A Data-Preprocessing Method Enabling ILP-Systems to Learn CARIN-$\mathcal{ALN}$Rules

Jörg-Uwe Kietz

Swiss Life, Corporate Center, IT Research & Development,
8022 Zürich, Switzerland,
uwe.kietz@swisslife.ch,
http://research.swisslife.ch/~kietz

Abstract. This paper describes a method which enables the embedding of CARIN-$\mathcal{ALN}$ rule subsumption and learning into datalog rule subsumption and learning. On the theoretical side, this allows us to transfer the learnability results known for ILP to CARIN-$\mathcal{ALN}$ rules. On the practical side, this gives us a preprocessing method, which enables ILP systems to learn CARIN-$\mathcal{ALN}$ rules just by transforming the data to be analyzed. We show, that this is not only a theoretical result in a first experiment: learning CARIN-$\mathcal{ALN}$ rules with the ILP system CILGG from the standard ILP MESH-Design dataset.
keywords: description logics, reformulation, polynomial learnability

1 Introduction

CARIN was proposed by [Levy and Rouset, 1998] as a combination of the two main approaches to represent and reason about relational knowledge, namely description logic (DL) and first-order horn-logic (HL). In Inductive Logic Programming (ILP) learning first-order horn-logic is investigated in depth, for learning DLs there exist first approaches [Kietz and Morik, 1994; Cohen and Hirsh, 1994b] and theoretical learnability results [Cohen and Hirsh, 1994a; Frazier and Pitt, 1994]. Recently, it was proposed to use CARIN-$\mathcal{ALN}$as a framework for learning [Rouveirol and Ventos, 2000]. This is a interesting extension of ILP as $\mathcal{ALN}$ provides a new bias orthogonal to the one used in ILP, i.e. it allows all quantified descriptions of body-variables, instead of the existential quantified ones in ILP. This allows to handle the difficult indeterminate relations efficiently by abstracting them into a determinate summary. It also has atleast and atmost restrictions, which allow to quantify the amount of indeterminism of these relations. However, up to now there are neither practical nor theoretical results concerning learning the CARIN-$\mathcal{ALN}$ language.

This paper is intended to close this gap, by showing how CARIN-$\mathcal{ALN}$ learning can be embedded into first-order horn-logic learning as done by ILP-methods. Even, if DL and HL have been shown to be quite incomparable concerning their expressive power [Borgida, 1996] with respect to their usual semantic interpretation of primitive concepts and roles we show that reasoning in DL can be simulated by reasoning in horn logic. A simple invertible function translates normalized concept descriptions into horn clauses using new predicates with an external semantics (as Borgida has show they are not expressible in horn logic) to represent the DL terms. The aim of this translation, of course, is not to provide another algorithm to do deductive reasoning, i.e. subsumption, equivalence and satisfiability checking, but to show how inductive (i.e. ILP) methods can be used to learn $\mathcal{ALN}$ concept descriptions and CARIN-$\mathcal{ALN}$ rules. This also allows us to transfer the known boarder-line of polynomial learnability for ILP to CARIN-$\mathcal{ALN}$.

In section 2 we describe the description logic $\mathcal{ALN}$ and define the basis of our translation into horn logic. In section 3 we extend that translation to CARIN-$\mathcal{ALN}$. In section 4, we investigate in a local closed-world-assumption (CWA) to overcome the inadmissibility (in the sense of [Shapiro, 1983]) of ground HL facts to learn CARIN-$\mathcal{ALN}$-rules. In section 5, we characterize the boarder line of polynomial learnability of CARIN-$\mathcal{ALN}$ rules using that translation to horn logic. Finally in section 6, we demonstrate with a first experiment, that this transformation can be used to learn CARIN-$\mathcal{ALN}$ rule using normal ILP-systems on extended, i.e. preprocessed data-sets.

2 The Description Logic $\mathcal{ALN}$

Starting with KL-ONE [Brachman and Schmolze, 1985] an increasing effort has been spent in the development of formal knowledge representation languages to express knowledge about concepts and concept hierarchies. The basic building blocks of description logics are concepts, roles and individuals. Concepts describe the common properties of a collection of individuals and can be considered as unary predicates which are interpreted as sets of objects. Roles are interpreted as binary relations between objects. A whole family of knowledge representation systems, e.g. NIKL [Moser, 1983], KL-TWO [Vilain, 1985], KRYPTON [Brachman et al., 1985], BACK [von Luck et al., 1987; Peltason et al., 1989], CLASSIC [Borgida et al., 1989], FLEX [Quantz et al., 1996] FACT [Horrocks, 1998], have been built using these languages and for most of them complexity results for the main reasoning tasks are known.

Each description logic defines also a number of language constructs that can be used to define new concepts and roles. In this paper we use the very basic[1] description logic $\mathcal{ALN}$ under its normal open-world (OWA) semantics, with the language constructs in table 1. P denotes a primitive concept, i.e. an unary predicate, R denotes a primitive role, i.e. a binary predicate, n is a positive integer and the C_i are concept terms, i.e. anything in the left row of the table. These language constructs can be used to build complex terms, e.g. the term $\mathit{train} \sqcap \forall\ \mathit{has\_car}.(\mathit{car} \sqcap \leq 0\ \mathit{has\_load})$ can be used to define empty trains in Michalski's well-known train domain.

The main reasoning tasks with description logic terms are subsumption, equivalence and satisfiability checking. We do not consider classification and

[1] See http://www-db.research.bell-labs.com/user/pfps/papers/krss-spec.ps for further language constructs considered in description logics.

Term (Math)	Term (Classic)	Interpretation
$\top$	**everything**	Δ^I
$\bot$	**nothing**	$\emptyset$
P	P	P^I
$\neg P$	**not** P	$\Delta^I \setminus P^I$
$C_1 \sqcap \ldots \sqcap C_n$	**and** $C_1 \ldots C_n$	$C_1^I \cap \ldots \cap C_n^I$
$\forall R.C$	**all** R C	$\{x \in \Delta^I \mid \forall y \in \Delta^I :< x, y >\in R^I \Rightarrow y \in C^I\}$
$\geq nR$	**atleast** n R	$\{x \in \Delta^I \mid \ \|\{y \in \Delta^I \mid < x, y >\in R^I\}\| \geq n\}$
$\leq nR$	**atmost** n R	$\{x \in \Delta^I \mid \ \|\{y \in \Delta^I \mid < x, y >\in R^I\}\| \leq n\}$

Table 1. Concept terms in $\mathcal{ALN}$and their model-theoretic interpretation

instance checking as we neither use a terminological nor an assertional component in this paper. From the learning point of view the least common subsumer (lcs) operation is also important.

