# Towards Learning in CARIN-$\mathcal{ALN}$

Céline Rouveirol, Véronique Ventos

Laboratoire de Recherche en Informatique
Bâtiment 490, Université Paris-Sud
91405 - Orsay Cedex (France)
{celine,ventos}@lri.fr

**Abstract.** In this paper we investigate a new language for learning, which combines two well-known representation formalisms, Description Logics and Horn Clause Logics. Our goal is to study the feasability of learning in such a hybrid description - horn clause language, namely CARIN-$\mathcal{ALN}$ [LR98b], in the presence of hybrid background knowledge, including a Horn clause and a terminological component. After setting our learning framework, we present algorithms for testing example coverage and subsumption between two hypotheses, based on the existential entailment algorithm studied in[LR98b]. While the hybrid language is more expressive than horn clause logics alone, the complexity of these two steps for CARIN-$\mathcal{ALN}$ remains bounded by their respective complexity in horn clause logics.

## 1 Introduction

Description Logics (DL in the remainder of the paper) have been specifically designed for representing and manipulating hierarchical knowledge. They have been extensively studied wrt two tasks: *least common subsumer* computation and *instance checking*. Although they lack some of the expressive power of Definite Logics (in particular for representing relations between objects), they offer, among other things, interesting connectives (such as cardinality connectives) and a limited form of negation, which can be quite useful in some applications. Moreover, some of them exhibit polynomial complexity in one or both of the tasks quoted above.

Definite Logics offers certainly a greater flexibility for representing complex relations, but the cost for this is high complexity for elementary learning tasks such as subsumption checking. Our goal is to study the feasability of learning in a hybrid description logic - horn clause language in the presence of hybrid background knowledge, including a Horn clause and a terminological component. The added expressive power of such hybrid languages wrt their component languages will enable us to express rich constraints on classes of individuals in the form of a concept hierarchy, while still maintaining the ability to use predicates of any arity and the ability to express arbitrary joins between relations.

We rely on previous theoretical work on a family of such hybrid languages, CARIN [LR98b], which has already been studied in the context of knowledge base verification [LR98a]. The expected pay-off for learning is that we can increase the expressivity of the target concept language without increasing the complexity of the learning process, which remains bounded by the complexity of learning in horn clause logics.

The paper is structured as follows : we first introduce in section 2 the basic notions and notations for DL languages, the CARIN family of languages, and the central inference of CARIN, namely existential entailment. Section 3 introduces our learning framework. Section 4 and 5 respectively detail how we can adapt the existential entailment algorithm in order to test whether a hypothesis covers an example and whether a hypothesis is more general than another. Section 6 studies the complexity of such elementary inferences in our language. Finally, section 7 compares our work to previous approaches and sketches the perspectives of this line of research. Each step will be illustrated by examples inspired from the well-known Michalski's train domain [MMPS94].

## 2 Knowledge representation: the CARIN languages

CARIN is a family of languages, each of them combining a description logic $\mathcal{L}$ with *a set of Horn Clauses*. A CARIN-$\mathcal{L}$ language contains three components. The first is a description logic terminology described using a DL $\mathcal{L}$, the second is a set of Horn clauses and the third is a set of ground facts. The terminology is a set of statements in $\mathcal{L}$ about concepts and roles in the domain. Concepts and roles can also appear as predicates in the antecedents of the Horn clauses and in the ground facts. In contrast, previous hybrid languages (e.g. [DLNS91]) only allow concepts to appear in the Horn clauses. Besides, the variables used in atoms of concepts must appear elsewhere in the rule which is not the case in CARIN.

### 2.1 Terminological component: the $\mathcal{ALN}$ description logic

Description Logics are a family of knowledge representation formalisms which stem from KL-ONE [Bra78]. Several systems have been built based on DLs (e.g. CLASSIC [BPS94], FLEX [QDBK96]) and they have been used in real-world applications (e.g. CLASSIC in [WWB$^+$93]). In addition, DLs facilitate the use of background knowledge and are more expressive than attribute-value representations. DLs are restrictions of first-order logic different from Horn clauselanguages[1], in which the subsumption computation and its complexity have been deeply studied [DLNN91].

In DLs, there are three kinds of formal objects: *individuals*, sets of individuals called *concepts* which are represented as unary predicates, and relationships between two individuals called *roles*, which are represented as binary predicates. A *concept* is defined as a set of properties satisfied by individuals that are instances of the concept. These properties are expressed by terms that are built

[1] Note that comparing the expressive power of DLs and restrictions of First Order Logic used in ILP is still an open issue [CH94b].

from atomic concepts and roles and from a set of connectives. Concepts are partially ordered by a subsumption relation which expresses the inclusion relation between concepts and is usually based on a standard model-based logical semantics. The basic inference tasks in DLs are determining *subsumption* relations between concepts (this operation enables us to automatically compute the terminology) and checking the membership of an individual in a concept (this operation is called *instance checking*).

In DLs, knowledge is mainly separated into two components: a *terminological component* (refered to as *T-box*), which contains the definition of concepts and an *assertional component* (*A-box*), which contains statements about individuals. The T-Box of a CARIN-$\mathcal{L}$ language will be a classical DL T-box, the ground fact component (section 2.3) will stand for an extension of a classical DL A-box.

The complexity of reasoning in a CARIN-$\mathcal{L}$ knowledge base depends on the expressivity of the terminological components. DLs and their tractability properties vary depending on the set of allowed connectives. For instance, CLASSIC is the most expressive implemented DL for which the subsumption computation is polynomial according to a non-standard semantics. However, this is not enough to guarantee efficient learning in Valiant's sense of PAC-learnability [Val84]. It has been shown in [CH94b,CH94a] that although CLASSIC is not PAC-learnable, some of its subsets are learnable (e.g. C-CLASSIC, k-core-CLASSIC). Because of these results and some complexity results about reasoning in CARIN (see section 6 for a short complexity study), we consider another subset of CLASSIC called $\mathcal{ALN}$.

The terminological language of $\mathcal{ALN}$ is rather simple and is defined using a set $\mathbf{R}$ of atomic roles, a set $\mathbf{P}$ of atomic concepts, the constants $\top$ and $\bot$, and the following syntactic rule ($C$ and $D$ are concepts, $P$ is an atomic concept, $R$ is a atomic role, and $n$ is an integer):

| | | |
|---|---|---|
| $C, D \rightarrow$ | $\top$ | the most general concept |
| | $\mid \bot$ | the most specific concept |
| | $\mid P$ | atomic concept |
| | $\mid \neg P$ | negation of atomic concept |
| | $\mid C \sqcap D$ | concept conjunction |
| | $\mid \forall R.C$ | value restriction |
| | $\mid \geq nR$ | cardinality for $R$ (minimum) |
| | $\mid \leq nR$ | cardinality for $R$ (maximum) |

This syntactic rule is used to define $\mathcal{ALN}$ *terms*.

