Accepted to appear in: S. Matwin, C. Sammut (ed.), Proc of the 12th Int. Conf. on Inductive Logic Programming, ILP-2002, Sydney, Australia.

On the Learnability of Description Logic Programms

Jörg-Uwe Kietz

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

Abstract. Carin-$\mathcal{ALN}$ as proposed recently by Rouveirol and Ventos [2000] is an interesting new rule learning bias for ILP. By allowing description logic terms as predicates of literals in datalog rules, it extends the normal bias used in ILP as it allows the use of all quantified variables in the body of a clause, instead of the normal exist quantified variables and it has atleast and atmost restrictions to access the amount of indeterminism of relations. From a complexity point of view Carin-$\mathcal{ALN}$ allows to handle the difficult indeterminate relations efficiently by abstracting them into determinate aggregations.
This paper describes a method which enables the embedding of Carin-$\mathcal{ALN}$ rule subsumption and learning into datalog rule subsumption and learning with numerical constraints [Sebag and Rouveirol, 1996]. 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.

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 an 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 with simple numeric constraints as formalized in [Sebag and Rouveirol, 1996] and as present in most ILP-systems, e.g. Foil [Quinlan and Cameron-Jones, 1993] or Progol [Muggleton, 1995]. A simple invertible function encodes 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 encoding, 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-$\mathcal{ALN}$as this encoding can be interpreted as a prediction preserving reduction as used in [Cohen, 1995] to obtain PAC-learnability results for ILP.

In section 2 we repeat the basic definitions of the description logic $\mathcal{ALN}$ and define the basis of our new encoding into horn logic. In section 3 we define the description logic program (DLP) formalism. We show how it is related to DLP (3.1) and ILP (3.2) and we define a reasoning procedure by extending the encoding from section 2 to DLP rules (3.3). In section 3.4, we state characterize the boarder line of polynomial learnability of DLP rules using the encoding into horn logic. Finally in section 4, we demonstrate with a first experiment, that this encoding can be used to learn DLP rules using a normal ILP-systems on an extended, i.e. preprocessed data-set.

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. 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.

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

Definition 1 ($\mathcal{ALN}$ -terms and their interpretation).
Let P denote a primitive concept, i.e. an unary predicate, and R denote a primitive role, i.e. a binary predicate, n is a positive integer and the C_i are concept terms. The set of all $\mathcal{ALN}$-concept-terms ($\mathcal{C}$) consist of everything in the left row of the table 1 (and nothing more) and their interpretation (I, Δ) (see def. 7 below) with respect to the interpretation of primitive concept and roles is defined in the rigth row of the table 1 (C^I is used as a shortcut for $I(C)$, if C is not primitive). The set of all $\mathcal{ALN}$-role-terms ($\mathcal{R}$) is just the set of all primitive roles R.

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	$\{x \in \Delta^I \mid \ \lVert\{y \in \Delta^I \mid< x, y >\in R^I\}\rVert \geq n\}$
$\leq nR$	**atmost** n R	$\{x \in \Delta^I \mid \ \lVert\{y \in \Delta^I \mid< x, y >\in R^I\}\rVert \leq n\}$

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

These language constructs can be used to build complex terms, e.g. the term *train* $\sqcap$ $\forall$ *has_car*.(*car* $\sqcap \leq 0$ *has_load*) can be used to define empty trains (all cars do not have a load), in Michalski's well-known train domain. An statement **not** expressible as a logic program, if we are restricted to the predicates already present in the train domain and cannot use externally computed predicates.

The main reasoning tasks with description logic terms are subsumption, equivalence and satisfiability checking for deduction and the least common subsumer (lcs) for learning.

Definition 2 (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, 1990b] 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 3 (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 $(P_1) < (\neg P_1) < ... < (P_n) < (\neg P_n) < (\geq n_1 R_1) < (\leq m_1 R_1) < (\forall R_1.C_1) < ... < (\geq n_n R_n) < (\leq m_n R_n) < (\forall R_n.C_n)$. 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 $C_i = \top$
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. $(\forall R.C_1 \sqcap \forall R.C_2) \mapsto \forall R.(merge^2(C_1 \sqcap C_2))$

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. $\square$

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 $norm(C) \neq \bot$).

[2] Merging two sorted lists into one sorted list as in merge-sort. In an optimized implementation the recursive application of the normalisation rules should be integrated into that linear (in the size of the conjunctions) process.

Definition 4 ($\mathcal{ALN}$encoding into constraint horn logic).
$\Phi(C) = h(X) \leftarrow \Phi(norm(C), X).$
$\Phi(\bot, X) = \bot(X)$
$\Phi(P\ \{\sqcap\ C\}, X) = cp_P(X)\{, \Phi(C, X)\}$
$\Phi(\neg P\ \{\sqcap\ C\}, X) = cn_P(X)\{, \Phi(C, X)\}$
$\Phi(\geq nR \sqcap \leq mR \sqcap \forall R.C_R\ \{\sqcap\ C\}, X) = rr_R(X, [n..m], Y), \Phi(C_R, Y)\{, \Phi(C, X)\}$
$\Phi(\leq mR \sqcap \forall R.C_R\ \{\sqcap\ C\}, X) = rr_R(X, [0..m], Y), \Phi(C_R, Y)\{, \Phi(C, X)\}$
$\Phi(\geq nR \sqcap \forall R.C_R\ \{\sqcap\ C\}, X) = rr_R(X, [n..*], Y), \Phi(C_R, Y)\{, \Phi(C, X)\}$
$\Phi(\geq nR \sqcap \leq mR\ \{\sqcap\ C\}, X) = rr_R(X, [n..m], Y)\{, \bot(Y)\ \textit{if}\ m = 0\}\{, \Phi(C, X)\}$
$\Phi(\forall R.C_R\ \{\sqcap\ C\}, X) = rr_R(X, [0..*], Y), \Phi(C_R, Y)\{, \Phi(C, X)\}$
$\Phi(\leq mR\ \{\sqcap\ C\}, X) = rr_R(X, [0..m], Y)\{, \bot(Y)\ \textit{if}\ m = 0\}\{, \Phi(C, X)\}$
$\Phi(\geq nR\ \{\sqcap\ C\}, X) = rr_R(X, [n..*], Y)\{, \Phi(C, X)\}$
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. For any normalised concept term only the first matching one has to by applied.