Definition 1 (subsumption, equivalence, satisfiability and least common subsumer). *The concept description D subsumes the concept description C ($C \sqsubseteq D$) iff $C^I \subseteq D^I$ for all interpretations I; C is satisfiable if there exists an interpretation I such that $C^I \neq \emptyset$; C and D are equivalent ($C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$; and E is the least common subsumer (lcs) of C and D, iff $C \sqsubseteq E$ and $D \sqsubseteq E$ and if there is an E' with $C \sqsubseteq E'$ and $D \sqsubseteq E'$, then $E \sqsubseteq E'$.*

We have chosen $\mathcal{ALN}$, because subsumption checking between $\mathcal{ALN}$ descriptions is polynomial [Donini et al., 1991], and in this paper we always consider an empty terminological component since subsumption checking between $\mathcal{ALN}$ terms with respect to an acyclic terminological component is coNP-Complete [Nebel, 1990][2]. A polynomial time algorithm for the lcs computation of $\mathcal{ALN}$ concepts can be found in [Cohen et al., 1992].

$\mathcal{ALN}$ descriptions are in general not normalized, i.e. there exist several possibilities to express the same concept, e.g. $\bot \equiv (A \sqcap \neg A) \equiv (\geq 2R \sqcap \leq 1R)$. However, they can be normalized, e.g. by the following set of rewrite rules.

Definition 2 (Normalization of $\mathcal{ALN}$ descriptions). *$norm(C) = C'$ iff $sorted(C) \mapsto \ldots \mapsto C'$ and no rewrite rule is applicable to C', i.e as a first step any conjunction $(C_1 \sqcap \ldots \sqcap C_n)$ is replaced by its sorted equivalent for a total order $<$ that respects $(P_1) < (\neg P_1) < \ldots < (P_n) < (\neg P_n) < (\geq n_1 R_1) < (\leq m_1 R_1) < (\forall R_1.C_1) < \ldots < (\geq n_n R_n) < (\leq m_n R_n) < (\forall R_n.C_n)$. Then the following rewrite rules are applied to all conjunctions not only the top-level one, as long as possible.*

1. $(C_1 \sqcap \ldots \sqcap C_n) \mapsto \bot$, if any $C_i = \bot$
2. $(C_1 \sqcap \ldots \sqcap C_n) \mapsto C_1 \sqcap \ldots \sqcap C_{i-1} \sqcap C_{i+1} \sqcap \ldots \sqcap C_n$, if any $C_i = \top$

[2] Practical experiences with DL systems have shown that this normally does not matter in practise, i.e. in practical applications the results of the paper can be used with a filled terminological component by using the - only in the worst-case - exponential larger expanded concept-terms as input.

3. $(P \sqcap \neg P) \mapsto \bot$
4. $(\leq nR \sqcap \geq mR) \mapsto \bot$, if $n < m$.
5. $(C \sqcap C) \mapsto C$
6. $(\geq 0R) \mapsto \top$
7. $(\geq n_1 R \sqcap \geq n_2 R) \mapsto \geq maximum(n_1, n_2)R$
8. $(\leq n_1 R \sqcap \leq n_2 R) \mapsto \leq minimum(n_1, n_2)R$
9. $(\leq 0R \sqcap \forall R.C) \mapsto \leq 0R$
10. $(\forall R.\bot) \mapsto \leq 0R$
11. $(\forall R.\top) \mapsto \top$
12. $(\forall R.C_1 \sqcap \forall R.C_2) \mapsto \forall R.(merge^3(C_1 \sqcap C_2))$

Lemma 1. *For any two $\mathcal{ALN}$ concept descriptions C_1 and C_2: $C_1 \equiv C_2$, iff $norm(C_1) = norm(C_2)$ and $norm(C)$ can be computed in $O(n\ log\ n)$, with n being the size of C.*

Proof. It is easy to verify, that all these rewrite rules produce equivalent concepts, i.e. are sound (if $norm(C_1) = norm(C_2)$, then $C_1 \equiv C_2$), The other direction, i.e. their completeness (if $C_1 \equiv C_2$, then $norm(C_1) = norm(C_2)$), is less obvious, but follows from the fact that all primitive concepts P_i are independent and after normalization occur at most once inside any conjunction. The primitive roles R_i are independent as well and for every role there is at most one set of consistent $\leq n_i R_i$, $\geq m_i R_i$ and $\forall R_i.C_i$ inside any conjunction. The complexity $O(n\ log\ n)$ is the worst case complexity of sorting C. Anything else can be done during a linear scan of the sorted C as all rules either delete the whole conjunction or an $\top$ inside, or apply to the current term and either it's left or right neighbor and they always produce a term that is sorted if inserted in place of the left-hand-side terms. All rules except rule 10 reduce the size atleast by 1, i.e. they are applicable atmost n times and rule 10 is also applicable at most once for any source literal, i.e. there are atmost $2n$ rule applications. □

As well known in DL, this provides also polynomial $O(n\ log\ n)$ algorithms for subsumption checking ($C \sqsubseteq D$, iff $norm(C \sqcap D) = norm(C)$); and satisfiability ($C$ is satisfiable, iff $norm(C) \neq \bot$).

