# CARIN: A Representation Language Combining Horn Rules and Description Logics

**Alon Y. Levy**[1] and **Marie-Christine Rousset**[2]

**Abstract.**
We describe CARIN, a novel family of representation languages, which integrate the expressive power of Horn rules and of description logics. We address the key issue in designing such a language, namely, providing a sound and complete inference procedure. We identify *existential entailment* as a core problem in reasoning in CARIN, and describe an existential entailment algorithm for CARIN languages whose description logic component is $\mathcal{ALCNR}$. This algorithm entails several important results for reasoning in CARIN, most notably: (1) a sound and complete inference procedure for non recursive CARIN-$\mathcal{ALCNR}$, and (2) an algorithm for determining rule subsumption over $\mathcal{ALCNR}$.

## 1 Introduction

Horn rule languages have formed the basis for many Artificial Intelligence application languages because their expressive power is sufficient for many applications, and they have good computational properties. 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. The computational and expressive properties of description logics have been extensively studied. Several systems have been built based on description logics (e.g., CLASSIC [5], LOOM [15], BACK [18], KRIS [3]), they have been used in several applications (e.g., [20, 19]). 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 elements in the domain, and is defined by the conditions that must be satisfied by elements in the set. A description logic contains a set of *constructors* that allows to define these conditions at various levels of complexity. Much of the research in description logics has concentrated on algorithms for determining subsumption relations between classes, and for checking membership of an object in a class. In general, the expressive powers of Horn-rules and descriptions logics are uncomparable (i.e., one does not subsume the other) [4].

Naturally, many applications would significantly benefit from having the expressive power of both formalisms. This has been the driving force behind previous work on designing hybrid languages (e.g.,[6, 11, 2]). Two prime examples of such applications are information gathering from multiple sources [10, 14], where we require the expressive power of both formalisms in order to model *descriptions* of information sources, and the problem of modeling of complex physical devices [12], in which we need to model both the structural aspects of a device (which are best modeled by a description logic), and its behavior aspects (which are naturally modeled using Horn rules).

This paper describes CARIN, a novel family of languages, that extend Horn rules with the expressive power of description logics. CARIN combines the two formalisms by allowing rules to have concept and role atoms in their antecedents. The concepts and roles are defined in a *terminological* component of the knowledge base, as in description logics. The semantics of CARIN is derived naturally from the semantics of its component languages.

The key problem in developing such a hybrid language is designing a *sound* and *complete* inference procedure for answering queries in the language. As observed in [11] for an even simpler language than CARIN, applying standard Horn rule inference procedures (e.g., SLD resolution) to CARIN rules is sound, but not complete. This is because such procedures apply each rule in *isolation,* and they try to *instantiate* the antecedent of the rule in order to derive its consequent.

We address the reasoning problem in CARIN, and identify the *existential entailment* problem as a core problem for CARIN reasoning. Informally, the existential entailment problem is to decide whether a set of ground facts and a terminology entails the union of the antecedents of a set of rules. We describe an existential entailment algorithm for CARIN rules in which the description logic is $\mathcal{ALCNR}$ [8], which is the most expressive decidable implemented description logic [3]. This existential entailment algorithm entails several important results. First, it provides a sound and complete inference procedure for CARIN knowledge bases in which the Horn rules are not recursive and the description logic used is $\mathcal{ALCNR}$. In particular, it provides the first algorithm for answering *arbitrary* conjunctive queries over an $\mathcal{ALCNR}$ knowledge base. Second, our result provides an algorithm for rule subsumption over $\mathcal{ALCNR}$. Finally, our existential entailment algorithm can be combined with the *constrained-resolution* algorithm described by Bürckert [9] to yield a refutation complete SLD-resolution algorithm for recursive CARIN-$\mathcal{ALCNR}$.

## 2 Description Logics

A description logic contains unary relations (called concepts) which represent classes of objects in the domain and binary relations (called roles) which describe relationships between objects. Concepts and roles can be either primitive or complex. Complex concepts and roles are defined via *descriptions* built from a set of constructors. Description logics vary in the set of constructors they allow, and consequently in the complexity of determining class subsumption. In this paper we consider CARIN languages in which the description logic component is any subset of the rather expressive language $\mathcal{ALCNR}$ [8],

[1] AT&T Bell Laboratories, levy@research.att.com
[2] L.R.I. U.R.A C.N.R.S, University of Paris-Sud, mcr@lri.lri.fr


ECAI 96. *12th European Conference on Artificial Intelligence*
Edited by W. Wahlster
Published in 1996 by John Wiley & Sons, Ltd.

