological component of a CARIN-$\mathcal{L}$ knowledge base contains a set of formulas in the description logic $\mathcal{L}$. A description logic contains unary relations (called concepts) which represent sets of objects in the domain and binary relations (called roles) which describe relationships between objects. Expressions in the terminology are built from concept and role names and from concept and role *descriptions*, which denote complex concepts and roles. Descriptions are built from the set of *constructors* of $\mathcal{L}$. The set of constructors vary from one description logic to another, and consequently so does the complexity of reasoning. In this paper we consider CARIN languages in which the description logic component is any subset of the expressive language $\mathcal{ALCNR}$ [Buchheit *et al.*, 1993]. Descriptions in $\mathcal{ALCNR}$ can be defined using the following syntax (A denotes a concept name, P_i's denote role names, C and D represent concept descriptions and R denotes a role description):

$C, D \rightarrow$	$A \mid$	(primitive concept)
	$\top \mid \bot \mid$	(top, bottom)
	$C \sqcap D \mid C \sqcup D \mid$	(conjunction, disjunction)
	$\neg C \mid$	(complement)
	$\forall R.C \mid$	(universal quantification)
	$\exists R.C \mid$	(existential quantification)
	$(\geq n\, R) \mid (\leq n\, R)$	(number restrictions)
$R \rightarrow$	$P_1 \sqcap \ldots \sqcap P_m$	(role conjunction)

The formulas in the terminological component $\Delta_{\mathcal{T}}$ of a CARIN knowledge base Δ are either *concept definitions* or *inclusion statements*. A concept definition is a statement of the form $A := D$, where A is a concept name and D is a concept description. We assume that a concept name appears on the left hand side of at most one concept definition. An inclusion statement is of the form $C \sqsubseteq D$, where C and D are concept descriptions. Intuitively, a concept definition associates a definition with a name of a concept. An inclusion states that every instance of the concept C must be an instance of D.[1] A concept name A is said to *depend* on a concept name B if B appears in the concept definition of A. A set of concept definitions is said to be *cyclic* if there is a cycle in the dependency relation. When the terminology contains only concept definitions and has no cycles we can *unfold* the terminology by iteratively substituting every concept name with its definition. As a result, we obtain a set of concept definitions, where all the concepts that appear in the right hand sides do not appear on the left hand side of any definition.

The semantics of the terminological component are given via *interpretations*. An interpretation I contains a non-empty domain $\mathcal{O}^I$. It assigns a unary relation C^I to every concept in $\mathcal{T}$, and a binary relation R^I over $\mathcal{O}^I \times \mathcal{O}^I$ to every role R in $\mathcal{T}$. The extensions of concept descriptions are given by the following equations: ($\sharp\{S\}$ denotes the cardinality of a set S):

$\top^I = \mathcal{O}^I$, $\bot^I = \emptyset$, $(C \sqcap D)^I = C^I \cap D^I$,
$(C \sqcup D)^I = C^I \cup D^I$, $(\neg C)^I = \mathcal{O}^I \setminus C^I$,
$(\forall R.C)^I = \{d \in \mathcal{O}^I \mid \forall e : (d,e) \in R^I \rightarrow e \in C^I\}$
$(\exists R.C)^I = \{d \in \mathcal{O}^I \mid \exists e : (d,e) \in R^I \wedge e \in C^I\}$
$(\geq n\, R)^I = \{d \in \mathcal{O}^I \mid \sharp\{e \mid (d,e) \in R^I\} \geq n\}$
$(\leq n\, R)^I = \{d \in \mathcal{O}^I \mid \sharp\{e \mid (d,e) \in R^I\} \leq n\}$
$(P_1 \sqcap \ldots \sqcap P_m)^I = P_1^I \cap \ldots \cap P_m^I$

An interpretation I is a model of $\Delta_{\mathcal{T}}$ if $C^I \subseteq D^I$ for every inclusion $C \sqsubseteq D$ in the terminology, and

[1] A concept definition can also be given by two inclusion statements. However, we single out concept definitions here because they will be of special interest later on.

$A^I = D^I$ for every concept definition $A := D$. We say that the concept C *is subsumed by* the concept D w.r.t. $\Delta_\mathcal{T}$ if $C^I \subseteq D^I$ in every model I of $\Delta_\mathcal{T}$.

Example 1: Consider the following terminology $\mathcal{T}_1$:

american-assoc-company := company⊓∃associate.american
foreign-assoc-company := company⊓∃associate.¬american
allied-company := company ⊓ (american ⊔ american-assoc-company)
assoc-company := company ⊓ (≥ 1 associate)
conglomerate := company ⊓ (≥ 2 associate)

company and american are primitive concepts, and associate is a role. An american-assoc-company (respectively, foreign-assoc-company) is defined to be the set of companies that have an associate company that is American (resp. not American). The concept allied-company represents the companies that are either American or have an American associate. The concept assoc-company represents the set of companies that have at least one associate, and conglomerate represents the set of companies with at least 2 associates.