Definition 3 ($\mathcal{ALN}$ translation to horn logic).
$\Phi(C) = h(X) \leftarrow \Phi(norm(C), X)$.
$\Phi(\bot, X) = \bot(X)$
$\Phi(P\ \{\sqcap\ C\}, X) = cp_P(X)\{, \Phi(C, X)\}$
$\Phi(\neg P\ \{\sqcap\ C\}, X) = cn_P(X)\{, \Phi(C, X)\}$
$\Phi(\geq nR \sqcap \leq mR \sqcap \forall R.C_R\ \{\sqcap\ C\}, X) = rr_R(X, [n..m], Y), \Phi(C_R, Y)\{, \Phi(C, X)\}$
$\Phi(\leq mR \sqcap \forall R.C_R\ \{\sqcap\ C\}, X) = rr_R(X, [0..m], Y), \Phi(C_R, Y)\{, \Phi(C, X)\}$
$\Phi(\geq nR \sqcap \forall R.C_R\ \{\sqcap\ C\}, X) = rr_R(X, [n..*], Y), \Phi(C_R, Y)\{, \Phi(C, X)\}$
$\Phi(\geq nR \sqcap \leq mR\ \{\sqcap\ C\}, X) = rr_R(X, [n..m], Y)\{, \Phi(C, X)\}$
$\Phi(\forall R.C_R\ \{\sqcap\ C\}, X) = rr_R(X, [0..*], Y), \Phi(C_R, Y)\{, \Phi(C, X)\}$

[3] Merging two sorted lists into one sorted list. During that linear (in the size of the conjunctions) process these rules should be applied as well.

$\Phi(\leq mR\ \{\sqcap C\}, X) = rr_R(X, [0..m], Y)\{, \Phi(C, X)\}$
$\Phi(\geq nR\ \{\sqcap C\}, X) = rr_R(X, [n..*], Y)\{, \Phi(C, X)\}$
where Y is always a new variable not used so far and $\{\sqcap C\}$ means, if there are conjuncts left, recursion on them $\{, \Phi(C, X)\}$ has to continue.

Let us illustrate our translation function on our train example:
$\Phi(train \sqcap \forall\ has\_car.(car \sqcap \leq 0\ has\_load)) =$
$h(X) \leftarrow cp\_train(X)$, $rr\_has\_car(X, [0..*], Y)$, $cp\_car(Y)$, $rr\_has\_load(Y, [0..0], Z)$, $\bot(Z)$. Note, that this clause has a very different meaning than the clause $h(X) \leftarrow train(X)$, $has\_car(X, Y)$, $car(Y)$, $has\_load(Y, Z)$. The first one must be interpreted as being true for every **set of empty trains**, whereas the second one is true for every **single train with a load**. That means, that the predicates (and variables) in the first clauses are meta-predicates (set-variables) over the ones in the second clause (individual variables), e.g. like `findall` in Prolog with a specific call using predicates of the second clause. This difference becomes especially important in CARIN-$\mathcal{ALN}$, i.e. in the next section, where both kinds of literals can occur in a single clause.

Nevertheless, we are now nearly able to simulate DL subsumption with θ-subsumption and lcs with lgg. There are only two very small and easy extensions of θ-subsumption and lgg (to $\theta_{I\bot}$-subsumption and $lgg_{I\bot}$) needed:

- The handling of subterms representing intervals of numbers, e.g. a term like $[0..*]$ should θ_I-subsume a term like $[1..5]$. More precisely an interval $[Min_1..Max_1]$ θ_I-subsumes an interval $[Min_2..Max_2]$, iff $Min_1 \leq Min_2$ and $Max_2 \leq Max_1$; and the lgg_I of two intervals $[Min_1..Max_1]$ and $[Min_2..Max_2]$ is the interval $[minimum(Min_1, Min_2)..maximum(Max_1, Max_2)]$[4].
- The handling of nothing, i.e. $\bot(X)$ should be $\theta_\bot$-subsumed by $\Phi(C, X)$ for any concept description C, e.g. by any subclause containing the relation-chains starting with X; and the $lgg_\bot$ of $\bot(X)$ and $\Phi(C, X)$ is $\Phi(C, X)$.

Theorem 1 (Simulation of subsumption and lcs). *A concept description C subsumes a concept description D ($D \sqsubseteq C$), if and only if the translation of C $\theta_{I\bot}$-subsumes the translation of D ($\Phi(C) \vdash_{\theta_{I\bot}} \Phi(D)$), and $lcs(C, D) \equiv \Phi^{-1}(lgg_{I\bot}(\Phi(C), \Phi(D)))$*

This theorem follows from the similarity between the translation Φ and $\theta_{I\bot}$-subsumption and the structural subsumption algorithm given in [Cohen et al., 1992]. There are some more nice properties for learning:

- Any clause which subsumes a set of such clauses, does so deterministically [Kietz and Lübbe, 1994], i.e. is determinate [Muggleton and Feng, 1992], as any $rr_R(X, I, Y)$ occurs just once for any variable X.

[4] I use that extension, as it was already present in CILGG [Kietz, 1996] to generalize numbers. The normal way for ILP is to provide the built-in predicate $\leq$, e.g. in FOIL and PROGOL, and use a translation into $rr_R(X, n_{atleast}, m_{atmost}, Y)$, together with learning c_{min} and c_{max} in $c_{min} \leq n_{atleast}, m_{atmost} \leq c_{max}$.

- The relation chains in the clause (i.e. the chains of $rr_R(X, I, Y)$ literals) are not only acyclic but even tree-structured (as the DL-Term is a tree), i.e. subsumption (coverage test), learning, and propositionalisation are very easy, e.g. the depth limit i needed for the polynomial learnability of (cyclic) ij-determinate clauses is not needed.
- Φ is a bijective, invertible function, i.e $\Phi^{-1}(\Phi(C)) \equiv C$, i.e. it allows to retranslate generalized (i.e. learned) clauses: If D is a linked clause[5], and $D \vdash_{\theta_{I\perp}} \Phi(C)$ for any $\mathcal{ALN}$description C, then $\Phi^{-1}(D)$ is totally invertible and produces a valid description logic term.