and we allow terminological cycles. Complex concepts and roles in $\mathcal{ALCNR}$ can be defined using the following syntax ($A$ denotes a primitive concept, $P_i$'s denote primitive roles, $C$ and $D$ represent arbitrary concepts and $R$ denotes an arbitrary role):

| | | |
|---|---|---|
| $C, D \rightarrow$ | $A \mid$ | (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) |

An $\mathcal{ALCNR}$ knowledge base contains two components, a terminological component $\mathcal{T}$ and an assertional component $\mathcal{A}$. The terminological component $\mathcal{T}$ contains a set of *inclusions* of the form $C \sqsubseteq D$. Intuitively, an inclusion states that every instance of the concept $C$ must be an instance of $D$. The terminological component is also used to define a set of *named* complex concepts. Such a definition can be given by two inclusions, $C \sqsubseteq D$ and $D \sqsubseteq C$, (abbreviated $C := D$), where $C$ is the name and $D$ is an arbitrary description. The assertional part of an $\mathcal{ALCNR}$ knowledge base $\mathcal{A}$ contains a set of ground atoms of the form $C(a)$ or $R(a, b)$, where $C$ is a named concept (primitive or complex) and $R$ is a role.

The semantics of a description logic knowledge base are given via *interpretations*. An interpretation $I$ contains a non-empty domain $\mathcal{O}^I$. It assigns an element $a^I$ in $\mathcal{O}^I$ to every constant $a$ in $\mathcal{A}$, and 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 $\mathcal{T}$. We make the *unique names assumption*, i.e., if $a$ and $b$ are constants in $\mathcal{A}$, then $a^I \neq b^I$. An interpretation $I$ is a model of $\mathcal{T}$ if $C^I \subseteq D^I$ for every inclusion $C \sqsubseteq D$ in the terminology, and the extensions of the complex concepts satisfy the following equations ($\sharp\{S\}$ denotes the cardinality of a set $S$):

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

We say that $C$ *is subsumed by* $D$ w.r.t. $\mathcal{T}$ if $C^I \subseteq D^I$ in every model $I$ of $\mathcal{T}$. As an example terminology, consider the following $\mathcal{T}_1$:

company $\sqsubseteq$ $\exists$associate.company
european-company $\sqcap$ american-company $\sqsubseteq \bot$
eur-ass-company := company $\sqcap$ $\exists$associate.european-company
am-ass-company := company $\sqcap$ $\exists$associate.american-company
fellow-company := company $\sqcap$ $\forall$associate.american-company
no-fellow-company := company $\sqcap$ $\forall$associate.$\neg$ american-company
conglomerate := company $\sqcap$ $(\geq 2$ associate)
int-company := eur-ass-company $\sqcap$ am-ass-company
foreign-associate-company := eur-ass-company $\sqcup$ am-ass-company

For example, the class eur-ass-company is defined to be the class of individuals who are instances of class company, and have at least one filler on the role associate, which is a member of class european-company.

As an example of subsumption inference, we can infer that the class int-company is subsumed by the class conglomerate. This is because instances of int-company are required to be instance of both eur-ass-company and of am-ass-company, and therefore to have both an American associate and a European associate. However, since European and American companies are disjoint sets, it entails that instance of int-company have at least two associates. As an example of inference involving individuals, given the ground facts fellow-company$(a)$ and associate$(a, b)$ we can infer that american-company$(b)$.

## 3 CARIN: Syntax and Semantics

The motivation behind the design of CARIN is to obtain a language in which unary and binary predicates whose properties are defined in the description logic can be exploited in Horn rules. CARIN is a family of languages parameterized by the specific description logic used. Formally, an CARIN-$\mathcal{L}$ knowledge base contains three parts:

**Terminology $\mathcal{T}$:** A terminological component which is a terminology defined in $\mathcal{L}$, as described above.

**Rule base $\mathcal{R}$:** a set of extended Horn rules, $\mathcal{R}$, i.e., logical sentences of the form:

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

where all the arguments are either variables or constants. We require that every variable that appears in $\bar{Y}$ also appear in $\bar{X} = \bar{X}_1 \cup \ldots \cup \bar{X}_n$. The predicates appearing in the antecedent of the rule may also be concepts or roles defined in $\mathcal{T}$. Predicates in $\mathcal{R}$ that are not concepts or roles are called *ordinary* predicates.

**Ground facts $\mathcal{A}$:** a set of ground facts for the concepts, roles and predicates appearing in $\mathcal{T}$ and $\mathcal{R}$.