**Example 1** *Train* $\sqcap \geq$ *2 has_car* $\sqcap \leq$ *4 has_car* $\sqcap\ \forall$*has_car.Square* $\sqcap\ \forall$*has_car.*$\leq$ *2 has_load describes all trains that have between two and four cars, and whose cars are square and have two or less loads.*

A terminological component *TC* is composed of T-box statements that have one of the following forms :

1. $A \equiv C$
2. $A \sqsubseteq C$

where $A$ is a concept name and $C$ is a *term* of the $\mathcal{ALN}$ terminological language. A concept name $A$ depends on a concept name $A'$ if $A'$ appears in the definition of $A$. A set of concept definitions is said acyclic if there is no cycle in the concept name dependency relation. Here, we only consider acyclic concept definitions. We assume that a concept name appears on the left hand side of at most one statement. A concept name $A$ occurring on the left-hand of a statement of type 1 is called a *defined concept* (i.e. $A$ has necessary and sufficient properties). A concept name $A$ occurring on the left-hand of a statement of type 2 is called a *primitive concept* (i.e. $A$ has necessary but not sufficient properties).

**Example 2** *The concept definition light_loaded_train* $\equiv$ *train* $\sqcap \geq$ *1 has_car* $\sqcap$ $\forall$*has_car.*($\geq$ *1 has_load* $\sqcap \leq$ *2 has_load) means that a train is not too loaded if it has more than one car, and that each of its cars contains one or two loads.*

Note that role fillers of individuals can be be precised whitout being enumerated. For instance, we can assert how many objects an individual is related to via some role whitout knowing these objects (e.g. $\geq 2 has-car(a)$: the train $a$ has at least two cars), or describe the fillers of a role whitout knowing them (e.g. $\forall has-car : square(a)$: all the cars of the train $a$ are square). The absence of the closed-world assumption enables to support partially known situations. When new knowledge is acquired incomplete information may be gradually refined if the additional knowledge do not contradict past data. The semantics of the terminological component is given via interpretations. Let $\Delta$ be the domain and an interpretation function $I$. The extensions of $\mathcal{ALN}$ terms are computed in the following manner:

- The extension of an atomic concept $P$ is a subset of $\Delta$ noted $\mathrm{P}^I$
- The extension of the negation of an atomic concept $\neg P$ is $\Delta$ - $\mathrm{P}^I$
- The extension of an atomic role R is a subset of $\Delta \times \Delta$ noté $\mathrm{R}^I$
- The extension of a general concept is:
  - $\top^I = \Delta$
  - $\bot^I = \emptyset$
  - $(C \sqcap D)^I = C^I \cap D^I$
  - $(\forall R.C)^I = \{\mathrm{x} \mid \mathrm{x} \in \Delta : \forall y, <x,y> \in R^I \Rightarrow y \in C^I\}$
  - $(\geq n\ \mathrm{R})^I = \{\mathrm{x} \in \Delta : |\{y :<x,y> \in R^I\}| \geq n\}$
  - $(\leq n\ \mathrm{R})^I = \{\mathrm{x} \in \Delta : |\{y :<x,y> \in R^I\}| \leq n\}$

**Definition 1 (TC models)** *An interpretation I is a model for a terminological component TC iff* $A^I = C^I$ *for every concept definition* $A \equiv C$ *in the terminology, and* $A^I \subseteq C^I$ *for every inclusion* $A \sqsubseteq C$

The subsumption is defined as follows:

**Definition 2 (Subsumption)** *A concept* $D$ *subsumes a concept* $C$ *wrt* $TC$ *iff for all models* $I$ *of* $TC$*, the extension of* $C$ *is included in the extension of* $D$*:*
$C \sqsubseteq D$ *iff* $C^I \subseteq D^I$ *for every model* $I$ *of* $TC$*.*

**Example 3**
$polygon \equiv closed\_shape \ \sqcap \ \geq 1 \ has\_side.$
$non\_polygon \equiv closed\_shape \ \sqcap \ \leq 0 \ has\_side.$
$triangle \equiv \ three\_sided.$
$three\_sided \equiv \ polygon \ \sqcap \ \geq 3 \ has\_side \ \sqcap \ \leq 3 \ has\_side.$
$four\_sided \equiv polygon \ \sqcap \ \geq 4 \ has\_side \ \sqcap \ \leq 4 \ has\_side.$
$parallelogram \equiv four\_sided \ \sqcap \ parallel\_sides.$
$rectangle \equiv parallelogram \ \sqcap \ square\_angles.$
$square \equiv rectangle \ \sqcap \ equal\_sides.$
$circle \equiv non\_polygon \ \sqcap \ has\_radius.$
$ellipse \equiv non\_polygon \ \sqcap \ has\_two\_focuses.$
$empty\_car \equiv car \ \sqcap \ \leq 0 \ has\_load.$
$empty\_train \equiv train \ \sqcap \ \forall has\_car.empty\_car.$

Classification allows the automatic generation of the taxonomy partially represented in figure 1.

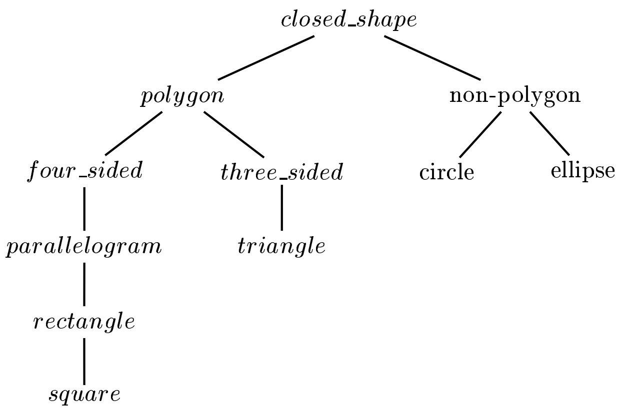


**Fig. 1.** Partial taxonomy for $TC$

## 2.2 Horn clause component

A Horn clause component R in CARIN is a set of Horn clauses of the form: $q(\bar{Y}) \leftarrow p_1(\bar{Y_1}) \wedge \ldots \wedge p_m(\bar{Y_m})$ where $\bar{Y}, \bar{Y_1}, \ldots, \bar{Y_m}$ are tuples of variables or constants. The clauses are safe (i.e., only variables belonging to one or more $\bar{Y_i}$ should be allowed in $\bar{Y}$). The predicates $p_i$ occurring in the body of the clause are of two kinds. They can be either *DL predicates*, (i.e., *concept* or *role* predicates), or *ordinary* predicates. A concept atom for concept $c$ defined in the terminology $TC$ is a literal of the form $c(X)$. A role atom for role $r$ used in the $TC$ terminology is a literal of the form $r(X, Y)$. Since an *ordinary predicate* does not correspond to any concept or role of $TC$, it can therefore be of any arity. The predicate of the head of the clause, $p$, must be an ordinary predicate: CARIN does not allow concept or role atoms to appear in the conclusion of Horn clauses, since it is assumed that $TC$ *completely* describes the terminological background knowledge. Finally, we are only addressing non-recursive Horn clause components, although [LR98b] also considers recursive Horn clause components.