3 Carin-$\mathcal{ALN}$ rules and Horn Logic

Carin as proposed in [Levy and Rouset, 1998] combines first-order function-free horn logic with description logic by allowing description logic terms as body literals[6] in horn rules. Concept terms represent unary predicates and role-terms[7] represent binary predicates. The direct use of primitive concepts and roles is indistinguishable from the use of unary and binary predicates in FOL, but the use of concept terms, i.e. descriptions contain all, atleast and atmost adds expressive power to the language. Here is an example of a Carin-$\mathcal{ALN}$ rule using this expressive power. Note, that this cannot be expressed in neither horn logic nor $\mathcal{ALN}$alone.

$$\begin{aligned} \mathit{east\_bound\_train}(X) \leftarrow\ & \mathit{has\_car}(X,Y),\ \mathit{has\_car}(X,Z), \mathit{same\_shape}(Y,Z), \\ & (\mathit{train} \sqcap \leq 2\ \mathit{has\_car} \sqcap \forall\ \mathit{has\_car}.(\mathit{car} \sqcap \leq 0\ \mathit{has\_load}))(X). \end{aligned}$$

This rule also contains implicit knowledge due to interactions between description logic literals (DL-literals) and normal literals (HL-literals[8]), i.e. the following literals can be deduced to be true in every model of the rule: $\mathit{car}(Y)$, $\mathit{car}(Z)$, $(\geq 1\ \mathit{has\_car})(X)$, $(\leq 0\ \mathit{has\_load})(Y)$, $(\leq 0\ \mathit{has\_load})(Z)$. We have to make such implicit literals explicit, as a reasoning procedure based on substructure matching like θ-subsumption would be incomplete otherwise, e.g. $h(X) \leftarrow a(X,Y) \Leftrightarrow h(X) \leftarrow (\geq 1a)(X)$, but $h(X) \leftarrow a(X,Y) \not\vdash_{\theta_{I\perp}} h(X) \leftarrow \Phi(\geq 1a, X)$ $h(X) \leftarrow \Phi(\geq 1a, X) \not\vdash_{\theta_{I\perp}} h(X) \leftarrow a(X,Y)$.

If the rules were ground, the interaction between HL- and DL-terms would correspond to what is called ABox reasoning in description logic, i.e. inferring HL-literals corresponds to ABox completion [Baader and Sattler, 2000] and inferring DL-literals corresponds to computing the most specific concept [Baader and Küsters, 1998] for every variable. Goasdoué, Rouveirol and Ventos [2001] have

[5] There is no partition of literals such that one partition does not share atleast one variable with any other partition

[6] The head literal must have a predicate, which must not be a terminological concept of role.

[7] In $\mathcal{ALN}$ there are only primitive roles, but other DLs also have operators for role terms.

[8] Primitive roles and concepts strictly belong to both classes, but we will reference and handle primitive roles and primitive concepts as HL-literals

put them together as completion rules to formalize example saturation for learning under OI-subsumption. The adaptation of their rules to θ-subsumption lead to the following completion rules to be applied on the body B of CARIN-$\mathcal{ALN}$-rules $H \leftarrow B$.

1. **apply atleast restriction:** if there exists a substitution σ such that $((\geq n\ r)(X_0))\sigma \in B$ for a $n \geq 1$ and $(\{r(X_0, X_1)\})\sigma \notin B$ then $B := B \cup (\{r(X_0, \alpha)\})\sigma$, and α is a new unique constant.
2. **apply value restriction:** if there exists a substitution σ such that $(\{(\forall r.C)(X_0), r(X_0, X_1)\})\sigma \subseteq B$ and $C(X_1)\sigma \notin B$ then $B := B \cup \{C(X_1)\sigma\}$.
3. **infer atleast restriction:** If there exists a substitution σ such that
   - $(\{r(X_0, X_1), \ldots, r(X_0, X_n)\})\sigma \subseteq B$, and
   - $((\geq n\ r)(X_0))\sigma \notin B$ and
   - for all pairs $\{X_i, X_j\}\sigma \subseteq \{X_1, \ldots, X_n\}\sigma$, $i \neq j$ either $X_i\sigma$ and $X_j\sigma$ are not unifiable terms, or $\{C_i(X_i), C_j(X_j)\}\sigma \subseteq B$ and $C_i \sqcap C_j \equiv \bot$,

   then $B := B \cup (\{(\geq n\ r)(X_0)\})\sigma$.
4. **infer value restriction:** If there exists a substitution σ such that $(\{(\leq n\ r)(X_0), r(X_0, X_1), \ldots,\ r(X_0, X_n), C_1(X_1), \ldots,\ C_n(X_n)\})\sigma \subseteq B$ and $((\forall r.C)(X_0))\sigma \notin B$ then $B := B \cup (\{(\forall r.lcs(C_1, \ldots, C_n)(X_0)\})\sigma$,
5. **collect description logic terms over the same variable:** if $C(X_0) \in B$ and $D(X_0) \in B$ with C and D are DL-terms then $B := B \backslash \{C(X_0), D(X_0)\} \cup (C \sqcap D)(X_0)$

Theorem 2 (Simulation of subsumption). *Let depth(C) the depth of the deepest $\forall$ nesting in C, let $\varphi(D, i)$ denote the completion of D up to depth i, i.e. the application of the rules 1-5 as often as possible, without generating a $\forall$ nesting deeper than i and Φ_r the extension of Φ to rules, such that all DL-literals $DL(X)$ in the rule are translated with $\Phi(norm(DL), X)$ and everything else is returned unchanged. A non-recursive* CARIN-$\mathcal{ALN}$ *rule C implies a non-recursive* CARIN-$\mathcal{ALN}$ *rule D ($C \models D$), if and only if the translation Φ_r of C $\theta_{I\perp}$-subsumes the translation Φ_r of the completion φ of D ($\Phi_r(C) \vdash_{\theta_{I\perp}} \Phi_r(\varphi(D, depth(C)))$).*

Proof. For a set of CARIN-$\mathcal{ALN}$ literals like $\{(\geq 2r \sqcap \forall r.(\geq 2r) \sqcap \ldots \sqcap \underbrace{\forall r. \ldots \forall r}_{n}.(\geq 2r))(a)\}$, we don't generate the whole (exponential in n) binary tree with rule 1, but just one path from the root to a leave (linear in n), as all these paths in the tree are indistinguishable under θ-subsumption, and any clause without these new unique constants subsumes the clause with the whole tree, if and only if it subsumes the clause with only one path. Also the application of rule 2 and rule 3 is polynomial (depth of DL-terms times number of literals) in the length of the CARIN-$\mathcal{ALN}$ rule, if applied to ground facts. Rule 3 is more complex in the case of variables, but we do not need that. The unrestricted application of rule 4 may lead to an infinite term for cyclic FOL-literals, e.g. $h(X) \leftarrow r(X, X), (\leq 1r)(X)$ implies the infinite term $\leq 1r \sqcap \forall r.(\leq 1r \sqcap \forall r.(\leq 1r \sqcap \forall r.(\ldots)))(X)$. In general there are two solutions, the use of cyclic definitions in the TBox as in [Baader and Küsters, 1998], or the one we use as we do not have a TBox, to restrict the depth of the terms to be generated. This does not lead to an incompleteness of the subsumption test if we choose the depth with respect to the clause we test subsumption against, i.e. greater than the deepest present DL-term. Rule 4 is polynomial (number of literals times depth limit i) as well. Rule 5 is needed to collect all terms together for normalization. $\square$