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 our DLP language, 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 $[minimum(Min_1, Min_2)..maximum(Max_1, Max_2)]$[3].
- 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-

[3] Formally this corresponds to reasoning and learning with simple numeric constraints as present in Constraint Logic Programs (CLP). In in most ILP-systems, e.g. Foil [Quinlan and Cameron-Jones, 1993] or Progol [Muggleton, 1995] this is done with the help of the computed built-in predicates $\leq$ and/or $\geq$. For Foil or Progol a more suitable encoding is $rr_R(X, n_{atleast}, m_{atmost}, Y)$, as they can learn c_{min} and c_{max} in literals like $c_{min} \leq n_{atleast}$ and $m_{atmost} \leq c_{max}$. [Sebag and Rouveirol, 1996] have analysed this CLP extension of ILP systems and it is easy to see that the properties of θ-subsumption also hold for θ_I-subsumption.

chains starting with X; and the $lgg_\perp$ of $\perp(X)$ and $\Phi(C,X)$ is $\Phi(C,X)$. This does not really need an extension of the logic, it can be simulated with normal θ-susumption and the following subclause for bottom: for any primitive role R: $rr_R(bottom, [*..0], bottom)$ and for any primitive concept C: $cp_C(bottom)$, $cn_c(bottom)$. Every $\Phi(C,X)$ θ-subsumes this cyclic[4] structure with a θ that maps every variable in it to the term *bottom* and the lgg of every $\Phi(C,X)$ and that subclause is $\Phi(C,X)$, i.e.
$lgg_I(rr_R(bottom, [*..0], bottom),\ rr_R(X, [Min..Max], Y)) = rr_R(X, [Min..Max], Y)$
and everything present in $\Phi(C,X)$ has a selection [Plotkin, 1970] within the lgg.

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 $\theta_{I\perp}$-subsumes the translation of D ($\Phi(C) \vdash_{\theta_{I\perp}} \Phi(D)$), and $lcs(C,D) \equiv \Phi^{-1}(lgg_{I\perp}(\Phi(C), \Phi(D)))$*

This theorem directly follows from the similarity between the translation Φ of the normalized concept terms and $\theta_{I\perp}$-subsumption and the correctness of 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.
- 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 Induction of Description Logic Programs

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

[4] This gives a possibility to represent cyclic concept definitions from a terminological component in a finite way. However we haven't checked the adequate semantic interpretation (descriptive, least or largest fixpoint [Nebel, 1990a]) of these cyclic terms so far, so we do not consider this possibility in this paper, even if it would be helpful with respect to ABox abstraction by most specific concepts [Baader and Küsters, 1998]

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

literals in horn rules. Concept terms represent unary predicates and role-terms 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. We bracket DL-terms with [], wherever we think that this will increase readability.

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

The following is a formal definition of syntax and semantic of Description Logic Programms (DLP) based on the usual first-order logic definitions adapted from [Görz, 1993, chapter 2], i.e. it is restricted to function-free clauses and extended to description logic by allowing $\mathcal{ALN}$-concept-terms as one-place predicates and $\mathcal{ALN}$-role-terms as two-place-predicates.

Definition 5 (variable, constant, term, atom, literal).
Let $\mathcal{V}$ be a countable set of variable-symbols, e.g. $X, Y, Z, U_1, U_2, U_3 \in \mathcal{V}$. Let $\mathcal{F}$ be a countable set of constant-symbols, e.g. $s, t, u, v, t_1, t_2, t_3 \in \mathcal{F}$. $\mathcal{T}(\mathcal{F}, \mathcal{V}) = \mathcal{F} \cup \mathcal{V}$ or as a shortcut $\mathcal{T}$, is the set of term over $\mathcal{F}$ and $\mathcal{V}$.

$\mathcal{P} = (\mathcal{P}_n)_{n \in \mathbb{N}}$ be a signature for predicate symbols, i.e. for $n \in \mathbb{N}$ is $p \in \mathcal{P}_n$ a predicatssymbols of arity n, e.g. $p, q, r \in \mathcal{P}_n$. The extension to description logic programms extends $\mathcal{P}_1$ by the set of all concept-terms $\mathcal{C}$ and $\mathcal{P}_2$ by the set of all role-terms $\mathcal{R}$ (see Definition 1).