As an example of a subsumption relationship that can be inferred from the terminology, the concept american-assoc-company ⊓ foreign-assoc-company is subsumed by conglomerate. This follows because instances of first description must have one associate that is American and one that is not American, and therefore must have at least two associates. □

Rules and Ground Facts in CARIN

The Horn-rule component $\Delta_\mathcal{R}$ of a CARIN knowledge base Δ contains a set of ground atomic facts and a set of Horn rules that are logical sentences of the form:

$$p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \Rightarrow q(\bar{Y})$$

where $\bar{X}_1, \ldots, \bar{X}_n, \bar{Y}$ are tuples of variables or constants. We require that the rules are safe, i.e., a variable that appears in $\bar{Y}$ must also appear in $\bar{X}_1 \cup \ldots \cup \bar{X}_n$. The predicates $p_1, \ldots, p_n$ may be either concept or role names, or predicates that do not appear in $\Delta_\mathcal{T}$, which are called *ordinary* predicates. Note that ordinary predicates can be of any arity. The predicate q must be an ordinary predicate. It should be noted that CARIN Horn rules are more general than previous languages such as AL-log [Donini *et al.*, 1991a], where in addition to ordinary predicates, only concept predicates were allowed in the rule, and a variable appearing in a concept atom has to appear in an atom of an ordinary predicate in the antecedent. The predicates of the ground facts can be either concept, role or ordinary predicates.

Example 2: In addition to the terminology $\mathcal{T}_1$ we have the following rules and ground facts, $\mathcal{R}_1$:

r_1 : company(X) ∧ associate(X,Y) ⇒ sameGroup(X,Y)
r_2 : sameGroup(X,Z)∧sameGroup(Z,Y) ⇒ sameGroup(X,Y)
r_3 : foreign-assoc-company(X) ∧ conglomerate(X) ∧ sameGroup(X,Y) ⇒ TaxLaw(Y,USA,*Domestic*)
r_4 : american-assoc-company(X) ∧ associate(X,Y) ⇒ TaxLaw(Y,USA,*Domestic*)

{foreign-assoc-company(c1), associate(c1,c2), company(c3), allied-company(c2), associate(c2,c3), ¬american(c3)}

Semantics of CARIN

The semantics of CARIN are derived in a natural way from the semantics of its components languages. An interpretation I is a model of a knowledge base Δ if it is a model of each of it components. An interpretation I is a model of a rule r if, whenever α is a mapping from the variables of r to the domain $\mathcal{O}^I$, such that $\alpha(\bar{X}_i) \in p_i^I$ for every atom of the antecedent of r, then $\alpha(\bar{Y}) \in q^I$, where $q(\bar{Y})$ is the consequent of r. Finally, I is a model of a ground fact $p(\bar{a})$ if $\bar{a}^I \in p^I$. We make the *unique names assumption*, i.e., if a and b are constants in Δ, then $a^I \neq b^I$.

It should be noted that CARIN does not allow concept and role atoms to appear in the consequents of the rules because of the underlying assumption that the terminological component *completely* describes the hierarchical structure in the domain, and therefore, the rules should not allow to make new inferences about that structure.

Reasoning in CARIN

The reasoning problem we address in CARIN is the following. Given a CARIN knowledge base Δ and a ground atomic query of the form $p(\bar{a})$, where p can be any predicate and $\bar{a}$ is a tuple of constants, does $\Delta \models p(\bar{a})$? i.e., is $p(\bar{a})$ satisfied in every model of Δ.

Example 3: Our example knowledge base $\Delta_1 = \mathcal{T}_1 \cup \mathcal{R}_1$ entails TaxLaw(c3,USA,*Domestic*). To see why, we need to consider two possible cases for the company c2, corresponding to the disjunction in the definition of the concept allied-company:

1. In the first case company c2 is an instance of american-assoc-company. In that case, the antecedent of rule r_4 is satisfied with X=c2, Y=c3.
2. In the second case company c2 is an instance of american. Since company c1 is an instance of foreign-assoc-company it follows that c1 has at least two different associates, and is therefore an instance of conglomerate. In that case, since rules r_1 and r_2 entail sameGroup(c1,c3), the antecedent of rule r_3 is satisfied for X=c1, Y=c3.

It is important to note that we could not derive TaxLaw(c3,USA,*Domestic*) by simply applying the Horn rules to the ground facts, and a careful analysis of the different cases was required. In general, standard Horn rule reasoning procedures are *not* complete for CARIN KBs [Donini *et al.*, 1991a].

A sound and complete inference procedure for the case in which the rules of Δ are not recursive was described in [Levy and Rousset, 1996]. This paper considers the inference problem for *recursive* CARIN knowl-

edge bases. We show exactly when it is possible to obtain a CARIN language in which the reasoning problem is decidable.