As Φ and φ are polynomial for ground clauses, the complexity of $C \models D$ is that of θ-subsumption, if C is ground as in our learning setting. From [Kietz and Lübbe, 1994] we know, that θ-subsumption is NP-complete in general and polynomial for an arbitrary horn clause D, if $C = C_0 \leftarrow C_{DET}, LOC_1, \ldots, LOC_n$ is a horn clause where $C_0 \leftarrow C_{DET}$ is determinate with respect to D and each $LOC_i, 1 \leq i \leq n$, is a k–local[9]. In that case, $C \vdash_\theta D$ can be decided with $O(\|C\| * \|C_{DET}\| * \|D\| + \|LOC_1, \ldots, LOC_n\|^2 + n * (k^k * \|D\|))$ unification attempts.

Theorem 3 (Simulation of lgg). *Let $E = \Phi_r^{-1} lgg_{I\perp}(\Phi_r(\varphi(C, i)), \Phi_r(\varphi(D, i)))$. E is the least general generalization (lgg) with depth atmost i of C and D, i.e. $E \models C$ and $E \models D$ and if there is an E' with depth at most i with $E' \models C$ and $E' \models D$, then $E' \models E$.*[10]

4 The admissibility of ground facts to learn Carin-$\mathcal{ALN}$

Note, that under the open-world-assumption (OWA) used in description logics and in this paper up to this point, no atmost restriction can be inferred from HL-literals. Therefore it is impossible to learn atmost and value restrictions from HL-literals only under the open-world-assumption used in the completion rules 1-5. Even if there are only n fillers of a role r present, we do not infer $\leq nr$ and if the n role fillers share a common property C, we only infer $\forall r.C$, if an $\leq nr$ is already present. This, of course, is not a deficiency of the rules, but a consequence of the semantics, i.e. as long as no atmost restriction forbid further role fillers, there are models with more role fillers, and therefore the atmost restriction is not true in all models and must not be inferred. To summarize ground HL-literals are not admissible [Shapiro, 1983] to learn Carin-$\mathcal{ALN}$ rules under OWA. There are two ways out. We require that the user provides the necessary facts about atmost restrictions. We assume a (locally) closed world (CWA), i.e. we change the semantics, such that the needed atmost restrictions can be deduced.

The CWA is quite natural for learning as learning can be formalized as deduction under CWA [Helft, 1989]. We do not want to close the world totally, but just locally as in Kluster [Kietz and Morik, 1994], i.e we assume that if an object is described at all it is described completely (as far as relevant for learning), but we don't want to assume that we know about all objects of any world. Formally, this can be seen as looking at all models which describe all

[9] It does not share variables with another local, which not also occur in C_0 or C_{DET} and it has atmost k literals (k–literal local) or atmost k variables (k–variable local)

[10] Without depth limit the lgg may not exist, i.e. is infinite due to rule 4.

objects in the minimal models (i.e. the models considered under CWA) as they are described in the minimal model, but may contain and describe other objects as well. The rules 1-5 are correct, but (as intended) incomplete for this revised semantic. The following two non-monotonic rules are missing to complete sets of ground facts (B here denotes the set of ground facts in the background knowledge) under local CWA. Theorem 2 and 3 hold correspondingly with rule 1-7 for $\models_{lCWA}$.

6. **infer atmost restriction:** If there exists a substitution σ, such that
   - $(\{r(X_0, X_1), \ldots, r(X_0, X_n)\})\sigma \subseteq B$, and
   - for all pairs $\{X_i, X_j\}\sigma \subseteq \{X_1, \ldots, X_n\}\sigma$, $i \neq j$, $X_i\sigma$ and $X_j\sigma$ are not unifiable terms, and
   - there is no $c \notin \{X_1, \ldots, X_n\}\sigma$ such that $r(X_0, c)\sigma \in B$, and
   - m is the greatest m such that $(\geq m\ r)(X_0))\sigma \in B$ or 0 if none is present,

   then $B := B \cup (\{(\leq maximum(n, m)\ r)(X_0)\})\sigma$.
7. **infer concept negation:** Let c be a constant appearing in some literal of B, and P be any primitive concept, if $P(c) \notin B$, then $B := B \cup \neg P(c)$.

Given N constants (or variables) in the rule, and M primitive concepts, only $N * M$ additional literals are introduced at the literal level by the local CWA rule 7, i.e. only a polynomial amount. An $\forall r.\neg P$ can only be added, by rule 4 out of the introduced literals. But $\forall r.\neg P$ does not follow for every role under CWA, i.e under CWA only the literal $\neg P(b)$ is added by rule 7 to $R(a, b), R(a, c), P(c)$, but $(\forall r.\neg P)(a)$ is obviously not true and not added. Theorem 2 proves polynomial complexity in the size of the input of rule 4 under OWA. Therefore the complexity under CWA is polynomial as well.

5 Learning Carin-$\mathcal{ALN}$

We use the normal ILP definition of learning to describe learning in Carin-$\mathcal{ALN}$.

Definition 4 (Learning Problem). *Given: a logical interpretation with a consequence relation $\models$, background knowledge B in a Language LB, positive and negative examples $E = E^+ \cup E^-$ in a language LE consistent with B (B,E $\not\models \Box$) and not a consequence of B (B $\not\models$ E), and a hypothesis language LH. Find a hypothesis $h \in LH$ such that:*

(I) (B,h,E $\not\models \Box$), h is consistent with B and E.
(II) (B,h $\models E^+$), h and B explain E^+.
(III) (B, h $\not\models E^-$), h and B do not explain E^-.