$AT(\mathcal{F}, \mathcal{P}, \mathcal{V})$, or as a shortcut AT, is the set of all atomar formulas or atoms *with $\mathcal{P}_0 \subseteq AT(\mathcal{F}, \mathcal{P}, \mathcal{V})$ and $p(t_1, \ldots, t_n) \in AT(\mathcal{F}, \mathcal{P}, \mathcal{V})$, if $p \in \mathcal{P}_n$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, for $n \in \mathbb{N}^+$.*

$LIT(\mathcal{F}, \mathcal{P}, \mathcal{V})$ or as a shortcut LIT, with $LIT(\mathcal{F}, \mathcal{P}, \mathcal{V}) = AT(\mathcal{F}, \mathcal{P}, \mathcal{V}) \cup \{\neg A \mid A \in AT(\mathcal{F}, \mathcal{P}, \mathcal{V})\}$, is the set of all literals *over $\mathcal{F}$, $\mathcal{P}$ and $\mathcal{V}$. The atoms in LIT are also called positiv literals, and the negated atoms are also called negativ literals.*

Definition 6 (clause, fact, rule, logic program).
A finite subset of $LIT(\mathcal{F}, \mathcal{P}, \mathcal{V})$ is called a clause over $\mathcal{F}$, $\mathcal{P}$ and $\mathcal{V}$. The empty clause $\{\}$ is also written as $\Box$. A clause consisting of exactly one positive literal is called fact (a fact $\{a\}$ is also writen as a.); a clause consisting of exactly one positive literal and one or more negative literals is called rule (a rule $\{a_0, \neg\ a_1, \ldots, \neg a_n\}$ is also writen $a_0 \leftarrow a_1, \ldots a_n$.); a finite set of facts and rules is called a description logic program (DLP).

Definition 7 (Interpretations and Models of clause sets and description logic programs).
Let Δ be a domain and I an interpretation function, mapping clauses to truth values $(0 and 1)$, n-ary predicates ($P_n \in \mathcal{P}$) to n-ary relations (written P_n^I) over the domain Δ, terms ($t \in \mathcal{F}$) to elements of Δ (written t^I), $\mathcal{ALN}$-concept-terms

($C \in \mathcal{C}$)as defined in def. 1 and $\mathcal{ALN}$-role-terms ($R \in \mathcal{R}$)as defined in def. 1. An interpretation (I, Δ) satisfies (is a model of) a clause set, iff every clause is satisfied by (I, Δ). A clause is satisfied by (I, Δ), iff the interpretation function assigns the truth value 1 to it. For all interpretations (I, Δ) the following must hold [6]*:*

1. $\forall t_1, t_2 \in F : t_1 \neq t_2$ *iff* $t_1^I \neq t_2^I$,[7]
2. $I(\Box) = 0$, *and* $\forall X \in \mathcal{T} : I(\bot(X)) = 0$
3. $I(\{l_1, \ldots, l_n\}) = 1$, *iff for all vectors* $\boldsymbol{a} = a_1, \ldots, a_m$ *with* $a_i \in \Delta, (1 \leq i \leq m) : I\langle X_1/a_1, \ldots, X_n/a_n\rangle(l_1) = 1$ *or* $\ldots$ *or* $I\langle X_1/a_1, \ldots, X_n/a_n\rangle(l_n) = 1$, *where* $\boldsymbol{X} = X_1, \ldots, X_m$ *is the vector of all variables in the clause* $\{l_1, \ldots, l_n\}$,
4. $I(\exists\{l_1, \ldots, l_n\}) = 1$, *iff there exists a vector* $\boldsymbol{a} = a_1, \ldots, a_m$ *with* $a_i \in \Delta, (1 \leq i \leq m) : I\langle X_1/a_1, \ldots, X_n/a_n\rangle(l_1) = 1$ *or* $\ldots$ *or* $I\langle X_1/a_1, \ldots, X_n/a_n\rangle(l_n) = 1$, *where* $\boldsymbol{X} = X_1, \ldots, X_m$ *is the vector of all variables in the clause* $\{l_1, \ldots, l_n\}$,[8]
5. $I(\neg A) = 1$ *iff* $I(A) = 0$
6. $I(P_n(t_1, \ldots, t_n)) = 1$ *iff* $t_1^I, \ldots, t_n^I \in P_n^I$
7. $I(C(t)) = 1$ *iff* $t^I \in C^I$, *with* C^I *defined in table 1*
8. $I(R(t_1, t_2)) = 1$ *iff* $t_1^I, t_2^I \in R^I$.

Definition 8 (logical consequence).
A clause C follows from (is a logical consequence of) a clause set P (written $P \models C$), iff all models of P are also models of C. A clause set P_1 follows from (is a logical consequence of) a clause set P_2 (written $P_2 \models P_1$), iff all models of P_2 are also models of P_1.

We also use the normal ILP definition of learning to define learning of description logic programs (called Induction of Description Logic Programms or short IDLP).

Definition 9 (The IDLP Learning Problem).
Given a logical language $\mathcal{L}$ (i.e. our langauage DLP) with a consequence relation $\models$, background knowledge B in a Language $LB \subseteq \mathcal{L}$, positive and negative examples $E = E^+ \cup E^-$ in a language $LE \subseteq \mathcal{L}$ consistent with B ($B, E \not\models \Box$) and not already a consequence of B ($\forall e \in E : B \not\models e$), and a hypothesis language $LH \subseteq \mathcal{L}$. Find a hypothesis $h \in LH$ such that:

(I) ($B, h, E \not\models \Box$), i.e. h is consistent with B and E.