The semantics of CARIN-$\mathcal{L}$ are derived in a natural way from the semantics of its component languages. An interpretation $I$ is a model of a CARIN knowledge base ($\mathcal{T}$,$\mathcal{A}$,$\mathcal{R}$) if it is a model of each of it components, defined as follows. Models of the terminological component were defined above. An interpretation $I$ is a model of a rule $r$ if, whenever $\alpha$ 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 $\bar{a}^I \in p^I$.

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 of classes in the domain, and therefore, the rules should not allow to make new inferences about that structure.

As an example of a CARIN-$\mathcal{ALCNR}$ knowledge base, we can consider the terminology $\mathcal{T}_1$, with the following rules, $\mathcal{R}_1$:

$r_1$ : made-by(X,Y) $\wedge$ no-fellow-company(Y) $\Rightarrow$ price(X,usa,high)
$r_2$ : made-by(X,Y) $\wedge$ company(Y) $\wedge$ associate(Y,Z) $\wedge$
american-company(Z) $\wedge$ monopoly(Y,X,usa) $\Rightarrow$ price(X,usa,high).

The following examples show the additional inferences that can be drawn from CARIN rules and not from either of it sublanguages alone. Suppose we have the ground facts:

$\mathcal{A}_1$ : {made-by(a,b), monopoly(b,a,usa), am-ass-company(b)}

Then we can infer $\mathcal{T}_1, \mathcal{A}_1, \mathcal{R}_1 \models \mathsf{price}(\mathsf{a}, \mathsf{usa}, \mathsf{high})$. The fact am-ass-company(b) and $\mathcal{T}_1$ entail that b has some associate that is an american-company. Therefore, even though no rule of $\mathcal{R}_1$ can be totally *instantiated* on $\mathcal{A}_1$, the missing conjuncts of $r_2$ are *entailed* by $\mathcal{A}_1 \cup \mathcal{T}_1$. Suppose we have the ground facts:

$\mathcal{A}_2$ : {made-by(a,b),monopoly(b,a,usa),
foreign-associate-company(b)}

Then we can infer $\mathcal{T}_1, \mathcal{A}_2, \mathcal{R}_1 \models \mathsf{price}(\mathsf{a}, \mathsf{usa}, \mathsf{high})$. Here $\mathcal{T}_1 \cup \mathcal{A}_2$ does not entail the antecedent of any *single* rule in $\mathcal{R}_1$. However, we can make the inference by reasoning by cases. If b has no American associates, then no-fellow-company(b) will be entailed, and price(a,usa,high) will follow from $r_1$. If b has at least one American associate, then price(a,usa,high) will follow because the antecedent of rule $r_2$ will be entailed, as explained above.

## 4 Reasoning in CARIN

In this paper we address the key issue of developing a *sound* and *complete* inference algorithm for CARIN knowledge bases. As illustrated in the example, there are two aspects of traditional Horn rule inference mechanisms that make them inadequate for CARIN knowledge bases. The first aspect is that they consider each rule in *isolation*, and the second aspect is that for each rule they try to *instantiate* the antecedent in order to derive the consequent. The example with $\mathcal{A}_1$ showed that a KB may entail the antecedent of a rule without the antecedent being instantiated in the KB. The example with $\mathcal{A}_2$ showed that a KB may entail the *union* of the antecedents of two rules without entailing either of them. These problems have also been observed in [11], even when role predicates were not allowed in the rules. Therefore, the process of instantiating a single rule needs to be replaced by an algorithm that decides whether a set of ground facts and a terminology *entails* the *union* of the antecedents of a set of Horn rules. We formalize this decision problem as the *existential entailment* problem. Formally, the existential entailment problem for a description logic $\mathcal{L}$ is the following. Let $\mathcal{T}$ be a terminology in $\mathcal{L}$, and let $\beta, Q_1, \ldots, Q_n$ be existential sentences of the form

$$(\exists\ \bar{Y})\ p_1(\bar{Y}_1) \wedge \ldots \wedge p_m(\bar{Y}_m)$$

where $p_1, \ldots, p_m$ are either roles or concepts defined in $\mathcal{T}$. The problem is to decide whether

$$\beta \cup \mathcal{T} \models Q_1 \vee \ldots \vee Q_n.$$

In Section 5 we present an algorithm for existential entailment for $\mathcal{ALCNR}$, establishing the following theorem:

**Theorem 1** *The existential entailment problem for $\mathcal{ALCNR}$ is decidable in time doubly exponential in the size of $\mathcal{T}$ and $\beta, Q_1, \ldots, Q_n$.*

As for the complexity bound, it should be noted that the complement of the existential entailment problem is harder than the $\mathcal{ALCNR}$-satisfiability problem considered in [8], for which a non-deterministic exponential time algorithm was given.