The tuple $(\models, LB, LE, LH)$ is called the learning problem. *Deciding whether there exists such an $h \in LH$, is called the* consistency problem. *An algorithm which accepts any $B \in LB$ and $E \in LE$ as input and computes such an $h \in LH$ if it exists, or "no" if it does not exist is called a* learning-algorithm.

With this definition of learning and theorem 2 we can immediately conclude the learnability of Carin-$\mathcal{ALN}$ rules.

Theorem 4. *The* CARIN-$\mathcal{ALN}$ *rule learning problems* ($\models_{OWA}$ *,ground* HL&DL $facts$*, ground* HL $facts$*,* CARIN-$\mathcal{ALN}$ *rules) resp.* ($\models_{ICWA}$ *,ground* HL $facts$*, ground* HL $facts$*,* CARIN-$\mathcal{ALN}$ *rules)) are polynomially learnable, if and only if the corresponding ILP learning problems* ($\models$, $\Phi_{r1-5}(\varphi$*(ground* HL&DL $facts$*,i)*, *ground* HL $facts$*,* $\Phi_r(\varphi($CARIN-$\mathcal{ALN}$ *rules,i)*) *resp* ($\models$, $\Phi_{r1-7}(\varphi$*(ground* HL $facts$*,i), ground* HL $facts$*,* $\Phi_r(\varphi($CARIN-$\mathcal{ALN}$ *rules,i)*)*is polynomial learnable.*

As the boarder line of polynomial learnability is quite well known for ILP [Kietz, 1996] due to a lot of positive and negative learnability results (see [Kietz and Džeroski, 1994; Cohen and Page, 1995] for overviews), we are now able to characterize it for CARIN-$\mathcal{ALN}$ rules as well. One of the most expressive (single horn clause) ILP-problem that is polynomial learnable, is learning a ij-determinate-k-literal-local horn clause (see [Kietz and Lübbe, 1994] for definitions and discussions of these restrictions). So a CARIN-$\mathcal{ALN}$ rule hypothesis space is polynomial learnable, if its translation produces such horn clauses. As shown in section 2 the translation of every DL-literals produces a (acyclic, tree-structured) determinate subclause, the subclause is determinate with respect to the overall clause, if it starts with a determinate variable. We can conclude, that a CARIN-$\mathcal{ALN}$ rule is polynomially learnable, if the HL-literals are ij-determinate-k-literal-local and the DL-literals occur only on determinate variables. If a DL-literal occurs on an indeterminate variable, the number of primitive roles and concepts in it count against the k of the k-literal-local restriction. The depth bound i of the DL-terms, is only needed to ensure the finiteness of the DL-terms, it is not necessary to fix it for polynomial learnability, i.e. does not count against the i in ij-determinate due to the acyclic tree-structure of the translation.

6 Learning MESH-Design in CARIN-$\mathcal{ALN}$

This translation does not only produce theoretical results it also works in practise. Let us demonstrate this with the ILP dataset (available from MLNet) for MESH-Design [Džeroski and Dolsak, 1992]. We have chosen MESH-Design as it only contains unary and binary predicates, i.e. perfectly fits description logic without the need for further preprocessing. We have chosen CILGG [Kietz, 1996] as the ILP-systems to learn the translated descriptions as it has interval handling, is optimized for determinate literals (e.g. like Golem [Muggleton and Feng, 1992]) and is able to learn k-literal-local clauses completely for small k.

As depth limit we had chosen 1 for HL-literals. In MESH-Design this produces 4-literal-local ground starting (bottom) clauses. We restricted DL-terms to the head variable/object as this is the only determinate one and to the depth of 3, as deeper terms are very difficult to understand. We have made three experiments, one with only the literals generated from the DL-term, one with only the HL-literals, and one with both (CL). CILGG [Kietz, 1996] has two possibilities to use the learned rules for testing, the normal deductive one for most general discriminations (generated similar as in Golem from the learned lggs), and a k-nearest neighbor classification (20-NN in this case) using the learned lggs. The results are in table 2 together with the results reported in [Džeroski and Dolsak, 1992] (i.e. their average CPU-Time is measured on computers from 1992!) and Indigo [personal note from P. Geibel, 1996]. The table gives the number of correctly classified edges per object using rules learned from the other objects. The results seem to indicate that the extended language helps to learn better rules in MESH-Design. This one experiment however says nothing about the **usefulness** of CARIN-$\mathcal{ALN}$as HL, it only shows, that the theoretical translation defined in this paper is **applicable** in practical applications as well.

	A	B	C	D	E	Σ	%	Avg.
Maximum	52	38	28	57	89	268		CPU
Default	9	9	6	13	23	60	23	Time
Foil	17	5	7	9	5	43	16	(5m)
mFoil	22	12	9	6	10	59	22	(2h)
Golem	17	9	5	11	10	52	20	(1h)
Indigo (1996)	21	14	9	18	33	95	36	(9h)
Claudien	31	9	5	19	15	79	30	(16m)
CILGG(1996)	19	16	6	10	9	60	22	(85s)
CILGG DL	16	8	5	10	12	51	19	8s
CILGG HL	22	14	8	13	5	62	23	11s
CILGG CL	19	16	7	14	23	79	30	22s
CILGG 20-NN DL	20	12	9	16	38	95	36	19s
CILGG 20-NN HL	17	14	9	18	41	99	38	23s
CILGG 20-NN CL	26	12	10	18	37	103	39	52s

Table 2. Comparison of ILP-Approaches learning the MESH-Data set

7 Summary and Outlook

We have characterizes the boarder line of polynomial learnability of CARIN-$\mathcal{ALN}$ rules from ground facts by reducing it to the well investigated boarder-line of polynomial learnability in ILP. This work should be extended to more expressive forms of background knowledge, e.g. terminological axioms (an ontology) and CARIN-$\mathcal{ALN}$ rules. We also showed in a first experiment, that this theoretical transformation is applicable in practice. However, careful experimentation about the usefulness of using CARIN-$\mathcal{ALN}$ as hypothesis language is still missing. But, we provided a data preprocessing method, that allows us to do that with a broad range of ILP-systems on a broad range of ILP-applications.