[6] $I\langle X/a\rangle(l)$ means that during the interpretation $I(l)$ the variable X in the literal l is always interpreted as the domain element a. It must not be confused with a substitution as a domain element a may not have a constant in $\mathcal{F}$ denoting it, i.e. $I^{-1}(a)$ may be undefined.

[7] In normal first-order logic such a unique name assumption for constants is usually not needed (makes no difference, except that skolemization, e.g. used for existence-quantor elimination would elimentate models), but for counting in DLs with number-restrictions it is very useful. It must not be confused with the object identity restriction on variables (substitutions must be injective) also used in some ILP approaches.

[8] This formula is not an expression **in** our language, but we need to interpret it in a lemma **about** our language.

(II) $(B, h \models E^+)$*, i.e. h and B explain* E^+*.*
(III) $(B, h \not\models E^-)$*, i.e. h and B do not explain* E^-*.*

The tuple $(\models, LB, LE, LH)$ *is called the* IDLP learning problem*. Deciding whether there exists such an* $h \in LH$*, is called the* IDLP 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 an* IDLP learning-algorithm*.*

As we are interested in polynomial learnability results [Cohen, 1995] for DLP, we are especially interested in IDLP problems, where B and E consist of ground (variable-free) facts, and H ist restricted to 1-clause Programms (see section 3.4).

3.1 The relation between (learning) CARIN-$\mathcal{ALN}$and (I)DLP

At first this looks quite different from the definition of CARIN-$\mathcal{ALN}$ in [Levy and Rouset, 1998] or the CARIN-$\mathcal{ALN}$learning problem as defined in [Rouveirol and Ventos, 2000], but it is in fact quite similar. We of course do not claim equivalence of the formalisms, but we think that we have transported the important (to us) ideas from CARIN-$\mathcal{ALN}$ into a more (inductive) logic programming oriented form. The DLP formalism defined above is in general two expressive compared with CARIN-$\mathcal{ALN}$, as so far we have neither forbidden recursion nor DL-literals as the head of rules, two things known to be quite difficult to reason with in CARIN-$\mathcal{ALN}$ [Levy and Rouset, 1998]. But we have to introduce such restrictions a well, as rules like $\mathit{connected}(X, Y) \leftarrow \mathit{node}(X),\ \mathit{node}(Y), [\leq 0\ \mathit{connected}](X)$ expressible in the formalism are difficult to interpret[9] under the normal model-theoretic semantics we used. But as in ILP we want to start with the general idea, and than introduce the restrictions needed (as e.g. already implied from the fact that datalog reasoning and learning is a subproblem of the DLP reasoning and learning) to come to a characterisation of polynomial learnability in the end. Given the goal of learning DLPs, recursive description logic programs are not very interesting anyway, as the set of know polynomial learnable recursive logic programs is very restricted [Cohen, 1993] and so far only of interest for synthesis of toy programms and not for real data analysis or mining applications.

The other main difference between DLP and CARIN-$\mathcal{ALN}$ is that we have unified assertional component (the set of facts from our DLP) and rule component (the set of rules from our DLP) of CARIN-$\mathcal{ALN}$, and - as already discussed in section 2 - we have dropped the terminological component of CARIN-$\mathcal{ALN}$, but are instead able to express rules and assertions, where concepts are already expanded with respect to the (acyclic) terminological axioms [Nebel, 1990a], e.g. instead of separating terminological axioms T, rules R and assertions A as in the following example from [Rouveirol and Ventos, 2000]:

[9] They do not have a (stable) model, so they have to be interpretated as contradictory, but an interpretation as default rule may be more adequate. But this is not a topic of this paper.

$T = \{empty\_car \equiv car \sqcap\ \leq 0\ has\_load,\ empty\_train \equiv train \sqcap \forall has\_car.empty\_car\}$
$R = \{east\_train(X) \leftarrow empty\_train(X)\}$
$A = \{empty\_car(d),\ train(a),\ has\_car(a, d),\ \leq 1 has\_car(a)\}$
we require the representation of the equivalent (with respect to assertional and rule consequences) terminological expanded rule and facts within one description logic program:
$east\_train(X) \leftarrow [train\ \sqcap\ \forall has\_car.[car\ \sqcap\ \ \leq 0\ has\_load]](X).$
$train(a).\ has\_car(a, d).\ [\leq 1 has\_car](a).\ [car\ \sqcap\ \ \leq 0\ has\_load](d).$

Except for the dropped $\mathcal{ALN}$ terminological component which makes polynomial learnabilty results possible (with a terminological component learning is atleast as hard as reasoning, i.e coNP-hard [Nebel, 1990b]) this is only a syntactic difference.

Concerning the learning framework in [Rouveirol and Ventos, 2000] and the IDLP problem, the main difference is that they use the learning from interpretation setting, whereas we use the normal ILP setting. For non-recursive clauses, this makes no difference, and they could be easily transformed into each other with only a polynomial size difference, if B and E consist of ground facts, i.e. take the whole B as interpretation for each example, or take the union of all interpretations as B.

3.2 The relation between (I)LP and (I)DLP

As known from CARIN-$\mathcal{ALN}$, there are also serious differences between DLP and LP. A set of rules may have more consequences than the union of the consequnces of the individual rules, e.g. let
$P_1 = \{train(a).\ east\_train(X) \leftarrow [train\ \sqcap\ \ \leq 1\ has\_car](X).\}$ and
$P_2 = \{train(a).\ east\_train(X) \leftarrow [train\ \sqcap\ \ \geq 1\ has\_car](X).\}$.
Neither $P_1 \models east\_train(a)$ nor $P_2 \models east\_train(a)$ but $P_1 \cup P_2 \models east\_train(a)$ as $[\geq 1\ has\_car](X) \vee [\leq 1\ has\_car](X)$ is a tautology.