The Undecidable Cases

In this section we show which of the constructors of $\mathcal{ALCNR}$ cause (each in isolation) the inference problem to become undecidable when combined with recursive function-free Horn rules. Interestingly, these constructors are generally considered to be at the core of description logics. Later we show that without these constructors we obtain a sublanguage of CARIN-$\mathcal{ALCNR}$ for which the reasoning problem is decidable. The following theorem shows that the constructors $\forall R.C$ and $(\leq n\,R)$ each cause undecidability.

Theorem 1: *The problem of determining whether $\Delta \models p(\bar{a})$ is undecidable, when Δ is a CARIN-$\mathcal{L}$ knowledge base with recursive function-free Horn rules, Δ has a non-cyclic terminological component that contains only concept definitions, and $\mathcal{L}$ is either*

1. *the description logic that includes only the constructor $\forall R.C$, or*
2. *the description logic that includes only the constructor $(\leq n\,R)$.*

The following theorem shows that introducing arbitrary (possibly cyclic) inclusion statements also causes undecidability.

Theorem 2: *The problem of determining whether $\Delta \models p(\bar{a})$ is undecidable, when Δ is a CARIN-$\mathcal{L}$ knowledge base with recursive function-free Horn rules, the terminological component of Δ allows arbitrary inclusion statements and $\mathcal{L}$ includes either only the constructor $\exists R.C$ or only the constructor $(\geq n\,R)$.*

The proofs of both theorems are obtained by encoding the execution of a Turing machine in a knowledge base of the form allowed in the theorems, and therefore obtaining a reduction from the halting problem.

Decidable Subset of Recursive CARIN-$\mathcal{ALCNR}$

We now show that in the language resulting from removing the constructors $\forall R.C$ and $(\leq n\,R)$ and terminological cycles the reasoning problem *is* decidable. Specifically, we consider the language CARIN-MARC[2] that includes the constructors $\sqcap$, $\sqcup$, $(\geq n\,R)$, $\exists R.C$ and negation on primitive concepts. Furthermore, CARIN-MARC allows only acyclic concept definitions in the terminological component. In what follows we describe a sound and complete inference procedure for CARIN-MARC.

Informally, our algorithm proceeds in two steps. In the first step we apply a set of *propagation rules* to the ground facts in the knowledge base, thereby constructing a finite number of *completions*. Each completion describes a subset of possible models of the ground facts and terminology of the knowledge base. Together, the union of the completions describes *all* the possible models of the ground facts and the terminology. In the second step, the algorithm applies a Horn-rule reasoning procedure in each of the completions. We show that a fact $p(\bar{a})$ is entailed by the knowledge base if and only if it is entailed in each of the completions that we construct, and we show how entailment can be checked in each completion.

[2]MARC stands for Maximal $\mathcal{ALCNR}$ Recursive CARIN.

The Inference Algorithm

Propagation Phase The propagation phase of our algorithm is based on the general method of *constraint systems* that was used in [Schmidt-Schauß and Smolka, 1991; Donini *et al.*, 1991b; Buchheit *et al.*, 1993] for deciding satisfiability of $\mathcal{ALCNR}$ knowledge bases. In this phase we manipulate sets of ground facts which we call *databases*. A ground fact has one of the following forms:

- $D(a)$, where D can be an arbitrary concept description,
- $R(a, b)$, where R can be an arbitrary role description,
- $p(\bar{a})$, where p is an ordinary predicate, or
- $a \neq b$.

We assume that all the concept definitions in $\Delta_{\mathcal{T}}$ are unfolded. The algorithm begins by creating an initial database S_Δ on which we perform the propagation phase. The initial database is constructed as follows. If $C(a)$ is a ground fact in Δ, where C is defined in $\Delta_{\mathcal{T}}$ by $C := D$, we add $D(a)$ to S_Δ (if C does not have a definition, then we simply add $C(a)$ to S_Δ). If $R(a, b) \in \Delta$, and R is defined in the terminology by $R = P_1 \sqcap \ldots \sqcap P_k$ then we add $P_1(a, b), \ldots, P_k(a, b)$ to S_Δ. We also add all the ground atomic facts of ordinary predicates in Δ to S_Δ. Finally, for every pair of constants (x, y) in Δ we add the fact $x \neq y$ to S_Δ.

Example 4: The initial example database S_{Δ_1} is:

(company $\sqcap$ ($\exists$ associate.$\neg$american))(c1),
(company $\sqcap$ (american $\sqcup$ $\exists$ associate.american))(c2),
$\neg$american(c3), associate(c1, c2), associate(c2, c3),
company(c3), c1 $\neq$ c2, c1 $\neq$ c3, c2 $\neq$ c3.