The existential entailment problem forms the core of several reasoning problems in CARIN. In particular, Theorem 1 implies the following results:

**Corollary 4.1:** *Let KB be an CARIN-$\mathcal{ALCNR}$ knowledge base whose Horn rules are not recursive. Let $q(\bar{a})$ be a ground atomic query, where $q$ is either a concept, role or ordinary predicate. The problem of determining whether $KB \models q(\bar{a})$ is decidable in time doubly exponential in the size of KB.*

The corollary follows from the observation that the ground facts of KB are a particular instance of the formula $\beta$ in the existential entailment problem, and from the fact that entailing antecedents of CARIN rules is not complicated by the fact that they contain ordinary predicates in addition to concepts and roles.

An important consequence of Corollary 4.1 is the first algorithm for answering arbitrary conjunctive queries (with existentially quantified variables) from an $\mathcal{ALCNR}$ knowledge base.

Finally, Theorem 1 also entails an algorithm for subsumption of conjunctive queries over $\mathcal{ALCNR}$. Such an algorithm is a key component for query optimization (e.g., [7]). The problem of query subsumption is the following.

**Definition 4.1:** *Let $Q_1$ and $Q_2$ be two conjunctive queries whose atoms contain concepts and roles that are defined in an $\mathcal{ALCNR}$ terminology $\mathcal{T}$. The query $Q_1$ is subsumed by $Q_2$ if, for any set of ground facts $\mathcal{G}$, the set of answers for $Q_1$ entailed by $\mathcal{G} \cup \mathcal{T}$ is a subset of the answers entailed for $Q_2$ from $\mathcal{G} \cup \mathcal{T}$.*

**Theorem 2** *Let $Q_1$ and $Q_2$ be two conjunctive queries whose atoms contain concepts and roles that are defined in an $\mathcal{ALCNR}$ terminology $\mathcal{T}$. The problem of determining whether $Q_1$ is subsumed by $Q_2$ is decidable in time doubly exponential in the size of $Q_1, Q_2$ and $\mathcal{T}$.*

As noted by Bürckert[3]the existential entailment algorithm can be combined with constrained SLD-resolution [9] to provide a goal directed backward chaining algorithm on arbitrary CARIN-$\mathcal{ALCNR}$ rules. Note that this procedure will yield a refutation complete procedure for CARIN-$\mathcal{ALCNR}$. That is, the algorithm will terminate if a CARIN-$\mathcal{ALCNR}$ knowledge base is not satisfiable, but may not terminate otherwise.

## 5 Existential Entailment Algorithm for $\mathcal{ALCNR}$

Our algorithm is based on the technique of *constraint systems,* also used in [8] for a satisfiability checking of $\mathcal{ALCNR}$ knowledge bases. Informally, in our setting, a constraint system represents a set of models of $\beta \cup \mathcal{T}$. The algorithm begins with an initial constraint system, constructed from $\beta \cup \mathcal{T}$, representing the set of *all* models of $\beta \cup \mathcal{T}$. It then applies a set of *propagation rules* that generate a set of *completions*. Each completion is a refinement of the initial constraint system, in which some implicit constraints have been made explicit, and some non-deterministic choices have been made. Thus, each completion represents a subset of the models of $\beta \cup \mathcal{T}$, and together they provide a finite representation of all the models of $\beta \cup \mathcal{T}$. The important property of completions is that checking whether entailment of $Q_1 \vee \ldots \vee Q_n$ can be done by checking whether it is satisfied in one constructable *canonical model* of the completion.

We first illustrate the use of constraint systems (e.g., as used in [8]) with an example. Consider the terminology $\mathcal{T}$ consisting of the single inclusion $C \sqsubseteq \exists R.C$, and $\beta = C(a)$. That is, the concept $C$ is defined to be the set of elements who have at least one filler on role $R$ and that filler is a member of class $C$.