Theoretically (section b 3.4), this does not hinder our transfer of learnability results from LP to DLP very much as most results concerning polynomial learnability of ILP are restricted to one rule programs [Cohen, 1995], but this is a serious problem for applying these positive DLP results practically as done in ILP. Most ILP systems use the greedy strategy of learning rules individually such that no negative examples are covered by any rule until all positive examples are covered by a rule. This difference between DLP and LP means that a rule set may cover negative examples, even if no rule does. However, this is only a problem in learning DLPs from DLP facts, but not in the practically much more important subproblem of learning DLPs from ILP data sets, e.g. from relational databases. Under the usual open world assumption in DLP no rule ever learned from an ILP data set will contain an atmost or all restriction, as under the open-world assumption (OWA) a rule with an atmost or all restiction will never cover an example described by datalog facts only, e.g. P_1 will never cover a train as without an explicit atmost restriction in the examples, i.e. the OWA means we will never know that we know all cars of a train. This however makes learning DLP quite useless as well, as only atleast restrictions are learnable from ILP facts, but neither atmost nor all restriction.

There is an easy practical way out of this problem used in section 4 to learn DLP programs from ILP data sets namely assuming that the background knowledge contains all relevant background knowledge for the examples (a closed world assumption on B). In that case we can also learn atmost and all restriction and it is always possible to decide which side of a tautology is applicable, e.g. if we know that we know all the cars of the trains under consideration, we can decide by counting, if rule P_1 or P_2 or both are applicable. Also for a primitive concept P we are able to decide, if P or $\neg P$ is true. In summary, we do not have to worry about this reasoning problem with respect to the goals of this paper. Theoretically we only need reasoning with one clause programs, practically we need a closed world assumption to learn DLPs and in both cases this problem disappears.

Another difference between LP and DLP reasoning is that a DLP like the one above also contains implicit knowledge not only due to rule reasoning as usual in logic programing but also due to interactions between facts with description logic literals (DL-literals) and normal literals (HL-literals[10]), i.e. the fact $\forall has\_car.[car \sqcap \; \leq 0 \; has\_load]](a)$ is true in every model of the DLP as a consequence of the interaction between all the above facts and after we have deduced that $east\_train(a)$ can be deduced as a consequence of the facts and the rule.

3.3 Subsumption between DLP clauses

We have to make such implicit literals explicit, as a reasoning procedure based on substructure matching like 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 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 to a set of DLP facts, i.e. either background knowledge B or the body B of a DLP rule $H \leftarrow B$.

Definition 10 (Disjointness Clique). *A set of terms $\{X_1, ..., X_n\}$ is called a disjointness clique of size n of a rule body B, iff for all pairs $\{X_i, X_j\} \subseteq \{X_1, \ldots, X_n\}$, with $i \neq j$ either $X_i\sigma$ and $X_j\sigma$ are not unifiable terms, i.e. different constants, or $\{C_i(X_i), C_j(X_j)\}\sigma \subseteq B$ and $C_i \sqcap C_j \equiv \perp$, i.e. there must be no model, where they are interpreted as the same individual.*

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

Definition 11 (Completion for DLP under θ-Subsumption). *Let B a set of DLP facts.*

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 $B \rightharpoonup B \cup (\{r(X_0, U)\})\sigma$, and U is a new variable not occuring in B (and σ) so far.*
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 $B \rightharpoonup B \cup \{C(X_1)\sigma\}$.*
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 there is no further $r(X_0, X_{n+1})\sigma$ in B, and*
   - *k is size of the maximal disjointness clique of $\{X_1, \ldots, X_n\}\sigma$ in B, and*
   - *$((\geq m\ r)(X_0))\sigma \notin B$ for any $m \geq k$*

   then $B \rightharpoonup B \cup \{[\geq k\ r](X_0)\}\sigma$.
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 $\{X_1, ..., X_n\}\sigma$ is a disjointness clique is of size n, let $C = lcs(C_1, \ldots, C_n)$ and $((\forall r.C)(X_0))\sigma \notin B$ then $B \rightharpoonup B \cup \{[\forall r.C](X_0)\}\sigma$,*
5. **collect and normalize concept-terms over the same variable:** *if $C(X_0)\sigma \in B$ and $D(X_0)\sigma \in B$ with C and D are DL-terms then $B \rightharpoonup B \setminus \{C(X_0)\sigma, D(X_0)\sigma\} \cup \{[norm(C \sqcap D)](X_0)\}\sigma$*

This corresponds to ABox reasoning presented as equivalent to tableux-based terminological component reasoning in [Baader and Sattler, 2000], but for learning a tableux reasoning approach which does only consistency checking is not sufficient. We need a constructive normalisation approach to have the consequences available as input for learning. This of course is only feasible for a simple DL like $\mathcal{ALN}$. It in fact corresponds to infering for each term t occuring in B a depth-bounded acyclic approximation of it's (cyclic) most specific concept as done for $\mathcal{ALN}$-ABoxes in [Baader and Küsters, 1998].

Lemma 2 (The completion rules are correct). *For any B, if $B \rightharpoonup B'$, then $\exists B \models \exists B'$.*

Proof. For $\mathcal{ALN}$ it is easy to verify, that whenever an interpretation is a model of a B satisfying the premisses of a completion rule it is a model of the conclusion of this completion rule. For a complete proof see appendix A.2.