We now apply the set of *propagation rules* shown in Figure 1 to S_Δ. Each propagation rule corresponds to one of the constructors in our language, and its role is to make the constraints implied by this constructor *explicit*, by adding ground facts to the database. For example, the rule $\rightarrow_\sqcap$ adds the facts that are implicit in a given conjunction. Some of the rules (2, 3 and 5) are *non deterministic*, in that they can be applied in several ways. For example, the rule $\rightarrow_\sqcup$ can add either of the disjuncts of a disjunction. Because of the non deterministic rules we obtain a *set* of databases from

1. $S \rightarrow_{\sqcap} \{C_1(s), C_2(s)\} \cup S$
   if 1. $(C_1 \sqcap C_2)(s)$ is in S,
   2. $C_1(s)$ and $C_2(s)$ are not both in S.
2. $S \rightarrow_{\sqcup} \{D(s)\} \cup S$
   if 1. $(C_1 \sqcup C_2)(s)$ is in S,
   2. neither $C_1(s)$ nor $C_2(s)$ are in S,
   3. $D = C_1$ **or** $D = C_2$.
3. $S \rightarrow_{\exists} \{P_1(s,y), \ldots, P_k(s,y), C(y)\} \cup \{y \neq x \mid x \in s_R\} \cup S$
   1. $(\exists R.C)(s)$ is in S,
   2. $R = P_1 \sqcap \ldots \sqcap P_k$,
   3. there is no t such that t is an R-successor of s in S and $C(t)$ is in S,
   4. y is a new constant **or** one of the existing R-successors of s in S.
   5. s_R is $Succ(s,R) \setminus y$.
4. $S \rightarrow_{\geq} \{P_1(s,y_i), \ldots, P_k(s,y_i) \mid i \in 1..n_0\} \cup$
   $\{y_i \neq y_j \mid 1 \leq i,j \leq n_0, i \neq j\} \cup$
   $\{y_i \neq x \mid x \in Succ(s,R), 1 \leq i \leq n_0\} \cup S$
   if 1. $(\geq n\,R)(s)$ is in S,
   2. $R = P_1 \sqcap \ldots \sqcap P_k$,
   3. s has exactly m R-successors in S, and $n = m + n_0$
   4. $y_1, \ldots, y_{n_0}$ are new constants,
   5. there is no $l > n$, such that $(\geq l\,R)(s)$ is in S.
5. $S \rightarrow_{\neg} \{D(s)\} \cup S$
   if 1. A is a primitive concept and both $A(s)$ and $\neg A(s)$ are not in S,
   2. $D = A$ **or** $D = \neg A$

Figure 1: Propagation rules: t is said to be an R-successor of s in S if $P_1(s,t), \ldots, P_k(s,t) \in S$, and $R = P_1 \sqcap \ldots \sqcap P_k$. $Succ(s,R)$ denotes the set of R-successors of s.

applying the rules to S_Δ. A database is said to have a *clash* if it contains the facts $A(v)$ and $\neg A(v)$ for some primitive concept A. A database with a clash represents a non satisfiable set of ground facts. A database is considered to be a *completion* when no propagation rules can be applied to it. In the second phase of the algorithm we evaluate the Horn rules in each of the resulting clash-free completion databases.

Example 5 : We begin by applying $\rightarrow_{\sqcap}$ rule to c1 and c2. The facts company(c1), ($\exists$ associate.$\neg$american)(c1), company(c2) and (american $\sqcup$ $\exists$ associate.american)(c2) are added to S_{Δ_1} (see node 1 in Figure 2). At this point, we apply the $\rightarrow_{\sqcup}$ rule to c2, resulting in two possible databases: node 2 (in which ($\exists$ associate.american)(c2) is added), and node 3 (in which american(c2) is added). In node 2 we apply the rule $\rightarrow_{\exists}$ to c2. The fact ($\exists$ associate.american)(c2) implies that c2 has at least one filler on the role associate that is American. There are two options. This filler may be an existing one, i.e., c3 as in node 4, however, this causes a contradiction with an existing fact $\neg$american(c3). The second option is that there is another filler, v1, as in node 5. Since node 4 is contradictory, we do not consider it further. In node 5 we apply the rule $\rightarrow_{\exists}$ to c1. The constraint ($\exists$ associate.$\neg$american)(c1) implies that c1 has at least one filler on the role associate that is not American. Once again, there are two options, resulting in nodes 6 and 7. To complete this branch of the tree, we apply the $\rightarrow_{\neg}$ to v2, resulting in two completions (nodes 8 and 9), one in which v2 is a company and one in which it is not a company.

Similarly, we expand node 3 by applying the $\rightarrow_{\exists}$ to c1. The rule $\rightarrow_{\neg}$ is applied in node 11 to obtain the completions in nodes 12 and 13.