**Example 4**

$has\_square\_car(X) \leftarrow train(X), has\_car(X,Y), car(Y), square(Y).$
$same\_shape(X,Y) \leftarrow X \neq Y, square(X), square(Y).$
$same\_shape(X,Y) \leftarrow X \neq Y, rectangle(X), rectangle(Y).$
$\vdots$
$pattern(X,Y,Z) \leftarrow train(X), has\_car(X,Y), car(Y), has\_car(X,Z), car(Z),$
$\quad square(Z), has\_load(Z,U), same\_shape(Y,U).$
$east\_train(X) \leftarrow train(X), has\_car(X,Y), car(Y), has\_car(X,Z), car(Z),$
$\quad has\_load(Z,L), same\_shape(Y,L).$
$east\_train(X) \leftarrow train(X), \forall has\_car.polygon(X).$

*In this example, the ordinary predicates are has_square_car, same_shape, pattern and east_train. All other predicates are DL predicates.*

### 2.3 Ground fact component

The ground fact component is a set of ground atomic facts concerning either DL or ordinary predicates.

**Example 5**

$\mathcal{A} = \{train(a), has\_car(a,b), car(b), has\_load(b,c), load(c), same\_shape(b,c),$
$\quad has\_car(a,d), car(d), \leq 0 has\_load(d), car(e), has\_car(a,e), \leq 0\ has\_load(e),$
$\quad east\_train(a)\}.$

This ground fact component describes a train going east. A ground fact component represents a particular world or an interpretation in De Raedt's sense (see section 3.1).

### 2.4 Expressivity of the language

Considering formulas in $TC$ or $R$, some formulas can be represented in our language which cannot be represented in Datalog. For instance, if we consider the $TC$ rule $empty\_train \sqsubseteq train \sqcap \forall has\_car.empty\_car$, its translation in First Order Logics is $\forall X empty\_train(X) \rightarrow (train(X) \wedge (\forall Y has\_car(X,Y) \rightarrow empty\_car(Y)))$, which is obviously not a Horn clause because of the universal quantification of "car" variables that occur only in the body of the clause[2].

[2] This type of formula is a restricted case of first order languages with existential quantified variables, which has already been studied for learning, for example in [Gon97,NCVLRDR99].

Moreover, cardinality connectives in the DL allows for smooth and sound handling of simple numeric constraints. Finally, there is also some expressivity gain wrt representation of the ground fact component. In a CARIN$_{\mathcal{ALN}}$ ground fact component, we can express facts concerning both ordinary and DL predicates. As a consequence, we can easily express incomplete information about the observations. For instance, we can represent that a train has more than four cars, without explicitly naming them: $train \sqcap \geq 4has\_car$.

### 2.5 Reasoning in CARIN

The key reasoning problem in CARIN is the *existential entailment* problem, fully described in [LR98b] for CARIN-$\mathcal{ALCNR}$, a more complex CARIN language than the one we address here. The existential entailment algorithm standing at the core of several reasoning problems in CARIN, and will be used to order hypothesis our hypothesis space and to check example coverage (see section 4).

Formally, the existential entailment problem for CARIN-$\mathcal{ALN}$ as follows.

**Definition 3 (Existential entailment)** *Let $TC$ be a terminology in $\mathcal{ALN}$, let $\beta$ and $Q_i$ be existential sentences of the form: $(\exists \bar{Y})\ p1(\bar{Y_1}) \wedge \ldots \wedge pm(\bar{Y_m})$ where $p_i$ are either roles or concepts occurring in $TC$, $\bar{Y_i}$ are tuples of variables and constants and $\bar{Y} \subseteq \bar{Y_1} \cup \ldots \cup \bar{Y_m}$, let $Q$ be a sentence of the form $Q_1 \vee \ldots \vee Q_n$. The variables that are not existentially quantified in $Q$ or $\beta$ are considered universally quantified. Note that any variable appearing in $Q$ must also appear in $\beta$.*
*The existential entailment problem is to decide whether $\beta \cup TC \models Q$.*

In [LR98b], an existential entailment algorithm is given for the language CARIN-$\mathcal{ALCNR}$. This algorithm is based on the construction of *constraint systems* that represent set of models of $\beta \cup TC$.

The algorithm begins with an initial constraint system, $S\beta$, representing the set of all models of $\beta \cup TC$ (e.g. $a$ and $b$ are individuals, the presence of $C(a)$ and $R(a,b)$ in $\beta$ leads to put $a : C$ and $a\ R\ b$ in $S\beta$). Note that $TC$ is taken into account (e.g. if $C(a)$ is in $\beta$ and if the sentence: $C \equiv D \sqcap E$ is in $TC$, them both $a : C$, $a : D$, $a : E$ are in $S\beta$). A set of propagation rules $\mathcal{PR}$ is then applied on $S\beta$ to obtain a set of completions $\mathcal{S}$. A propagation rule corresponds to one of the connectives in the DL and allows us to add some implicit constraints (in this case, it is a kind of saturation) or to make some non-deterministic choices (such propagation rules are called *non-deterministic rules*). Propagation rules concerning the language $\mathcal{ALCNR}$ are fully described in [LR98b], p176.

We will now describe some of these propagation rules for $\mathcal{ALN}$. We say that an individual $t$ is a *R-successor* of an individual $s$ in a constraint system $S$ if $s\ R\ t \in S$.

The propagation rule linked to the concept conjunction connective, denoted $\rightarrow_{\sqcap}$, is as follows ($S$ is the constraint system, $s$ and $t$ are individuals, $C_1$, $C_2$ and $C$ are concepts and $R$ is a role):
$S \rightarrow_{\sqcap} S \cup \{s : C_1, s : C_2\}$ if

1. $s : C_1 \sqcap C_2$ belongs to $S$,
2. $s : C_1$ and $s : S_2$ are not both in $S$.