Acknowledgements: This work has been partially founded by the Swiss Government (BBW Nr.99.0158) as part of the European commission Research Project Mining Mart (IST-1999-11993). I thank François Goasdoué, Céline Rouveirol and Véronique Ventos for very helpful and interesting discussions and especially for supporting me with a preprint of their inspiring CARIN-$\mathcal{ALN}$ completion rules.

References

[Baader and Küsters, 1998] Baader, F. and R. Küsters: 1998, 'Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions'. In: O. Herzog and A. Günter (eds.): *Proc. of the 22nd Annual German Conference on AI, KI-98*. pp. 129–140, Springer–Verlag.

[Baader and Sattler, 2000] Baader, F. and U. Sattler: 2000, 'Tableau Algorithms for Description Logics'. In: R. Dyckhoff (ed.): *Proc. of the Int. Conf. on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*. pp. 1–18, Springer-Verlag.

[Borgida, 1996] Borgida, A.: 1996, 'On the relative expressiveness of description logics and predicate logics'. *Artificial Intelligence* **82**, 353 – 367.

[Borgida et al., 1989] Borgida, A., R. J. Brachman, D. L. McGuinness, and L. A. Resnick: 1989, 'Classic: A structural data model for objects'. In: *Proc. ACM SIGMOD-89*. Portland, Oregon, pp. 58 – 67.

[Brachman et al., 1985] Brachman, R. J., V. P. Gilbert, and H. J. Levesque: 1985, 'An Essential Hybrid Reasoning System'. In: *IJCAI-85*. pp. 532 – 539.

[Brachman and Schmolze, 1985] Brachman, R. J. and J. G. Schmolze: 1985, 'An Overview of the KL-ONE Knowledge Representation System'. *Cognitive Science* **9**(2), 171 – 216.

[Cohen and Page, 1995] Cohen, W. and C. Page: 1995, 'Polynomial Learnability and Inductive Logic Programming: Methods and Results'. *New Generation Computing, Special issue on Inductive Logic Programming* **13**(3-4), 369–410.

[Cohen et al., 1992] Cohen, W. W., A. Borgida, and H. Hirsh: 1992, 'Computing Least Common Subsumers in Description Logic'. In: *Proc. of the 10th National Conference on Artificial Intelligence*. San Jose, California, MIT–Press.

[Cohen and Hirsh, 1994a] Cohen, W. W. and H. Hirsh: 1994a, 'The Learnability of Description Logics with Equality Constraints'. *Machine Learning* **17**, 169–199.

[Cohen and Hirsh, 1994b] Cohen, W. W. and H. Hirsh: 1994b, 'Learning the CLASSIC Description Logic: Theoretical and Experimental Results'. In: *Proc. of the Int. Conf. on Knowledge Representation (KR94)*.

[Donini et al., 1991] Donini, F., M. Lenzerini, C. Nardi, and W. Nutt: 1991, 'Tractable Concept Languages'. In: *Proc. IJCAI-91*. pp. 458–463.

[Džeroski and Dolsak, 1992] Džeroski, S. and B. Dolsak: 1992, 'A Comparision of Relation Learning Algorithms on the Problem of Finite Element Mesh Design'. In: *Proc. of the ISEEK Workshop*. Ljubljana, Slovenia.

[Frazier and Pitt, 1994] Frazier, M. and L. Pitt: 1994, 'Classic Learning'. In: *Proc. of the 7th Annual ACM Conference on Computational Learning Theory*. pp. 23–34.

[Goasdoué et al., 2001] Goasdoué, F., C. Rouveirol, and V. Ventos: 2001, 'Optimized Coverage Test for Learning in CARIN-$\mathcal{ALN}$'. Technical report, L.R.I, C.N.R.S and Université Paris Sud. Work in progress.

[Helft, 1989] Helft, N.: 1989, 'Induction as nonmonotonic inference'. In: *Proceedings of the 1st International Conference on Knowledge Representation and Reasoning*.

[Horrocks, 1998] Horrocks, I.: 1998, 'The FaCT System'. In: H. de Swart (ed.): *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98*. pp. 307–312, Springer-Verlag, Berlin.

[Kietz, 1996] Kietz, J.-U.: 1996, 'Induktive Analyse Relationaler Daten'. Ph.D. thesis, Technical University Berlin. (in german).

[Kietz and Džeroski, 1994] Kietz, J.-U. and S. Džeroski: 1994, 'Inductive Logic Programming and Learnability'. *SIGART Bulletin* **5**(1).

[Kietz and Lübbe, 1994] Kietz, J.-U. and M. Lübbe: 1994, 'An Efficient Subsumption Algorithm for Inductive Logic Programming'. In: *Proc. of the Eleventh International Conference on Machine Learning (ML94).*

[Kietz and Morik, 1994] Kietz, J.-U. and K. Morik: 1994, 'A polynomial approach to the constructive Induction of Structural Knowledge'. *Machine Learning* **14**(2), 193–217.

[Levy and Rouset, 1998] Levy, A. Y. and M.-C. Rouset: 1998, 'Combining horn rules and description logic in CARIN'. *Artificial Intelligence* **104**, 165–209.

[Moser, 1983] Moser, M. G.: 1983, 'An Overview of NIKL, the New Implementation of KL-One'. In: Bolt (ed.): *Research in Knowledge Representation and Natural Language Understanding.* Cambridge, Mass., pp. 7 – 26, Beranek and Newman Inc. BBN Technical Report 5421.

[Muggleton and Feng, 1992] Muggleton, S. H. and C. Feng: 1992, 'Efficient induction of logic programs'. In: S. H. Muggleton (ed.): *Inductive Logic Programming.* Academic Press.

[Nebel, 1990] Nebel, B.: 1990, 'Terminological reasoning is inherently intractable'. *Artificial Intelligence* **43**, 235 – 249.

[Peltason et al., 1989] Peltason, C., A. Schmiedel, C. Kindermann, and J. Quantz: 1989, 'The BACK System revisited'. KIT-REPORT 75, Techn. Univ. Berlin, Berlin, W. Germany.

[Quantz et al., 1996] Quantz, J., G. Dunker, F. Bergmann, and I. Keller: 1996, 'Th flex system'. Technical report, KIT-Report, Technical University, Berlin, Germany.