Lemma 3 (The completion rules are complete). *For all atoms $a \in AT$, whenever B_0 is consistent ($\exists B_0 \not\models \exists\bot(X)$), if $\exists B_0 \models \exists a$, there exit a chain of $B_0 \rightharpoonup B_1 \rightharpoonup \ldots \rightharpoonup B_n$ such that for some substitution σ either $a\sigma \in B_n\sigma$ or $a\sigma = C_1(t)$ and there exist a $C_2(t)\sigma \in B_n\sigma$ and $C_2 \sqsubseteq C_1$.*

Proof. To show completeness, we have to argue that every atom deducible from a consistent B can be generated by the rules and whenever it is not possible to generate an atom with a rule we have to construct a model of B where the atom is not true. For a complete proof see appendix A.3.

Theorem 2 (Subsumption between completed encoded rules is sound). *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 DLP rule C implies a non-tautological DLP 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. Correctness and completness follow from Theorem 1 (Completness of Φ and θ-subsumption to test $C_2 \sqsubseteq C_1$), the correctness and completness of θ-subsumption for non-recursive non-tautological LP rules and the correctness (lemma 2)and completness (lemma 3) of the completion rules to infer for every term its most specifc concept (up to depth i), the fact that for a rule ($h \leftarrow B$) all variables only in the body B can be seen as existential quantified (as required in the above lemmata), the fact that a rule is tautological, if it's body B is contradictory (another requirement in the above lemmata). Further note that in a $C_2 \sqsubseteq C_1$ test, no subterm of C_2 deeper than the deepest subterm in C_1 is needed (could not be reached). $\square$

Theorem 3 (Completion is polynomial for depth-bounded DL terms).

Proof. For a set of CARIN-$\mathcal{ALN}$ atoms like $\{(\geq 2r \sqcap \forall r.(\geq 2r) \sqcap \ldots \sqcap \overbrace{\forall r. \ldots \forall r}^{n}.(\geq 2r))(a)\}$, we don't generate the whole (exponential in n) binary tree with rule 1 and 2, but just one path from the root to a leave (linear in n), as all these paths of existential variables are indistinguishable under θ-subsumption and would colapse into one path under equivalence (θ-reduction [Plotkin, 1970]). 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. The max-clique search needed in the case of variables in rule 3 and 4 is more complex (NP-hard), but to learn from ground-facts 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 terminological component as in [Baader and Küsters, 1998] (see footnote 4 as well), or 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 may generate a complete tree, i.e. in the worst case it is exponential in the depth i, but for a fixed depth limit i it is polynomial as well. Rule 5 is needed to collect all terms together for normalization and is linear. $\square$

As Φ and φ are polynomial for ground clauses, the complexity of $C \models D$ is that of θ-subsumption, if C is ground. 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[11] with respect to D and each $LOC_i, 1 \leq i \leq n$, is a k–local[12]. 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 4 (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 $E' \models E$.*[13]

3.4 Learning Carin-$\mathcal{ALN}$

With the definition of learning and theorem 2 we can immediately conclude on the learnability of DLP programms consisting of a single rule.

Corollary 1. *A IDLP rule learning problems ($\models$,ground DLP $facts$, ground HL $facts$, $1-clause$ DLP programms) is polynomially learnable, if and only if the corresponding ILP learning problems ($\models$, $\Phi_r(\varphi$(ground DLP $facts$,i)), ground HL $facts$, $\Phi_r(\varphi(1-clause$ DLP programms,i))) is polynomial learnable.*

As the boarder line of polynomial learnability is quite well known for ILP due to a lot of positive and negative learnability results (see [Kietz, 1996; Kietz and Džeroski, 1994; Cohen and Page, 1995] for overviews), we are now able to characterize it for DLP rules as well. One of the most expressive (single horn clause) ILP-problem that is polynomial learnable, is learning a i, j-determinate-k-literal-local horn clause (see [Kietz and Lübbe, 1994] for definitions and discussions of these restrictions).

Theorem 5. *A 1-clause DLP program, where the HL part is i, j-determimate-k-literal-local and DL-terms are depth-bounded by i and are only allowed on head or determinate variables is polynomial learnable.*

Proof. The encoding of such a clause is a $2*i, j$-determimate-k-literal-local horn clause as the DL-term encodings are in itself determinate, start on a determinate variables and are bound in deth by i as well and as such they are polynomial learnable [Cohen, 1995]. □

4 Applying the results

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].

[11] The n-ary generalisation of what is called an attribute in DL, i.e. there is a determinate way to find a unique match of the output variables.

[12] 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)

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

4.1 Learning from databases using closed world assumption

As discussed in section 3.2, datalog facts or databases are not adequate to learn DLPs under OWA and the CWA is quite natural a way out for learning [Helft, 1989] as well as for databases. We do not want to close the world totally, but just locally, i.e we assume that if an object is described at all it is described completely, but we don't want to assume that we know about all objects of any world. The following two non-monotonic rules are doing just that.

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 there is no further $r(X_0, X_{n+1})\sigma$ in B, and
   - k is size of the maximal disjointness clique of $\{X_1, \ldots, X_n\}\sigma$ in B, and
   - there is no $[\leq m\ r](X_0)\sigma$ in B such that $m \leq k$

   then $B := B \cup \{[\leq m\ r](X_0)\}\sigma$.
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.