The initial constraint system is $S_\beta = \{a : C, \forall x.x : \neg C \sqcup \exists R.C\}$, whose constraints correspond to $\beta$ and the inclusion. The first rule that is applied instantiates the second constraint with the individual $a$. It creates the constraint system $S_1 = S_\beta \cup \{a : \neg C \sqcup \exists R.C\}$. Because of the disjunction, two successor constraint systems to $S_1$ can be created, one in which $a : \neg C$ is added to $S_1$, and in the other we add $a : \exists R.C$. Since the first one is unsatisfiable, only the second is considered , and therefore $S_2 = S_1 \cup \{a : \exists R.C\}$ is obtained. The system $S_2$ implies that $a$ has a filler on the role $R$, therefore, a *generating* rule is applied, which adds a new variable $v_1$ to $S_2$ with the appropriate constraints, to obtain $S_3 = S_2 \cup \{aRv_1, v_1 : C\}$. Then, the same sequence of propagation rules that were applied to $a$ are also applied to $v_1$, to obtain the following constraint system: $S_4 = S_3 \cup \{v_1 : \neg C \sqcup \exists R.C, v_1 : \exists R.C, v_1Rv_2, v_2 : C\}$. At this point, the same propagation rules could be applied to $v_2$. However, $v_1$ and $v_2$ satisfy the same constraints in $S_4$. Hence, $v_2$ is said to be *blocked* by $v_1$, and the propagation terminates. To check whether the query is entailed by $S_4$, its canonical model $I$ is built. The domain of $I$ is $\{a, v_1, v_2\}$, and the extensions of the relations are: $C^I = \{a, v_1, v_2\}$, and $R^I = \{(a, v_1), (v_1, v_2), (v_2, v_2)\}$.

[3] personal communication.

The key point in using constraint systems is to decide when to terminate the construction. In the above example, we illustrated the termination condition used in [8], which was designed especially to check satisfiability of an $\mathcal{ALCNR}$ knowledge base. The subtle point in the algorithm is that when building the canonical model, fillers to the blocked variables *have* to be assigned. Above, a filler for role $R$ had to be assigned to $v_2$. In doing so cycles were introduced in the canonical model that do not exist in *every* model of $S_4$. For example, if the query $Q$ were $\exists\, x\ R(x, x)$, it would be satisfied in the canonical model, even though it is not entailed by $S_4$. Therefore, the key point in which our algorithm differs from that of [8] is that we develop a termination condition that depends on the query, and that guarantees that the resulting canonical model is sufficient for checking the entailment of the query. Finally, it is important to emphasize that the existential entailment problem *cannot* be reduced to a satisfiability problem of an $\mathcal{ALCNR}$ knowledge base, because the negation of the query *cannot* be expressed in an $\mathcal{ALCNR}$ knowledge base. Therefore, the algorithm in [8] does not suffice.

Formally, a constraint system is a non-empty set of constraints of the form $s : C$, $sPt$, $\forall x.x : C$, and $s \neq t$, where $C$ is a concept and $P$ is a role name.

We denote the set of variables and constants that appear in $Q$ or in $\beta$ by $\mathcal{V}$. From this point on, we refer to elements of $\mathcal{V}$ as *individuals*. In describing constraint systems, we introduce a new alphabet of variable symbols $\mathcal{W}$, with a well-founded total ordering $\prec_{\mathcal{W}}$. The alphabet $\mathcal{W}$ is disjoint from $\mathcal{V}$. We denote elements of $\mathcal{W}$ by the letters $u, v, w, x, y, z$. The term *object* refers to elements of $\mathcal{V} \cup \mathcal{W}$ (i.e., either variables or individuals). Objects are denoted by the letters $s, t$. Elements in $\mathcal{V}$ are denoted by the letters $a, b$.

Suppose $S$ is a constraint system and $R = P_1 \sqcap \ldots \sqcap P_k$ $(k \geq 1)$ is a role. We say that an object $t$ is an $R$-successor of an object $s$ if $sP_1t, \ldots, sP_kt \in S$. We say that $t$ is a *direct successor* of $s$ if it is the $R$-successor for some role $R$. The *successor* relationship denotes the transitive closure of the direct-successor relation. We say that $s$ and $t$ are *separated* in $S$ if $s \neq t \in S$. Finally, we denote by $S[x/s]$ the constraint system obtained from $S$ by replacing each occurrence of the variable $x$ by the object $s$.

For a variable $s$ in a constraint system $S$, we define the function $\sigma(S, s) := \{C \mid s : C \in S\}$. Two variables in $s, t \in S$ are said to be concept-equivalent if $\sigma(S, s) = \sigma(S, t)$. Intuitively, two variables are concept-equivalent in $S$ if, as far as the constraints in $S$ are concerned, they have the same properties, and therefore, unless they are separated in $S$, they *may* denote the same element in the domain. A constraint system contains a *clash*, if it contains $\{s :\perp\}$ or $\{s : A, s : \neg A\}$, or $\{s : (\leq\ n\, R)\} \cup \{sP_1t_i, \ldots, sP_kt_i \mid 1..n+1\} \cup \{t_i \neq t_j \mid i, j \in 1..n+1, i \neq j\}$ where $R = P_1 \sqcap \ldots \sqcap P_k$.