Horn Rule Evaluation Phase In the second phase of the algorithm we consider each of the clash-free completion databases generated in the first phase. Given a completion S and a query $q(\bar{a})$ we determine whether $S \cup \Delta_{\mathcal{R}} \models q(\bar{a})$. To do so, we can use any complete Horn rule reasoning method (e.g., backward or forward chaining) on each completion, except that the entailment of concept and role atoms has to be given special attention. The theorem below shows exactly when a ground fact is entailed from a completion. The conditions given in the theorem show how to modify a Horn-rule reasoning algorithm to obtain an algorithm for entailing facts from completions. The query $q(\bar{a})$ is entailed from Δ iff it is entailed from each clash-free completion database of S_Δ.

Theorem 3: *Let S be a clash-free completion resulting from applying the propagation rules on S_Δ.*

- *If $C(s)$ is an atom, where C is a concept name defined in $\Delta_{\mathcal{T}}$ by the description D, $S \cup \Delta_{\mathcal{R}} \models C(s)$ if and only if:*
  - *D is primitive or a negation of a primitive concept, and $D(s) \in S$,*
  - *$D = (\geq n\,R)$, and s has at least n R-successors in S,*
  - *$D = \exists R.C$, and s has an R-successor t such that $S \cup \Delta_{\mathcal{R}} \models C(t)$,*
  - *$D = C_1 \sqcap C_2$, and $S \cup \Delta_{\mathcal{R}} \models C_1(s)$ and $S \cup \Delta_{\mathcal{R}} \models C_2(s)$,*
  - *$D = C_1 \sqcup C_2$, and $S \cup \Delta_{\mathcal{R}} \models C_1(s)$ or $S \cup \Delta_{\mathcal{R}} \models C_2(s)$.*
- *$S \cup \Delta_{\mathcal{R}} \models R(s,t)$, where $R = P_1 \sqcap \ldots \sqcap P_k$ if and only if $P_i(s,t) \in S$ for i, $1 \leq i \leq k$,*
- *$S \cup \Delta_{\mathcal{R}} \models p(\bar{a})$, where p is an ordinary predicate, if and only if $p(\bar{a}) \in S$ or, there exists a rule $r \in \Delta_{\mathcal{R}}$ of the form $p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \Rightarrow p(\bar{Y})$ and a mapping ψ from the variables of r to constants, such that $\psi(\bar{Y}) = \bar{a}$, and $S \cup \Delta_{\mathcal{R}} \models \psi(p_i(\bar{X}_i))$ for i, $1 \leq i \leq n$.*

Example 6: We illustrate this phase on two completions shown in Figure 2. Consider the completion described in node 12 that includes the following facts from the original database S_{Δ_1}:

(($\exists$ associate.$\neg$american) $\sqcap$ company)(c1), company(c1), ($\exists$ associate.$\neg$american)(c1), company(c2), company(c3), (company $\sqcap$ (american$\sqcup$ $\exists$associate.american))(c2),

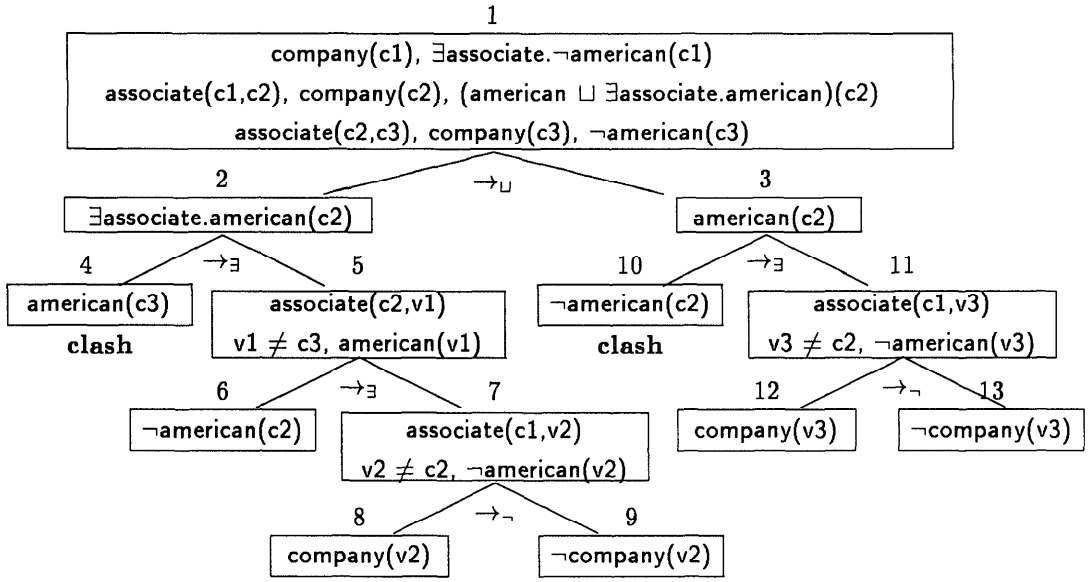

Figure 2: The application of propagation rules on S_{Δ_1}. Note that in every node we show only the facts that were *added* to the database. Under every node we show which propagation rule was applied to obtain its children. Note that nodes 8 and 9 should each have two children, one with company(v1) and the other with ¬company(v1).