[Rouveirol and Ventos, 2000] Rouveirol, C. and V. Ventos: 2000, 'Towards learning in CARIN-$\mathcal{ALN}$'. In: J. Cussens and A. M. Frisch (eds.): *Proc. Tenth International Conference on Inductive Logic Programming, ILP'00.* Berlin, Springer Verlag.

[Shapiro, 1983] Shapiro, E. Y.: 1983, *Algorithmic Program Debugging*, ACM Distinguished Doctoral Dissertations. Cambridge, Mass.: The MIT Press.

[Vilain, 1985] Vilain, M.: 1985, 'The Restricted Language Architecture of a Hybrid Reasoning System'. In: *IJCAI-85.* pp. 547 – 551.

[von Luck et al., 1987] von Luck, K., B. Nebel, C. Peltason, and A. Schmiedel: 1987, 'The Anatomy of the BACK System'. KIT-Report 41, Techn. Univ. Berlin, Berlin, West Germany.

A. Appendix

A.1 Schema for background knowledge to learn CARIN-$\mathcal{ALN}$

This little piece of code should enable any ILP-system (with intervals, but see footnote 2) to learn CARIN-$\mathcal{ALN}$ rules from ground facts under local CWA (But the CWA rule 6. is not applied to the HL-facts, only it's consequences for DL-Terms is coded), when it is added as background knowledge directly, or in form of the ground facts it produces. Make sure that your program makes only mode compatible calls to it, and that you limit the depth of the relation chain of your starting clauses or facts. The extension that everything subsumes $\bot(X)$ is already coded into it, as everything (every predicate) succeeds, if called with [], i.e. no instances. The first modeb declaration and the first line of code for rr_<R> 'connects' the DL-term with the HL-Terms.

Mode declarations are in Progol notation. <Name> indicates that we have to apply string concatenation to build the predicate or type name. Except for count_solutions - unifying the second argument with the number of solutions of the call in the first arg - and minimum (maximum) - get the minimum (maximum) out of a list of numbers - everything is normal Prolog. Dom and Range are (sorted) lists of terms, they should be treated like atoms, i.e. only their equality is important.

```
For every primitive concept <C> with a modeb(_,C(+<In>) add:
:- modeb(1,cn_<C>(+set_<In>)).
:- modeb(1,cp_<C>(+set_<In>)).
cn_<C>(Dom):- nonvar(Dom), forall(member(D,Dom),\+ <C>(D)),!.
cp_<C>(Dom):- nonvar(Dom), forall(member(D,Dom),<C>(D)), !.

For every primitive role <R> with a modeb(_,<R>(+<In>,-<Out>) add:
:- modeb(1,rr_<R>(+<In>,#,-set_<Out>)).
:- modeb(1,rr_<R>(+set_<In>,#,-set_<Out>)).
rr_<R>(Dom,Card,Range):-
    (atom(Dom),\+ Dom = [] -> DomL = [Dom] | Dom = DomL),
    findall(R,(member(D,DomL),<R>(D,R)),RangeD),
    sort(RangeD,Range),
    findall(C,(member(D,DomL),
               count_solutions(<R>(D,_),C)
              ),Cards),
    min(Cards,Atleast), max(Cards,Atmost),
    Card = [Atleast .. Atmost], !.
```

A.2 A Mesh-example as produced by the method

Examples as the following are used in the experiment in section 5. The first 4 line are the HL-literals up to depth 1, the rest are the deterministic DL-literals up to depth 3. For Cilgg HL only the first part is used, for Cilgg DL only the second part, and for Cilgg CL both parts. Note that this representation uses cycles, but these cycles could be broken up into a tree, up to the specified depth limit.

```
pos:mesh(a2,1):- not_important(a2), fixed(a2), not_loaded(a2),
   neighbour(a2,a1), long(a1), fixed(a1), not_loaded(a1),
   neighbour(a2,a54), long(a54), fixed(a54), one_side_loaded(a54),
   neighbour(a2,a3), usual(a3), fixed(a3), not_loaded(a3),

   rr_neighbour(a2, [3..3], [a1,a3,a54]),
      cp_fixed([a1,a3,a54]),  cp_normal([a1,a3,a54]),
      rr_neighbour([a1,a3,a54], [3..4],
                   [a1,a2,a24,a34,a4,a41,a44,a54]),
         rr_neighbour([a1,a2,a24,a34,a4,a41,a44,a54],
                      [3..4],
                      [a1,a2,a23,a24,a3,a34,a35,a36,a39,a4,
                       a40,a41,a42,a44,a5,a54]),
         rr_opposite([a1,a2,a24,a34,a4,a41,a44,a54],
                     [0..6],
                     [a13,a15,a17,a19,a22,a23,a34,a54]),
      rr_opposite([a1,a3,a54], [1..6],[a11,a13,a15,a17,a19,a22,a23,a34,a9]),
         rr_neighbour([a11,a13,a15,a17,a19,a22,a23,a34,a9], [3..5],
                      [a10,a12,a14,a16,a18,a20,a21,a22,a23,a24,a3,a35,a38,
                       a4,a40,a42,a43,a47,a48,a49,a50,a51,a52,a53,a55,a8]),
         rr_opposite([a11,a13,a15,a17,a19,a22,a23,a34,a9], [1..2],
                     [a1,a3,a32,a33,a54]),
            c_fixed([a1,a3,a32,a33,a54]),
            % The next one (depth 4) is present as is has no new output.
            rr_opposite([a1,a3,a32,a33,a54],[1..6],
                        [a11,a13,a15,a17,a19,a22,a23,a34,a9]),
   rr_opposite(a2, [0..0], []),
   % Everything is true on the Bottom concept []
   c_acircuit([]),       c_circuit([]),          c_circuit_hole([]),
   c_cont_loaded([]),    c_fixed([]),            c_free([]),
   c_half_circuit([]),   c_half_circuit_hole([]), c_hole([]),
   c_long([]),           c_long_for_hole([]),    c_normal([]),
   c_not_important([]),  c_not_loaded([]),       c_one_side_fixed([]),
   c_one_side_loaded([]), c_quarter_circuit([]),  c_quarter_circuit_hole([]),
   c_short([]),          c_short_for_hole([]),   c_two_side_fixed([]),
   c_two_side_loaded([]), c_usual([]),
   rr_neighbour([], [0..0], []),
   rr_opposite([], [0..0], []).
```