The semantics of constraint systems are given by interpretations, mapping each element $v$ of $\mathcal{V} \cup \mathcal{W}$ to an element $\alpha^{\mathcal{I}}(v)$ in the domain $\mathcal{O}^{\mathcal{I}}$, and specifying extensions for the concepts and roles. An interpretation satisfies the constraint $x : C$ if $\alpha^{\mathcal{I}}(x) \in C^{\mathcal{I}}$, the constraint $xRy$ if $(\alpha^{\mathcal{I}}(x), \alpha^{\mathcal{I}}(y)) \in R^{\mathcal{I}}$, the constraint $x \neq y$ if $\alpha^{\mathcal{I}}(x) \neq \alpha^{\mathcal{I}}(y)$, and the constraint $\forall x.x : C$ if $C^{\mathcal{I}} = \mathcal{O}^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of a constraint system $S$ if it satisfies every constraint in $S$. Note that a constraint system with a clash is unsatisfiable.

The initial constraint system $S_\beta$ is constructed as follows. If $p_i(\bar{Y}_i)$ is of the form $C(a)$, we put $a : C$ in $S_\beta$. If $p_i(\bar{Y}_i)$ is of the form $R(a, b)$ we put $aP_1b, \ldots, aP_nb$ in $S_\beta$, if $R = P_1 \sqcap \ldots \sqcap P_n$. If $C \sqsubseteq D$ is an inclusion in the terminology, we add $\forall x.x : \neg C \sqcup D$ to $S_\beta$. If $a$ and $b$ are two individuals in $\beta$ that are *not* existentially quantified, then we add $a \neq b$ to $S_\beta$. It is easy to see that the models of $S_\beta$ are precisely the models of $\beta$. Finally, for every concept $C$ that appears in $Q_1 \vee \ldots \vee Q_n$, we add the constraint $\forall x.x : C \sqcup \neg C$ to $S_\beta$. The last set of constraints enforces that in every completion $S$, $S \models C(v)$ or $S \models \neg C(v)$ for every object $v$. This is required in order to use constraint systems for entailment as opposed to satisfiability.

The algorithm applies the propagation rules described in [8] to the constraint system $S_\beta$. A constraint system is said to be a *completion* when no propagation rule applies to it. Two of the propagation rules are *generating* rules because they add new variables to the constraint system. In order to guarantee that the propagation process will terminate, the notion of *blocked variables* is used. For example, the following rule is applied in the presence of a constraint of the form $s : \exists\, R.C$, and is only applied if the variable $s$ is not blocked:

$S \rightarrow_\exists \{sP_1y, \ldots, sP_ky, y : C\} \cup S$
1. $s : \exists\, R.C$ is in $S$,
2. $R = P_1 \sqcap \ldots \sqcap P_k$,
3. $y$ is a new variable,
4. there is no $t$ such that $t$ is an $R$-successor of $s$ in $S$ and $t : C$ is in $S$,
5. if $s$ is not blocked.

The key point developing our algorithm based on constraint systems (and where we differ from the algorithm in [8]) is the definition of blocked variables and canonical models, on which we now elaborate.

As illustrated in our example by the query $\exists x\ R(x, x)$, the problem is that in the process of creating the canonical model, we are forced to assign successors to the blocked variables (in our example $(v_2, v_2) \in R^I$). Introducing these successors may cause the canonical model to contain connectivity patterns among variables that do not exist in *every* model of the completion. We now describe our revised termination condition that guarantees that such spurious patterns that may affect the satisfiability of the query are not introduced. Our condition will refine the previous one by considering the values of the $\sigma$ function not only of the variable itself, but also of its neighbors.

The *n-tree* of a variable $v$ is the tree that includes the variable $v$ and its successors, whose distance from $v$ is at most $n$ direct-successor arcs. We denote the set of variables in the $n$-tree of $v$ by $V_n(v)$. Two variables $v, w \in S$ are said to be $n$-tree equivalent in $S$ if there is an isomorphism $\psi : V_n(v) \rightarrow V_n(w)$, such that for every $s, t \in V_n(v)$ $sPt \in S$ if and only if $\psi(s)P\psi(t) \in S$, and $\sigma(S, \psi(s)) = \sigma(S, s)$. Intuitively, two variables are $n$-tree equivalent if the trees of depth $n$ of which they are roots are isomorphic. We denote by $D_Q$ the maximum number of literals in any of the $Q_i$'s. In our algorithm, a variable $v$ is blocked if the following holds:

1. $v$ is the leaf of a $D_Q$-tree rooted in a variable $v_1$,
2. there exists a variable $v_2$, such that $v_1$ is a successor of $v_2$, and $v_1$ and $v_2$ are $D_Q$-tree equivalent, and $v_1$ is not a node in the $D_Q$-tree of $v_2$.