(american ⊔ ∃associate.american)(c2), associate(c1, c2), associate(c2 c3), ¬american(c3).

and the following facts that were added during the propagation phase:

american(c2), associate(c1, v3), ¬american(v3), company(v3), v3 ≠ c2 .

The facts sameGroup(c1,c2) and sameGroup(c2,c3) are entailed by r_1, and therefore rule r_2 entails sameGroup(c1,c3). Since company c1 has two fillers on the role associate (c2 and v3), it is an instance of conglomerate. It is also given that c1 is an instance of foreign-assoc-company, and therefore, rule r_3 entails TaxLaw(c3,*USA*,*Domestic*).

Consider the completion in node 6 that has the following facts in addition to the initial database:

(∃associate.american)(c2), associate(c2,v1), american(v1), ¬american(c2), v1 ≠ c3.

In this completion company c2 is an instance of american-assoc-company, therefore, rule r_4 entails TaxLaw(c3,*USA*,*Domestic*). In fact, TaxLaw(c3, *USA*, *Domestic*) is entailed in all the clash-free completions, and therefore, it is entailed by Δ_1. □

Soundness, Completeness and Complexity

The soundness and completeness of our algorithm is established by the following theorem:

Theorem 4: *Let Δ be a* CARIN-*MARC knowledge base. $\Delta \models q(\bar{a})$ if and only if $S \cup \Delta_{\mathcal{R}} \models q(\bar{a})$ for every S that is a clash-free completion database of S_Δ.*

The key to proving Theorem 4 is Theorem 3 and the following lemma.

Lemma 5: *Let S be a clash-free database generated by applying a (possibly empty) sequence of propagation rules to S_Δ. Let ru be one of the propagation rules and let $S_1, \ldots, S_l$ be the databases that can be generated from S by applying ru. Then, $S \cup \Delta_R \models q(\bar{a})$ if and only if $S_i \cup \Delta_R \models q(\bar{a})$ for every i, $1 \leq i \leq l$.*

The complexity of reasoning in CARIN-MARC is given by the following theorem:

Theorem 6: *Let Δ be a* CARIN-*MARC knowledge base. Deciding whether $\Delta \models q(\bar{a})$ is CO-NP-Complete in the number of ground facts in Δ, CO-NP-Complete in the size of the terminology in Δ, and polynomial in the number of rules in Δ.*

CARIN-$\mathcal{ALCNR}$ with Role-Safe Rules

In this section we describe another way of obtaining a subset of CARIN-$\mathcal{ALCNR}$ for which sound and complete inference is possible for recursive function-free rules, while still allowing all the constructors of $\mathcal{ALCNR}$ and arbitrary (perhaps even cyclic) inclusion statements in the terminology. The subset language is obtained by restricting the Horn rules in the knowledge base to be *role-safe*, as we define below. Role-safe rules restrict the way in which variables can appear in role atoms in the rules. This restriction is similar in spirit to the *safety* condition on datalog KB's with order constraints (i.e., $=$, $\leq$, $<$ $\neq$), which is widely employed in deductive databases [Ullman, 1989]. Furthermore, many classical uses of recursion (e.g., connectivity on graphs whose edges are represented by ordinary predicates) can be expressed by role-safe rules.

Definition 1: *A rule r is said to be role-safe if for every atom of the form $R(x, y)$ in the antecedent, where R is a role and x and y are variables, then either x or y appear in an atom of an ordinary predicate in the antecedent of r.*

The following theorem shows that the reasoning problem in CARIN-$\mathcal{ALCNR}$ is decidable when all the Horn rules in the KB are role-safe.

Theorem 7: *Let Δ be a* CARIN-$\mathcal{ALCNR}$ *knowledge base in which all Horn rules are role-safe. The problem of determining whether $\Delta \models q(\bar{a})$ is decidable. It is co-NP-Complete in the number of ground facts in Δ, and is polynomial in the number of Horn rules in Δ. The entailment problem can be decided time doubly exponential in the size of the terminology of Δ,*

It should be noted that CARIN-$\mathcal{ALCNR}$ with role-safe rules is a strictly more expressive language than AL-Log [Donini *et al.*, 1991a], since AL-Log only allows concept atoms in the Horn rules and requires that the variables appearing in concept atoms appear in ordinary atoms as well.

Proof sketch: Our algorithm first adds to the terminological component of Δ the inclusion $\top \sqsubseteq C \sqcup \neg C$, for every concept C appearing in the Horn rules of Δ. The algorithm then proceeds in two phases as the previous one. The propagation rules used in the first phase are the rules used in the algorithm for deciding satisfiability of $\mathcal{ALCNR}$ knowledge bases, as described in [Buchheit *et al.*, 1993].