4.2 Learning Mesh-Design in DLP

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.

```
                                          mesh(A,2):- [usual ⊓
mesh(A,2):- [usual ⊓                                  ≥ 1 opposite ⊓
             ≥ 2 neighbour ⊓                          ≤ 1 opposite ⊓
             ≤ 3 neighbour ⊓                          ∀ opposite.[fixed ⊓
             ∀ neighbour.not_loaded                               ≥ 2 opposite ⊓
             ](A).                                                ≤ 2 opposite
                                                                  ]
                                                      ](A).
```

Fig. 1. Two of the description logic rules learned for Mesh-Design

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. However, this single experiment is not sufficient to claim the **usefulness** of CARIN-$\mathcal{ALN}$ as a hypothesis language in general. But it clearly shows, that the encoding defined in this paper is **applicable** in practical applications as well. It enables normal ILP-system to learn rules like the ones in Figure 1 efficiently, which are not in the normal ILP bias and which may be useful in some application domains.

	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

5 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 description logic programms. We also showed in a first experiment, that this theoretical encoding 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. On the theoretical side the IDLP framework could serve to analyse the learnability of the related new aggregaton-based ILP approaches [Krogel and Wrobel, 2001; Knobbe et al., 2001].

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, Ralf Molitor, Céline Rouveirol and Véronique Ventos for very helpful and interesting discussions about this topic and earlier drafts of this paper.

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.): *Proceedings of the 22nd Annual German Conference on Artificial Intelligence, 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.): *Proceedings of the International Conference 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.

[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, 1993] Cohen, W. W.: 1993, 'Pac-Learning a Restricted Class of Recursive Logic Programs'. In: *Proc. Tenth National Conference on Artificial Intelligence.* Cambridge, MA: MIT Press, pp. 86–92.

[Cohen, 1995] Cohen, W. W.: 1995, 'Pac-Learning non-recursive Prolog Clauses'. *Artificial Intelligence* **79**, 1–38.

[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.

[Görz, 1993] Görz, G. (ed.): 1993, *Einführung in die künstliche Intelligenz.* Addison Wesley.

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

[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.

[Knobbe et al., 2001] Knobbe, A. J., M. de Haas, and A. Siebes: 2001, 'Propositionalisation and Aggregates'. In: *Proceedings of the Fith European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'2001).* Berlin, New York, Springer Verlag.

[Krogel and Wrobel, 2001] Krogel, M. A. and S. Wrobel: 2001, 'Transformation-based Learning Using Mulirelational Aggregation'. In: *Proc. Elenth International Conference on Inductive Logic Programming, ILP'2001.* Berlin, New York, Springer Verlag.

[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.

[Muggleton, 1995] Muggleton, S. H.: 1995, 'Inverse Entailment and Progol'. *New Generation Computing* **13**.

[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, 1990a] Nebel, B.: 1990a, *Reasoning and Revision in Hybrid Representation Systems.* New York: Springer.

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

[Plotkin, 1970] Plotkin, G. D.: 1970, 'A note on inductive generalization'. In: B. Meltzer and D. Michie (eds.): *Machine Intelligence*, Vol. 5. American Elsevier, Chapt. 8, pp. 153 – 163.

[Quinlan and Cameron-Jones, 1993] Quinlan, R. and R. M. Cameron-Jones: 1993, 'FOIL: A Midterm Report'. In: P. Brazdil (ed.): *Proceedings of the Sixth European Conference on Machine Leaning (ECML-93).* Berlin, Heidelberg, pp. 3–20, Springer Verlag.

[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'2000.* Berlin, Springer Verlag.

[Sebag and Rouveirol, 1996] Sebag, M. and C. Rouveirol: 1996, 'Constraint Inductive Logic Programming'. In: L. de Raedt (ed.): *Advances in ILP.* IOS Press.

A. Appendix

This apendix is not considered part of the submitted paper, but additional material. The paper with the supplementing proofs integrated will be published as a technical report and on my web-side, if this paper is accepted.

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

This code should enable any ILP-system (with intervals, but see footnote 2) to learn DLP rules from ground facts under local CWA, i.e. it encodes the completion rules 3-7 under our encoding (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. The ILP-program must make only mode compatible calls to it, and that there is a limit on the depth of the relation chain of the 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>)).
%% use the largest positive integer instead of *
rr_<R>([],[* .. 0],[]):- !.
rr_<R>(Dom,Card,Range):-
    (atom(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 Correctness prove of the completion rules

lemma 2 (The completion rules are correct) For any B, if $B \rightharpoonup B'$, then $\exists B \models \exists B'$.

Proof. For $\mathcal{ALN}$ it is easy to verify, that whenever an interpretation is a model of a B satisfying the premisses of a completion rule it is a model of the conclusion of this completion rule.

If $((\geq n\ r)(X_0))\sigma$ is in B, then by table 1 and constraint 7 of definition 7 r^I of every model of B contains at least n tupples starting with $I(X_0\sigma)$. If $n \geq 1$ then $I(\exists r(X_0, U)\sigma) = 1$ in all these interpretations by Definition 7 constraint 4.

If $(\{(\forall r.C)(X_0), r(X_0, X_1)\})\sigma$ is in B, then by table 1 and constraint 7 of definition 7 $X_1^I \in C^I$ and therefore $I(C(X_1)) = 1$.