The *witness* of the blocked variable $v$ is $\psi(v)$, where $\psi$ is the isomorphism between the $n$-trees of $v_1$ and $v_2$. To ensure that we can correctly detect blocked variables, we apply the propagation rules in a breadth first fashion. That is, we only apply a rule to a variable whose distance from an individual in $S$ is $n$ direct-successor arcs, if no rule is applicable to a variable with distance less than $n$.

Given a completion $S$, we define its *canonical model* as follows. Its domain is the set of objects in S, and each individual in $S$ is mapped to itself (i.e., $\alpha^{I_S}(s) = s$). The extensions of concepts and roles are defined as follows. For a primitive concept $A$, $s \in A^{I_S}$ if and only if $s : A \in S$. For a role $R$, $(s, t) \in R^{I_S}$ if and only if $sRt \in S$, or, $s$ is blocked with $w$ as its its witness, and $wRt$ is in $S$

The following theorem establishes the main property of our termination condition:

**Theorem 3** *Let $S$ be a completion of $S_\beta$, and let $I_S$ be its canonical model. If $I_S \models Q$, then $S \models Q$.*

**Proof sketch:** The proof of the theorem is based on showing that if $\rho$ is a mapping from the variables of $Q$ to the domain of $I_S$, that shows that $Q$ is satisfied in $I_S$, then there exists a mapping $\rho'$, that shows that $Q$ is satisfied, such that $\rho'$ does *not* use any arc emanating from a blocked variable in $I_S$. Since $\rho'$ uses only constraints that are *explicit* in $S$, these constraints will be satisfied in every model of $S$. □

Suppose our query is $Q = \exists X_1, X_2 R(X_1, X_2) \wedge R(X_2, X_2)$ The propagation rules would also generate the variables $v_3, \ldots, v_6$ similar to $v_1$ and $v_2$. The variable $v_6$ would be deemed blocked, because it is the leaf of a 2-tree rooted in $v_4$, and the 2-trees rooted in $v_1$ and $v_4$ are isomorphic. The witness of $v_6$ would be $v_3$, and therefore, the canonical model of the completion would have $(v_6, v_4)$ in $R^{I_S}$. The query $Q$ is not satisfied in the canonical interpretation.

The following establishes the correctness of our algorithm.

**Theorem 4** *Applying the propagation rules to $S_\beta$ will terminate with a finite number of completions, and $\mathcal{T} \cup \beta \models Q_1 \vee \ldots \vee Q_n$ holds if and only if $Q_1 \vee \ldots \vee Q_n$ is satisfied in the canonical model of every clash-free completion of $S_\beta$.*

**Proof sketch:** The first part follows from the observation that there are only a finite number of possible $n$-trees. For the second part, the *only if* direction follows from showing that the canonical interpretation of a completion of $S_\beta$ is a model of $\mathcal{T} \cup \beta$. The *if* direction follows from Theorem 3 and showing that every model of $\mathcal{T} \cup \beta$ is a model of *some* clash-free completion of $S_\beta$. □

## 6 Discussion

We described CARIN, a family of representation languages that extend Horn rules with the expressive power of description logics. We addressed the key issue of designing sound and complete inference procedures for CARIN knowledge bases. We describe an existential entailment algorithm for $\mathcal{ALCNR}$, which forms the core of several reasoning and optimization algorithms for CARIN KBs. This paper describes the first step in studying the inference problem for CARIN knowledge bases. We are currently exploring more efficient inference algorithms for CARIN knowledge bases that contain terminologies in languages more restricted than $\mathcal{ALCNR}$, and extensions of CARIN to recursive Horn rule knowledge bases. The expressive power of CARIN has been key to the development of the Information Manifold system that combines information from multiple sources on the World Wide Web [14].