In the second phase of the algorithm we consider each of the finite number of clash-free completions computed in the first phase (note that the definition of a clash in [Buchheit *et al.*, 1993] is more elaborate to account for the new constructors). For each such completion we create a model of Δ from the *canonical interpretation* [Buchheit *et al.*, 1993] of the completion the ground facts and the rules in Δ, and check whether $q(\bar{a})$ is satisfied in this model. We show that $\Delta \models q(\bar{a})$ if and only if $q(\bar{a})$ is satisfied in every model that we check. □

Example 7: We illustrate the algorithm with the following simple example. Consider a knowledge base Δ_2 that contains the concept C, the role R, and the ordinary binary predicates e and p. The terminology has the single cyclic inclusion statement $C \sqsubseteq \exists R.C$, and we have the ground facts $C(a), C(b), e(a, b)$ and $e(b, c)$. Finally, we have the rules:

$s_1 : e(x, y) \wedge R(x, z) \Rightarrow p(x, y)$
$s_2 : p(x, z) \wedge p(z, y) \Rightarrow p(x, y)$.

The propagation phase would create the completion that includes the following facts in addition to those in the initial database: $R(a, v_1)$, $C(v_1)$, $R(v_1, v_2)$, $C(v_2)$, $R(b, u_1)$, $C(u_1)$, $R(u_1, u_2)$ and $C(u_2)$, where v_1, v_2, u_1 and u_2 are newly created constants. We create a model I of the completion as follows. The domain of I is $\{a, b, c, v_1, v_2, u_1, u_2\}$. The interpretations of the concepts and roles are

$C^I = \{a, b, v_1, v_2, u_1, u_2\}$,
$R^I = \{(a, v_1), (v_1, v_2), (v_2, v_2), (b, u_1), (u_1, u_2), (u_2, u_2)\}$.

The interpretation of e is taken directly from the ground facts in Δ_2: $e^I = \{(a, b), (b, c)\}$. Finally, the interpretation of p is constructed from the rules in Δ_2: $p^I = \{(a, b), (b, c), (a, c)\}$.

The important point is to note that the extension of R includes the tuples (v_2, v_2) and (u_2, u_2) that do not appear in the completion, but are necessary in order to obtain a finite model, while satisfying the inclusion $C \sqsubseteq \exists R.C$. The potential fallacy is that we would use these tuples to entail the existence of certain tuples in the extension of p. However, because of the fact that the rules are role-safe, these tuples are *not* needed in order to obtain the interpretation of the predicate p. In contrast, if instead of rule s_1 we would have the rule

$s_3 : R(x, y) \Rightarrow p(x, y)$

which is not role-safe, entailing facts for the predicate p would be more subtle. In particular, the interpretation of p that is built from the canonical interpretation would always include a cycle in the relation p, though there are models of Δ_2, in which no cycle exists. □

Related Work and Conclusions

We have shown the exact transition points in which combining recursive function-free Horn rules with description logics moves from a language in which the reasoning problem is decidable to a language in which it is undecidable. In particular, we showed that two of the core constructors of description logics (namely, $\forall R.C$ and $(\leq n\, R)$) lead by themselves to languages in which the reasoning problem is undecidable. We identified a maximal subset of CARIN-$\mathcal{ALCNR}$ in which reasoning is decidable. Moreover, we identified a restriction on the form of the rules (namely, role-safety) that guarantees decidable reasoning even in the existence of all the constructors of $\mathcal{ALCNR}$ and terminological cycles. This restriction covers many common usages of recursive rules, and is analogous to the safety condition assumed for datalog programs in deductive databases. It should be noted that by combining function-free Horn rules with description logics we obtain a limited form of using functions in rules, *without* leading to undecidability, as in the case of recursive Horn rules with arbitrary function symbols.

Combining $\mathcal{ALCNR}$ with non-recursive Horn rules was shown to be decidable in [Levy and Rousset, 1996]. The existential entailment algorithm described in [Levy and Rousset, 1996] combined with the *constrained-resolution* algorithm described by Bürckert [Bürckert, 1994] yields a refutation complete SLD-resolution algorithm for recursive CARIN-$\mathcal{ALCNR}$. Recall that the existence of refutation complete algorithm only guarantees semi-decidability.

The language AL-Log [Donini *et al.*, 1991a] combines the description logic $\mathcal{ALC}$ (which is a subset of $\mathcal{ALCNR}$) with recursive datalog rules. However,

AL-Log allows only concept predicates to appear in the rules, and furthermore, requires that all variables in the rules appear in ordinary predicates. KRYPTON [Brachman *et al.*, 1985] was the first system that combined a description logic subsystem with an assertional component that included statements in first order logic. KRYPTON allowed statements that were more expressive than Horn rules, but used a very limited description logic. The reasoning engine of KRYPTON was based on modifying a resolution engine to account for terminological inferences, and was not complete. MacGregor [MacGregor, 1994] and Yen [Yen, 1990] consider the combination of Horn rules with description logics in the CLASP system based on LOOM, and in order to determine rule-specificity and classification of arbitrary predicates. As LOOM is an undecidable language, their algorithms are also not complete.