If $(\{r(X_0, X_1), \ldots, r(X_0, X_n)\})\sigma$ is in B and for each pair X_i, X_j of $\{X_1, ..., X_n\}\sigma$ $I(X_i) \neq I(X_j)$ can be shown by definition 7 contraint 1 then r^I contains atleast n tuples starting with $I(X_0)$, therefore $I([\geq n\ r](X_0)\}\sigma) = 1$

If $(\{r(X_0, X_1), C_1(X_1), \ldots, r(X_0, X_n), C_n(X_n)\})\sigma$ is in B and for each pair X_i, X_j of $\{X_1, ..., X_n\}\sigma$ $C_i \sqcap C_j \sqsubset \bot$ can be show then it follows by definition 7 constraint 2 that $I(X_i) \neq I(X_j)$ and therefore r^I contains atleast n tuples starting with $I(X_0)$, i.e. $I([\geq n\ r](X_0)\}\sigma) = 1$

If $(\{(\leq n\ r)(X_0), r(X_0, X_1), \ldots, r(X_0, X_n), C_1(X_1), \ldots, C_n(X_n)\})\sigma$ is in B and for each pair $I(X_i) \neq I(X_j)$ can be shown as above, then $X_1, \ldots X_n$ are all posible fillers of role r and therefore everything true on all of them (see definition lcs) is true as all restriction.

If $C(X_0)\sigma$ is in B then $X_0 \in C^I$ and if $D(X_0)\sigma$ is in B then $X_0 \in D^I$, therefore $X_0 \in C^I \cap D^I$ and $[I(C \sqcap D](X_0)) = 1$. □

A.3 Completness prove of the completion rules

lemma 3 (The completion rules are complete) For all atoms $a \in AT$, whenever B_0 is consistent $(\exists B_0 \not\models \exists\bot(X))$, if $\exists B_0 \models \exists a$, there exit a chain of $B_0 \rightharpoonup B_1 \rightharpoonup \ldots \rightharpoonup B_n$ such that for some substitution σ either $a\sigma \in B_n\sigma$ or $a\sigma = C_1(t)$ and there exist a $C_2(t)\sigma \in B_n\sigma$ and $C_2 \sqsubseteq C_1$.

Proof. To show completeness, we have to argue that every atom deducible from a consistent B can be generated by the rules and whenever it is not possible to generate an atom with a rule we have to construct a model of B where the atom is not true. From logic programming we know that atoms not related to the DL-part of B are not deducible if not already present except for variable renaming, but that is covered by the $\exists$ quantification in the lemma. If you look at table 1 it is clear that the membership of a term in a primitive concept, it's negation or an atmost restriction (see the OWA discussion above) can only follow from conjunctions or all restrictions. For conjunctions this is covered by the $C_2 \sqsubseteq C_1$ test in the lemma. For the all-restrictions this is covered by rule 2, which one two one encoded the semantic of all. Membership of a tuple in a primitive role cannot be deduced as well, only the existence of a role-filler can be required by an atleast restriction, this is encoded one to one by rule 1. In all these cases it is simple to construct an interpretation, which is not a model of the atom in question, as B is only able to specify the minimal content of an interpretation to be a model of B, i.e. by simply extending the interpretation by violating elements. As said B is able to specify the minimal content of an interpretation, and by that atleast restrictions become deducible. Whenever B contains n different constants as fillers of a role (constants are required to be interpreted as different domain elements by definition 7 restrriction 1) for an element an atleast n restriction restriction can be infered, that is done by rule 3. For variables this is in general not true and with pure horn-logic facts (without a build-in $\neq$) it is impossible to prevent unification of variables as it is impossible to enforce that an relation is interpreted as irreflexive (the reflexive tuples can always be added to the interpretation). That is only different for the DL-literal pairs $P(X), \neg P(Y)$ and $[\leq n_1 r](X), [\geq n_2 r](Y)$ (with $n_1 \leq n_2$) which enforce that there is no model, where X and Y can be interpreted as the same element. We have already argued that our rules are complete for primitive concepts, their negations, primitive roles and atmost restrictions, i.e. there exist a chain of B_i, such that they are present, if they are true (base case of complete induction). The open question is, is there a sequence of B_i such that all atleast restictrions needed to infer the next one are already present (inductive step). The answer is, if one is needed to prevent unification of the role-filler it is a model that the role-filer are unified and the atleast restriction does not hold. This completes the prove of completeness of rule 3 to infer all atleast restriction (without reasoning by case). The prove of completness of rule 4 to infer all all restrictions is similar. Just note, that rule 4 together with rule 5 is able to handle a set of facts B like the following correctly: $B = \{[\leq 2r](X),\ r(X, Y_1), p_1(Y_1), C_1(Y_1),\ r(X, Y_2), \neg p_1(Y_2), C_1(Y_2), r(X, Y_3), p_2(Y_3), C_2(Y_3), r(X, Y_4), \neg p_2(Y_4), C_2(Y_4)\}$. In this example $\{Y_1, Y_2\}$ and $\{Y_3, Y_4\}$ are disjointness-cliques of size 2. So rule 4 is applicable to both of them leading to $[\forall r.C_1](X)$ and $[\forall r.C_2](X)$ for which rule 5 (together with normalisation) produces $[\forall r.[C_1 \sqcap C_2]](X)$, which is exactly ($[\geq 2r](X)$ of course follows from rule 3) what follows from such a B, as in all possible models of B either $Y_1, Y3$ and $Y_2, Y4$ or $Y_1, Y4$ and $Y_2, Y3$ have to be interpreted as the same domain elements. $\square$