Several other works have discussed the integration of Horn rules and description logics. Some works (e.g., AL-log [11], TaxLog [1], LIFE [2] or [17]) had the goal of using a description logic or other object-oriented component as a *typing* language on the variables already appearing in the rules (which could also be recursive). In those works, only unary predicates from the description logics are allowed in the rules. Other works (e.g., [6, 13]) considered a more tight integration of the two formalisms, which is more in the spirit of CARIN, e.g, KRYPTON [6] combined an assertional component (more expressive than Horn rules) with a less expressive description logic than $\mathcal{ALCNR}$. The reasoning engine was modified by either adding resolution steps to consider the inferences sanctioned by the terminological component, or by modifying the unification operation underlying the resolution engine. These approaches are either incomplete or guarantee only refutation completeness.

A different approach to integrating rules and description logics is to add rules as an additional constructor (e.g., (CLASSIC [5], BACK [18], LOOM [15]). 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. MacGregor [16] and Yen [21] describe algorithms for determining rule-specificity and classification of arbitrary predicates in LOOM, which are an instance of the existential entailment problem described here. However, since subsumption in LOOM is undecidable, their algorithms are not complete either.

## REFERENCES


[1] Andreas Abecker and Holger Wache, 'A layer architecture for the integration of rules, inheritance, and constraints', in *ICLP 94 post conference workshop on the Integration of Declarative Paradigms*, (1994).

[2] Hasan Aït-Kaci and Andreas Podelski, 'Towards the meaning of LIFE', in *Proceedings of the 3rd International Symposium on Programming Language Implementation and Logic Programming*, (1991).

[3] F. Baader and B. Hollunder, 'A terminological knowledge representation system with complete inference algorithm', in *In Proceedings of the Workshop on Processing Declarative Knowledge, PDK-91, Lecture Notes in Artificial Intelligence*, pp. 67–86. Springer-Verlag, (1991).

[4] Alexander Borgida, 'On the relationship between description logic and predicate logic', in *Proceedings of CIKM-94*, (1994).

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

[6] Ronald J. Brachman, Victoria P. Gilbert, and Hector J. Levesque, 'An essential hybrid reasoning system: Knowledge and symbol level accounts of KRYPTON', in *Proceedings of IJCAI-85*, (1985).

[7] M. Buchheit, M. Jeusfeld, W. Nutt, and M. Staudt, 'Subsumption between queries to object-oriented databases', *Information Systems*, **19(1)**, 33–54, (1994).

[8] Martin Buchheit, Francesco M. Donini, and Andrea Schaerf, 'Decidable reasoning in terminological knowledge representation systems', *Journal of Artificial Intelligence Research*, **1**, 109–138, (1993).

[9] Hans-Jürgen Bürckert, 'A resolution principle for constrained logics', *Artificial Intelligence*, **66**, 235–271, (1994).

[10] T. Catarci and M. Lenzerini, 'Representing and using interschema knowledge in cooperative information systems', *Journal of Intelligent and Cooperative Information Systems*, (1993).

[11] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf, 'A hybrid system with datalog and concept languages', in *Trends in Artificial Intelligence*, volume LNAI 549, Springer Verlag, (1991).

[12] Brian Falkenhainer and Ken Forbus, 'Compositional modeling: Finding the right model for the job', *Artificial Intelligence*, **51**, 95–143, (1991).

[13] Philipp Hanschke and Knut Hinkelman. Combining terminological and rule-based reasoning for abstraction processes. DFKI Report, 1992.

[14] Alon Y. Levy, Divesh Srivastava, and Thomas Kirk, 'Data model and query evaluation in global information systems', *Journal of Intelligent Information Systems*, **5 (2)**, (September 1995).

[15] R. M. MacGregor, 'A deductive pattern matcher', in *Proceedings of AAAI-88*, (1988).

[16] Robert M. MacGregor, 'A description classifier for the predicate calculus', in *Proceedings of AAAI-94*, (1994).

[17] Margarida Mamede and Luis Monteiro, 'A constraint logic programming scheme for taxonomic reasoning', in *Proceedings of the International Conference on Logic Programming, pp. 255-269*, (1992).

[18] C. Petalson, 'The BACK system : an overview', in *Proceedings of the SIGART bulletin*, volume 2(3), pp. 114–119, (1991).

[19] W. Wahlster, E. Andre, W. Finkler, H.J. Profitlich, and T. Rist, 'Plan-based integration of natural language and graphics generation', *Artificial Intelligence*, **63(1-2)**, 387–428, (1993).

[20] J.R Wright, E.S. Weixelbaum, K. Brown, G.T Vesonder, S.R Palmer, J.I Berman, and H.H Moore, 'A knowledge-based configurator that supports sales, engineering and manufacturing at AT&T network systems', in *Proceedings of IAAI-93*, (1993).

[21] John Yen, 'A principled approach to reasoning about the specificity of rules', in *Proceedings of AAAI-90*, (1990).