The propagation rule linked to value restriction on roles, $\rightarrow_\forall$, is:
$S \rightarrow_\forall S \cup \{t : C\}$ if

1. $s : \forall R.C$ belongs to $S$,
2. $t$ is an R-successor of $s$,
3. $t : C$ is not in $S$.

$s : \forall R.C$ means that if there exists individuals that are in relation with $s$ by $R$, these individuals must belong to the concept $C$. For instance, $s : \forall child.Happy$ expresses the fact that if $s$ has children, all her children are happy. If we know that $t$ is a child of $s$ ($s\ child\ t \in S$) we must add the fact that $t$ is happy ($t : Happy$ is added to $S$) according to $s : \forall child.Happy$. The $\rightarrow_\forall$ propagation rule leads us to add such constraints.

The following propagation rule, $\rightarrow_\geq$, is linked to number restriction on roles:
$S \rightarrow_\geq S \cup \{s\ R\ y_i, i \in 1, \ldots, n\} \cup \{y_i \neq y_j, i, j \in 1, \ldots, n, i \neq j\}$ if

1. $s : (\geq nR)$ is in $S$,
2. $y_1, \ldots, y_n$ are new variables,
3. there are no $n$ pairwise separated R-successors of $s$ in $S$,
4. $s$ is not blocked (blocked variables are used in order to avoid infinite loops, see [LR98b] for more details)

The obtained completions correspond to different refinements of the constraint system. Some completions represent unsatisfiable constraint systems (e.g. $\{s : A, s : \neg A\}$ states that the individual $s$ belongs both to concept $A$ and to its negation). Clash-free completions represent a subset of the models of $\beta \cup TC$, but the entailment of a formula $Q$ from a clash-free completion can be done by checking whether or not the formula is satisfied in one canonical model of the completion. Finally, for each clash-free completion of $\mathcal{S}$, a canonical interpretation is built and existential entailment checks whether or not $Q$ is satisfied in all the canonical interpretations obtained.

The proof of correctness of the algorithm for $\mathcal{ALCNR}$ has been given in [LR98b] with some complexity results. If we consider $\mathcal{ALN}$ as the DL language, the existential entailment problem is much simpler. An optimized version of the existential entailment algorithm for $\mathcal{ALN}$ is currently being implemented. Note that the $\rightarrow_\geq$ rule creates additional variables (or additional constants in our framework) in the constraint system in order to ensure that the $\geq nR$ is satisfied. This is the reason why a Horn clause of the form: $p(X) \leftarrow R(X, Y)$ can be triggered by a ground fact component $\mathcal{A}$ even if no statement of the form $R(a, b)$ occurs in $\mathcal{A}$.

**Example 6** *Let* $TC = \{square \equiv rectangle \sqcap equal\_sides;\ rectangle \equiv parallelogram \sqcap square\_angles\}$ *(for simplicity, we assume here that parallelo– gram, equal_sides and square_angles are atomic concepts).*
$\mathcal{A} = \{train(a),\ \geq 1\ has\_car(a),\ \forall has\_car.rectangle(a),\ \forall has\_car.equal\_sides(a)\}$.

*The initial constraint system of $\mathcal{A} \cup \mathcal{TC}$, $S\beta$ is $\{a : train, a : \geq 1\ has\_car,\ a : \forall\, has\_car.rectangle,\ a : \forall has\_car.equal\_sides,\ a : \forall has\_car.parallelogram,\ a : \forall has\_car.square\_angles\}$. After applying the propagation rules of $\mathcal{PR}_{\mathcal{ALN}}$, the constraint system $S$ is $\{a : train,\ a : \geq 1\ has\_car,\ a : \forall has\_car.rectangle,\ a : \forall has\_car.equal\_sides,\ a : \forall has\_car.parallelogram,\ a : \forall has\_car.square\_angles,\ a\ has\_car\ b,\ b : rectangle,\ b : equal\_sides,\ b : parallelogram,\ b : square\_angles\}$*

*The constant $b$ has been added because of the $\rightarrow_{\geq}$ rule (and thereafter the constraint $a\ has\_car\ b$). The $\rightarrow_{\forall}$ rule yields the addition of the constraints $b : rectangle$, $b : equal\_sides$, $b : parallelogram$, $b : square\_angles$. If $H_1$ is the clause $p(X) \leftarrow has\_car(X,Y), square\_angles(Y)$, we obtain that $H_1, \mathcal{A}, TC \models p(a)$ since $S$ is a model of the antecedent of $H_1$ (with $Y = b$).*

*Let $H_2$ be the clause: $p(X) \leftarrow has\_car(X,Y), square(Y)$. $S$ is also a model of the antecedents of $H_2$, but the evaluation of $H_2$ needs to use a special procedure (described p. 195 in [LR98b]) which checks entailment of a ground atom of a concept or role predicate. The evaluation of $H_2$ leads first to state that $H_2, \mathcal{A}, TC \models p(a)$ iff $\mathcal{A}, TC \models \exists Y, has\_car(a,Y), square(Y)$. Let $compl(\mathcal{A})$ be the set of ground facts constructed for $S$ (e.g. the presence of the constraint $b : rectangle$ in $S$ leads the addition of the ground fact $rectangle(b)$ in $compl(\mathcal{A})$). $square(b)$ is not in $compl(\mathcal{A})$, but the special procedure checks entailment of $square(b)$ according to the following rule: if $C(s)$ is an atom, where $C$ is a concept name defined in $TC$ by the description $D$ (i.e., $C \equiv D$ is in $TC$), $compl(\mathcal{A}) \models C(s)$ if $D = C_1 \sqcap C_2$ and $compl(\mathcal{A}) \models C_1(s)$ and $compl(\mathcal{A}) \models C_2(s)$. In our example, $compl(\mathcal{A}) \models square(b)$ since $square \equiv rectangle \sqcap equal\_sides$ and $compl(\mathcal{A}) \models rectangle(b)$ and $compl(\mathcal{A}) \models equal\_sides(b)$.*

*Note that $\mathcal{A}$ entails the antecedents of $H_1$ and $H_2$ without them being all explicit in the initial $\mathcal{A}$.*

This example illustrates that traditional Horn clause inference mechanisms are inadequate to reason in CARIN knowledge bases. Another aspect concerns the fact that a CARIN KB may entail the disjunction of the antecedents of two clauses without entailing either of them (i.e., we cannot consider each clause in isolation as it is the case in the traditional ILP approaches).

The existential entailment provides the basis for a sound and complete reasoning algorithm for CARIN-$\mathcal{ALN}$ knowledge bases. For learning tasks, we use particular cases of the existential entailment, which are described in sections 4 and 5. The complexity results about these tasks are given in section 6.