A different approach to integrating rules and description logics is to add rules as an additional constructor in description logics (e.g., (CLASSIC [Brachman *et al.*, 1991], BACK [Petalson, 1991], LOOM [MacGregor, 1988]). These works allowed only rules of a restricted form: $C(x) \Rightarrow D(x)$, where C and D are concepts. Furthermore, the rules are generally not integrated in subsumption inferences but they are just used to derive additional knowledge about concept instances. LIFE [Aït-Kaci and Podelski, 1991] is a logic programming language that combined a rule component with a structural component of ψ-terms that differ from description logics in several significant ways. The most important difference is that in ψ-terms all roles are functional, thereby making the inferences about number restrictions and existential statements trivial. On the other hand, the variables in ψ-terms enable to express complex coreference constraints, that cannot be expressed in $\mathcal{ALCNR}$.

We are currently exploring several methods for optimizing inference in CARIN-$\mathcal{ALCNR}$. Specifically, we are considering optimizations that are based on identifying completions that are irrelevant to the given query, and therefore they do not have to be created or can be ignored once created. Further optimizations are based on *interleaving* the two phases of our algorithms and combining terminological reasoning steps with resolution steps similar other hybrid reasoning procedures.

Acknowledgements

The authors would like to thank David McAllester and Shuky Sagiv for helpful discussions.

References

Aït-Kaci, Hasan and Podelski, Andreas 1991. Towards the meaning of LIFE. In *Proceedings of the 3rd International Symposium on Programming Language Implementation and Logic Programming.*

Borgida, Alexander 1994. On the relationship between description logic and predicate logic. In *Proceedings of CIKM-94.*

Brachman, Ronald J.; Gilbert, Victoria P.; and Levesque, Hector J. 1985. An essential hybrid reasoning system: Knowledge and symbol level accounts of KRYPTON. In *Proceedings of IJCAI-85.*

Brachman, R. J.; Borgida, A.; McGuinness, D. L.; Patel-Schneider, P. F.; and Resnick, L. A. 1991. Living with CLASSIC: When and how to use a KL-ONE-like language. In Sowa, John, editor 1991, *Principles of Semantic Networks.* Morgan Kaufmann, San Mateo, CA. 401–456.

Buchheit, Martin; Donini, Francesco M.; and Schaerf, Andrea 1993. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research* 1:109–138.

Bürckert, Hans-Jürgen 1994. A resolution principle for constrained logics. *Artificial Intelligence* 66:235–271.

Donini, Francesco M.; Lenzerini, Maurizio; Nardi, Daniele; and Schaerf, Andrea 1991a. A hybrid system with datalog and concept languages. In *Trends in Artificial Intelligence*, volume LNAI 549. Springer Verlag.

Donini, Francesco. M.; Lenzerini, Maurizion; Nardi, Daniele; and Nutt, Werner 1991b. The complexity of concept languages. In *Proceedings of KR-91.*

Frisch, Alan, editor 1991. *Working Notes of the AAAI Fall Symposium on Principles of Hybrid Reasoning.* American Association for Artificial Intelligence.

Levy, Alon Y. and Rousset, Marie-Christine 1996. CARIN: a representation language integrating rules and description logics. In *Proceedings of ECAI-96.*

Levy, Alon Y.; Rajaraman, Anand; and Ordille, Joann J. 1996. Query answering algorithms for information agents. In *Proceedings of AAAI-96.*

MacGregor, R. M. 1988. A deductive pattern matcher. In *Proceedings of AAAI-88.*

MacGregor, Robert M. 1994. A description classifier for the predicate calculus. In *Proceedings of AAAI-94.*

Petalson, C. 1991. The BACK system : an overview. In *Proceedings of the SIGART bulletin*, volume 2(3). 114–119.

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

Ullman, Jeffrey D. 1989. *Principles of Database and Knowledge-base Systems, Volumes I, II.* Computer Science Press, Rockville MD.

Wahlster, W.; Andre, E.; Finkler, W.; Profitlich, H.J.; and Rist, T. 1993. Plan-based integration of natural language and graphics generation. *Artificial Intelligence* 63(1-2):387–428.

Wright, J.R; Weixelbaum, E.S.; Brown, K.; Vesonder, G.T; Palmer, S.R; Berman, J.I; and Moore, H.H 1993. A knowledge-based configurator that supports sales, engineering and manufacturing at AT&T network systems. In *Proceedings of IAAI-93.*

Yen, John 1990. A principled approach to reasoning about the specificity of rules. In *Proceedings of AAAI-90.*