## 3 The learning setting

Our learning framework is that of learning from interpretations [DR97]. Learning from interpretations upgrades the attribute value setting in that it allows a system to learn non recursive First Order Logic programs from ground sets of facts (interpretations). The implicit assumption is that each example is stand-alone, (i.e. it contains all necessary information for the learning task to perform) and that the coverage test of a hypothesis wrt an example can be extensional (local) yielding a coverage test wrt to a set of examples linear in the number of examples.

Let us focus here on the reformulation of this setting for a classification learning task, as presented in [BDRJ99]. Given :

- a set $C$ of classes (each class is labelled by a predicate $p\_i$, $1 \leq i \leq n$, all class predicates with the same arity)
- a set $E$ of classified examples (each example is of the form $(e, p_i(\bar{a_j}))$, where $p_i$ is the class of the example and $e$ is a set of ground facts where constants $\bar{a_j}$ occur
- a background knowledge $BK$

Find a hypothesis $H$ such that for all $(e, p_i(\bar{a_j}) \in E$

- $H \wedge e \wedge BK \models p_i(\bar{a_j})$, and
- $H \wedge e \wedge BK \not\models p_j(\bar{a_j})$, for $1 \leq j \leq n$ and $j \neq i$.

In this setting, and after the presentation of CARIN-$\mathcal{ALN}$, let us now specify now what are the languages of our examples, background knowledge and hypotheses.

### 3.1 Background knowledge

Our background knowldedge is represented as a CARIN-$\mathcal{ALN}$ knowledge base. The hierarchical part of the background knowledge is represented as a CARIN-$\mathcal{ALN}$ terminological component $TC$ (see section 2.1), and any knowledge that cannot be represented in the $TC$ language is represented in $R$, the Horn clause component of CARIN-$\mathcal{ALN}$. This includes, for instance, clauses that require existential variables, clauses that express equalities between role restrictions, or rules that express relationships among more than two objects (see example 4).

### 3.2 Examples

An example $e_i$ is represented in a ground fact component $A_i$. A ground fact component represents exactly one example. The set of classified examples is therefore a set of ground fact components representing the classified examples. Each example is a model of the set of clauses defining its class predicate. In addition, each example $e_i$ is associated to a tuple of constants $\bar{a_i}$ representing the instanciation of the target predicate for $e_i$ ($\bar{a_i}$ is called the target tuple of constants of $e_i$).

**Example 7** *$east\_train(X)$ is the target predicate, $e_1$ is represented by* $\mathcal{A}_1 = \{train(a), has\_car(a,b), car(b), square(b), has\_load(b,c), load(c), has\_car(a,d), car(d), ellipse(d), has\_load(d,e), load(e), \leq 2\ has\_car(a), \leq 1\ has\_load(b), \leq 1\ has\_load(d), same\_shape(c,e)\}$*, we associate to* $e_1$ $\bar{a_1} = a$.

### 3.3 Hypothesis language

A hypothesis is a set of Horn clauses described using the language of the Horn clause component and concluding on the target concept. This means that the learning task we address is not to extend $TC$, but to extend $R$ with new concept definitions. $TC$ is assumed to be static in this work.

**Example 8**
$H = \{east\_train(X) \leftarrow train(X), \geq 1\ has\_car(X), \forall\ has\_car. \leq 1\ has\_load(X), has\_car(X,Y), polygon(Y); east\_train(X) \leftarrow train(X), \leq 0\ has\_car(X)\}$

## 4 Testing Coverage of examples

We now address the problem of testing whether an example $e_i$, represented as a CARIN-$\mathcal{ALN}$ ground fact component, is covered by a hypothesis $H$ defining a target concept $p$, wrt a CARIN-$\mathcal{ALN}$ background knowledge $BK$ that includes a terminological component and a Horn rule component. This is the case if $p(\bar{a})$, where $\bar{a}$ is the tuple of constants associated to $e_i$, is logically entailed by $BK$. More formally:

**Definition 4 (Covers relation)** *Given a CARIN-$\mathcal{ALN}$ knowledge base $BK$ ($BK$ contains a terminological component $TC$ and a Horn rule component $R$), a hypothesis $H$ defining a target concept $p$, $e_i$ an example represented by the ground fact component $\mathcal{A}_i$ together with $\bar{a}_i$ its target tuple of constants, $H$ covers $e_i$ if and only if:*

$$BK \cup H \cup \mathcal{A}_i \models p(\bar{a}_i)$$

Testing coverage amounts to computing the canonical model(s) of $TC \cup R \cup H \cup \mathcal{A}_i$ and then to checking whether $p(\bar{a}_i)$ is satisfied in such model(s).

Existential entailment (see section 2.5), in the case in which $\beta$ is a ground fact component $\mathcal{A}_i$, a conjunction of ground facts representing an example, enables us to decide whether

$$\mathcal{A}_i, TC \models Q_1 \vee \ldots \vee Q_n$$

i.e., to decide whether the disjunction of a set of Horn clause bodies is entailed from $\mathcal{A}$ and $TC$[3].

We therefore need to reformulate the coverage problem as one that can be handled by existential entailment. One possibility for compute these canonical models is to handle first the background knowledge expressed in $TC$ as well as the logical rules defining the DL connectors and their interactions (it is done using the constraint system approach described in section 2.5), yielding a completed ground fact component *compl*$(\mathcal{A}_i)$. The second step is then to compute

[3] It has been shown in [LR98b] that entailing antecedents of CARIN rules is not complicated by the fact that they contain ordinary predicates in addition to concepts and roles

the models of $compl(\mathcal{A}_i) \cup R \cup H$. This can be done by a forward chaining (saturation)[4] of $compl(\mathcal{A}_i)$ given $\mathcal{R} \cup H$. Since the propagation and forward chaining algorithms[5] are correct and complete, we will obtain *every* logical consequences of $BK \cup H \cup \mathcal{A}_i$.

However, we just need to obtain logical consequences about $p(\bar{a_i})$. This is why we present in the next section a coverage computation guided by the target predicate, which can avoid to consider every rule of $R$. One method would be to try to prove $p(\bar{a_i})$ using clauses of $R \cup H$ and facts of $compl(\mathcal{A}_i)$. The coverage testing of a set of $n$ examples therefore leads us to unfold rules for each example (i.e., we will unfold $p(\bar{a_1}), \ldots, p(\bar{a_n})$). Another approach consists of applying first an unfolding process on $p(\bar{X})$ wrt clauses of $R \cup H$ independently of any example. This process generates the disjunction of all possible ways to prove $p(\bar{X})$ given $R \cup H$. We can see this step as a kind of pre-compilation of $\mathcal{R} \cup H$ wrt $p(\bar{X})$. This pre-compilation is made once for each target predicate definition, and it is then sufficient to compute for an example its completions wrt $\mathcal{TC}$ and to check whether these completions satisfy any of the disjuncts obtained in the previous step.

Re-using the arguments that took place some years ago in the Explanation based Learning community, we cannot state that one approach is better than the other independently of a particular CARIN $KB$ and in particular independently of any example presentation : in some cases the pre-compilation will cost a lot with few advantages, in other cases the pre-compilation may save of lot of computation time. In this paper, we choose to present the algorithm based on pre-compilation but the precise comparison of the two approaches is outside the scope of this paper.

### 4.1 Coverage test computation

The following coverage test algorithm uses algorithms described in [LR98b].

$coverage\_test(H, \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}, TC, R)$

1. Precompilation of $R \cup H$ wrt $p$: build the disjunction $Q_1 \vee \ldots \vee Q_n$ such that for all $\mathcal{A}_i$,
$$\mathcal{A}_i, TC \models Q_1 \vee \ldots \vee Q_n \text{ iff } BK, H, \mathcal{A}_i \models p(\bar{a_i})$$
2. Coverage test: for i $\in$ {1,...,n}, use the existential entailment algorithm in order to decide whether $\mathcal{A}_i, TC \models Q_1 \vee \ldots \vee Q_n$ or not.

More precisely, the first step amounts to partial evaluation [VHB88] of $p(\bar{X})$ wrt clauses of $R \cup H$, i.e., to compute (not guided by any example) all the ways to

[4] This is not a simple forward chaining since the evaluation of concept and role predicates requires the application of a special procedure (see example 6). In the remainder of the paper by forward chaining we intend forward chaining upgraded with this special evaluation procedure.

[5] Classification systems such as CLASSIC are usually implemented as forward-chaining inference systems.

prove $p(\bar{X})$. As partial evaluation is performed wrt $R \cup H$ and since CARIN does not allow concept or role atoms to appear in the conclusion of $R$ clauses, only ordinary predicates are "unfolded". As $R \cup H$ is non recursive, this process terminates, yielding a disjunction of $Q_i$. The DL part of Horn clauses obtained is left unchanged by this step : the saturation of $\mathcal{A}_i$ wrt $TC$ is done in the next step, as part of the existential entailment algorithm.

The goal of the second step is to check whether:
$\mathcal{A}_i, TC \models Q_1 \vee \ldots \vee Q_n$ where the $Q_i$s are the disjuncts obtained by the partial evaluation process.

Let $\mathcal{ADL}$ be the set of ground facts in $\mathcal{A}_i$ built on role and concept predicates and let $\mathcal{AR}$ be the set of ground facts in $\mathcal{A}_i$ built on ordinary predicates. The set of propagation rules $PR_{\mathcal{ALN}}$ is applied on the initial constraint system of $\mathcal{ADL}$. The obtained completions correspond to the set of models of $\mathcal{ADL} \cup TC$. Canonical models are computed and existential entailment checks whether one of the disjuncts obtained during at step 1 is satisfied in all the canonical models obtained. If it is the case, the hypothesis $H$ covers the example represented by $\mathcal{A}_i$, else $H$ does not cover the example.

**Example 9** *Let us consider $H$ after example 8 and $e_1$ after exemple 7, let $BK$ be $TC$ be described after example 3 and let $R$ be empty, $H_1$ covers $e_1$ since $TC \cup H \cup \mathcal{A}_1 \models east\_train(a)$. If we consider the ground fact component of example 6, $TC \cup H \cup \mathcal{A} \not\models east\_train(a)$ since $TC \cup H \cup \mathcal{A} \not\models \forall has\_car. \leq 1\ has\_load(a)$.*

**Example 10** *Consider $e$ as $\mathcal{A} = \{train(a), \geq 2\ has\_car(a), \forall has\_car.car(a), \forall has\_car.square(a)\}$. If $H = \{east\_train(X) \leftarrow train(X), \geq 1\ has\_car(X), \forall has\_car.polygon(X)$, $H$ covers $e$ wrt $TC$ and $R$ (empty).*
*If we now consider $R$ as in exemple 4 and $H = \{east\_train(X) \leftarrow train(X), \geq 1\ has\_car(X), has\_car(X,Y), car(Y), has\_car(X,Z), car(Z), same\_shape(Y,Z)$, $H$ covers $e$ wrt $TC$ and $R$.*

Another advantage in using the existential entailment algorithm instead of a simple Horn clause inference mechanism is that it allows one to detect when the *disjunction* of $Q_i$s is entailed by $TC \cup H \cup \mathcal{A}_i$, while none of the $Q_i$s are true in isolation (see [LR98b], p172 for a detailed explanation of using a forward chaining mechanism).

**Example 11** *Consider $e$ as $\mathcal{A} = \{train(a), has\_car(a,b), square(b)\}$ and $H = \{east\_train(X) \leftarrow train(X), has\_car(X,Y), rectangle(Y), \geq 2\ has\_load(Y)(H_1)$; $east\_train(X) \leftarrow train(X), has\_car(X,Y), equal\_sides(Y), \leq 4\ has\_load(Y)(H_2)\}$. $H$ covers $e$ wrt $TC$ and $R$ (empty) since $body(H_1) \vee body(H_2)$ is entailed while neither $body(H_1)$ nor $body(H_2)$ are entailed (no information about the number of loads of $b$ is present in $e$).*

## 5 Hypothesis ordering

Lastly, let us define the partial ordering that structures our search space. Roughly speaking, it is an extension of generalized subsumption [Bun88] when background knowledge is made of a set of non recursive horn clauses plus a terminological component.

A hypothesis is a set of Horn clauses described using the language of the Horn clause component of CARIN-$\mathcal{ALN}$ and concluding on the target concept. Intuitively, a hypothesis $H_g$ is more general than a hypothesis $H_s$ if it covers more examples. For simplicity, we only consider here conjunctive hypotheses. If $head(H_s) = head(H_g)$[6], as for generalized subsumption, we only need to compare hypotheses bodies (i.e., existentially quantified conjunctions of literals). More formally:

**Definition 5 (Hypothesis order)** *Let $H_g$ and $H_s$ be two hypotheses of the form:*

$$H_g = p(\bar{X}) \leftarrow p_1(\bar{Y_1}) \wedge \ldots \wedge p_m(\bar{Y_m})$$
$$H_s = p(\bar{X}) \leftarrow p'_1(\bar{Y'_1}) \wedge \ldots \wedge p_n(\bar{Y_n})$$

*$H_g$ is more general than $H_s$ wrt $TC$ and $R$ iff the body of $H_s$, together with $TC$ and $R$, logically entails the body of $H_g$:*

$$TC, R, (\exists \bar{Y'})p'_1(\bar{Y'_1}) \wedge \ldots \wedge p_n(\bar{Y_n}) \models (\exists \bar{Y})p_1(\bar{Y_1}) \wedge \ldots \wedge p_m(\bar{Y_m})$$

Once again, we can use a particular case of the existential entailment in order to determine whether a hypothesis is more general than another hypothesis. Let us recall that existential entailment decides whether $\beta \cup TC \models Q_1 \vee \ldots \vee Q_n$ where $\beta$ and $Q_i$ are existential sentences of the form: $(\exists \bar{Y})p_1(\bar{Y_1}) \wedge \ldots \wedge p_m(\bar{Y_m})$. In the case where $n = 1$, the existential entailment problem enables us to decide whether $\beta \cup TC \models Q_1$ (this subcase is refered to as *query containment* in [LR98b]).

Definition 5 states that we need to compare the bodies of $H_s$ and $H_g$, denoted $body(H_s)$ and $body(H_g)$[7], wrt background knowledge expressed as $TC \cup R$. The existential entailment algorithm enables us to decide whether $body(H_s) \cup TC \models body(H_g)$, i.e., to compare the two hypotheses wrt $TC$ but not wrt $R$.

Let us recall that the first step of the existential entailment algorithm consists of computing the completions wrt $TC$ of the left hand formula of the implication (a ground fact component in the coverage test, an existential sentence in the subsumption ordering problem, but the principle of the algorithm is similar). Let us call $compl(H_s)$ the existential sentence corresponding to one completion of $body(H_s)$ obtained as the application of the set of propagation rules $\mathcal{PR}_{\mathcal{ALN}}$ to $body(H_s)$. $compl(H_s)$ is, so to say, saturated according to $TC$.

It is now sufficient to apply forward chaining wrt $R$ on $compl(H_s)$ and to check whether the obtained sentence entails $body(H_g)$. $H_g$ is more general than $H_s$ if this is the case, otherwise not. Once again, the completeness and soundness of existential entailment (proved in [LR98b]) and of forward chaining enables us to have a sound and complete inference mechanism for checking subsumption between two hypotheses.

[6] We assume here that all hypotheses of the target concept are rectified, (i.e., they have the same head variables).

[7] $body(H_g)$ and $body(H_s)$ are existential sentences (the presence of ordinary predicates is not a problem).

**Example 12** *Let $TC$ be reduced to the sentence :*
$circle \equiv non\_polygon \sqcap has\_radius$
*Let $R$ be reduced to the Horn clause :*
$non\_polytrain(X) \leftarrow train(X), has\_car(X,Y), non\_polygon(Y).$
*Let $H_s$ be:* $east\_train(X) \leftarrow train(X), \geq 1 has\_car(X), \forall has\_car.circle(X).$
*Let $H_g$ be:* $east\_train(X) \leftarrow non\_polytrain(X).$

*$H_g$ is more general than $H_s$ since:*
$train(X), \geq 1 has\_car(X), \forall has\_car.circle(X) \cup TC \cup R \models non\_polytrain(X).$

## 6 Complexity study

It has been shown in [LR98b] that the time complexity of existential entailment is deterministic doubly exponential since this problem requires that all completions be checked, and there is, in the worst case, a doubly exponential number of completions. However, when the terminological language is $\mathcal{ALN}$ and not $\mathcal{ALCNR}$, the existential entailment problem is much simpler and can be optimized. For the problem of query expansion in CARIN-$\mathcal{ALN}$, which is a similar but slightly more complex problem than our coverage or subsumption test[8], [GLMC] exhibits a time complexity exponential in the depth of ordinary atoms unfolding and in the maximal size of concept expressions in the query. This shows that unless the Horn clause component is restricted, the complexity of both the coverage test and the subsumption checking in CARIN-$\mathcal{ALN}$ are dominated by the the complexity of the coverage test and the subsumption checking in Horn clause logic. This rough complexity study has to be further refined in the case of restricted Horn clause components (ij-determinate [MF90] or k-local [KL94]).

## 7 Related work and future directions

We have presented a coverage test and a subsumption ordering for learning from interpretations in a hybrid language that combines a simple description logic with horn clause logic.

LIFE ([AKP91]) is a language combining logic programming with $\psi$-terms used to represent sub-typing in record-like data structures. On the one hand, the variables in $\psi$-terms enable one to express co-reference constraints which cannot be expressed using $\mathcal{ALN}$. On the other hand, $\psi$-terms only allow attributes (i.e., functional roles), they are therefore not suited to express, for instance, number restrictions about role fillers. Note that adding the *same-as* constructor of CLASSIC would allow us to express limited co-reference constraints, however Cohen and Hirsh showed in [CH92] that the DL with only the $\sqcap$, $\forall$ and *same-as* constructors is not pac-learnable.

[8] Roughly speaking, query expansion amounts to computing all possible rewritings of a given query, given a terminological component $TC$ and a horn clause component $R$.

The field of Description Logics has raised increasing interest in the last few years in the Machine learning community. [KM94] presents a constructive induction program named KLUSTER that uses a DL formalism to represent concept definitions. This DL is provided with a *Least Common Subsumer* algorithm. KLUSTER upgrades a T-Box given examples represented as A-Boxes, but it does not learn in a hybrid language target concept language as we do, and does not handle either hybrid background knowledge, including a horn clause component.

Cohen and Hirsh [CH94b,CH94a] give theoretical and experimental results on the learnability of description logics. In particular, they prove that C-CLASSIC is PAC-learnable. From a practical point of view, the authors propose several algorithms that allow C-CLASSIC concepts to be learned from positive and negative examples of these concepts. The language of both concepts and examples is the terminological language of the DL. Note that this single trick representation leads to a loss of information since the examples are abstracted. For instance, let the example be defined by the initial description: $train(a), has\_car(a, b), circle(b)$, $has\_car(a, c), rectangle(c)$, the abstraction of the example in C-CLASSIC is $(train \sqcap \geq 2\ has\_car)$. In our approach, examples are represented in a ground fact component, which enables us to reason using the whole information.

In [CH94b,CH94a], the *covers* relation that specifies how hypotheses relate to examples is the subsumption relation: a hypothesis $H$ covers an exemple $e$ if and only if $e \sqsubseteq H$ (i.e., $H$ subsumes $e$). The aim is then to find a hypothesis $H$ that covers all positive examples (completeness) and none of the negative examples (consistency). As C-CLASSIC only contains a limited kind of disjunction (the ONE-OF connective), many target concepts of practical interest can not be expressed using a single term of C-CLASSIC. One way to overcome this limitation is to consider algorithms that learn a disjunction of terms rather than a single term. A hypothesis $H \equiv H_1 \vee H_2 \ldots \vee H_n$ then covers an example $e$ (represented as a concept) if and only if $\exists H_i \in H$, $e \sqsubseteq H_i$ (i.e., $H_i$ subsumes $e$).

However, in [FP96], Pitt and Frazier shows that the union of terms of C-CLASSIC can subsume a concept even though neither member of the union susbsumes this concept (let us recall that there is no closed-world assumption in DLs). Reasoning about the possibility of such interactions makes the problem of learning in this situation challenging. In our framework, the existential entailment enables us to take into account such interaction since clauses are not considered in isolation (see [LR98b]). Finally, Cohen and Hirsh suggest that learning systems based on description logics may prove to be a useful complement to ILP systems. The work presented here is a first tentative at combining the two frameworks.

The perspectives of this work are numerous. Now that the framework in CARIN-$\mathcal{ALN}$ has been set and the complexity results seems encouraging, an actual learning system can be developed, providing a solid base for experimentation. A lgg-based approach does seem, intuitively, the most appropriate direction to investigate, as input horn clauses to the lgg process, once completed by the different inference mechanisms described in the paper will be quite large. A more promising direction will be to design a data-driven top-down operator for CARIN-$\mathcal{ALN}$, as in [AR00] or [BCS98]. A more theoretical line of research will be to look for a (if any) restricted CARIN language that may be PAC-learnable.

**Acknowledgements** We wish to thank M.-Ch. Rousset, for the many discussions we had concerning the CARIN language and the existential entailment algorithm, as well as J-U Kietz for helpful discussions on previous versions of the paper. Last but not least, we are grateful to Lise Fontaine, who proof-read a nearly final version of the paper.

## References

[AKP91] H. Ait-Kaci and A. Podelski. Towards the meaning of LIFE. In J. Maluszynski and M. Wirsing, editors, *Proceedings on 3rd International Symposium on Programming Language Implementation and Logic Programming*, pages 255–274, Berlin, 1991.

[AR00] E. Alphonse and C. Rouveirol. Lazy propositionalisation for relational learning. In W. Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence*, Berlin, 2000. IOS Press. to appear.

[BCS98] P. Brézellec, M. Champesme, and H. Soldano. Tabata, a learning algorithm searching a minimal space using a tabu strategy. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence*, pages 420–424, Brighton, 1998. Wiley.

[BDRJ99] H. Blockheel, L. De Raedt, and B. Jacobs, N.and Demoen. Scaling up inductive logic programming by learning from interpretaions. *Data Mining and Knowledge Discovery*, 3(1):59–93, 1999.

[BPS94] A. Borgida and P. F. Patel-Schneider. Complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:278–308, 1994.

[Bra78] R.J. Brachman. A structural paradigm for representing knowledge. Technical Report 3605, BBN Report, 1978.

[Bun88] W. Buntine. Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, 36:375–399, 1988.

[CH92] W. W. Cohen and H. Hirsh. Learnability of the classic knowledge representation language. Technical report, ATT Bell Laboratories, New York, 1992.

[CH94a] W. W. Cohen and H. Hirsh. The learnability of the description logics with equality constraints. *Machine Learning*, 2(4):169–199, 1994.

[CH94b] W. W. Cohen and H. Hirsh. Learning the CLASSIC description logic: Theoretical and experimental results. In *International Conference on Knowledge Representation and Reasoning*, pages 121–133. 1994.

[DLNN91] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In J.A. Allen R. Fikes and E Sandewall, editors, *Principles of Knowledge Representation and Reasoning: 2nd International Conference*, pages 151–162, Cambridge, Mass., 1991.

[DLNS91] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. A hybrid system with datalog and concept languages. In E. Ardizzone, S. Gablio, and

F. Sorbello, editors, *Trends in Artificial Intelligence*, volume 549 of *Lecture Notes in Artificial Intelligence*, pages 88–97. Springer-Verlag, 1991.

[DR97] L. De Raedt. Logical settings for concept learning. *Artificial Intelligence*, 95:187–201, 1997.

[FP96] M. Frazier and L. Pitt. Classic learning. *Machine Learning*, 25:151–193, 1996.

[GLMC] F. Goasdoué, V. Lattès, and Rousset M.-Ch. The use of carin language and algorithms for information integration: The picsel system. *International Journal of Cooperative Systems*. submitted.

[Gon97] M-E. Goncalves. Handling quantifiers in ILP. In S. Muggleton, editor, *Proc. of the 6th International Workshop on Inductive Logic Programming*, pages 337–357. Springer Verlag, 1997.

[KL94] J-U. Kietz and M. Lübbe. An efficient subsumption algorithm for inductive logic programming. In William W. Cohen and Haym Hirsh, editors, *Proc. 11th International Conference on Machine Learning*, pages 130–138. Morgan Kaufmann, 1994.

[KM94] J. U. Kietz and K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14(2):193–217, 1994.

[LR98a] A. Levy and M.-Ch. Rousset. Verification of knowledge bases based on containment checking. *Artificial Intelligence*, 101(1-2), 1998.

[LR98b] A. Y. Levy and M.-Ch. Rousset. Combining horn rules and description logics in carin. *Artificial Intelligence*, 104:165–209, 1998.

[MF90] S. Muggleton and C. Feng. Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*, pages 368–381. Ohmsma, Tokyo, Japan, 1990.

[MMPS94] D. Michie, S. Muggleton, D. Page, and A. Srinivasan. To the international computing community: A new East-West challenge. Technical report, Oxford University Computing laboratory, Oxford,UK, 1994. URL: ftp://ftp.comlab.ox.ac.uk/pub/Packages/ILP/trains.tar.Z.

[NCVLRDR99] S.-H. Nienhuys-Cheng, W. Van Laer, J. Ramon, and L. De Raedt. Generalizing refinement operators to learn prenex conjunctive normal forms. In S Džeroski and P. Flach, editors, *Proc. of the 9th International Conference on Inductive Logic Programming*, pages 245–256. Springer Verlag, 1999.

[QDBK96] J. Quantz, G. Dunker, F. Bergmann, and I. Kellner. The flex system. Technical report, KIT-Report, Technische Universität, Berlin, Germany, 1996.

[Val84] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[VHB88] F. Van Harmelen and A. Bundy. Explanation based generalisation = partial evaluation. *Artificial Intelligence*, 36:401–412, 1988.

[WWB+93] 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 att bell network systems. In *Proceedings of the Innovative Applications of Artificial Intelligence Conferences*, pages 183–193, Menlo Park, California, 1993.