GENERALIZING REFINEMENT OPERATORS TO LEARN PRENEX CONJUNCTIVE NORMAL FORMS

SHAN-HWEI NIENHUYS-CHENG, WIM VAN LAER, JAN RAMON
AND LUC DE RAEDT

ERIM REPORT SERIES *RESEARCH IN MANAGEMENT*	
ERIM Report Series reference number	ERS-2000-39-LIS
Publication status / version	March 2000
Number of pages	19
Email address first author	Cheng@few.eur.nl
Address	Erasmus Research Institute of Management (ERIM) Rotterdam School of Management / Faculteit Bedrijfskunde Erasmus Universiteit Rotterdam PoBox 1738 3000 DR Rotterdam, The Netherlands Phone: # 31-(0) 10-408 1182 Fax: # 31-(0) 10-408 9640 Email: info@erim.eur.nl Internet: www.erim.eur.nl

ERASMUS **R**ESEARCH **I**NSTITUTE OF **M**ANAGEMENT

REPORT SERIES

RESEARCH IN MANAGEMENT

BIBLIOGRAPHIC DATA AND CLASSIFICATIONS		
Abstract	Inductive Logic Programming considers almost exclusively universally quantied theories. To add expressiveness, prenex conjunctive normal forms (PCNF) with existential variables should also be considered. ILP mostly uses learning with refinement operators. To extend refinement operators to PCNF, we should first do so with substitutions. However, applying a classic substitution to a PCNF with existential variables, one often obtains a generalization rather than a specialization. In this article we define substitutions that specialize a given PCNF and weakly complete downward refinement operator. Moreover, we analyze the complexities of this operator in different types of languages and search spaces. In this way we lay a foundation for learning systems on PCNF. Based on this operator, we have implemented a simple learning system PCL on some type of PCNF.	
Library of Congress Classification (LCC)	5001-6182	Business
	5201-5982	Business Science
	HB 143	Mathematical Programming
Journal of Economic Literature (JEL)	M	Business Administration and Business Economics
	M 11	Production Management
	R 4	Transportation Systems
	C 61	Optimization Techniques, Programming Models
European Business Schools Library Group (EBSLG)	85 A	Business General
	260 K	Logistics
	240 B	Information Systems Management
	250 B	Logical Theory
Gemeenschappelijke Onderwerpsontsluiting (GOO)		
Classification GOO	85.00	Bedrijfskunde, Organisatiekunde: algemeen
	85.34	Logistiek management
	85.20	Bestuurlijke informatie, informatieverzorging
	31.80	Toepassingen van de wiskunde
Keywords GOO	Bedrijfskunde / Bedrijfseconomie	
	Bedrijfsprocessen, logistiek, management informatiesystemen	
	Logisch programmeren	
Free keywords	PCNF, substitutions, refinement, learning, completeness	
Other information	Essential differences between this version and [NLRR99]: (1) We add proofs to almost all the results. (2) Section 6 about the complexities is new. (3) The section about Learning PCNF in Practice is extended with more explanations and some complexity analysis. (4) More motivations are given.	

Generalizing Refinement Operators to Learn Prenex Conjunctive Normal Forms

Shan-Hwei Nienhuys-Cheng, cheng@few.eur.nl
Wim Van Laer,Jan n Ramon, {wimv, janr}@cs.kuleuven.ac.be
Luc De Raedt, deraedt@informatik.uni-freiburg.de
Department of Computer Science, Katholieke Universiteit Leuven

March 1, 2000

Abstract. Inductive Logic Programming considers almost exclusively universally quantified theories. To add expressiveness, prenex conjunctive normal forms (PCNF) with existential variables should also be considered. ILP mostly uses learning with refinement operators. To extend refinement operators to PCNF, we should first do so with substitutions. However, applying a classic substitution to a PCNF with existential variables, one often obtains a generalization rather than a specialization. In this article we define substitutions that specialize a given PCNF and a *weakly complete* downward refinement operator. Moreover, we analyze the complexities of this operator in different types of languages and search spaces. In this way we lay a foundation for learning systems on PCNF. Based on this operator, we have implemented a simple learning system PCL on some type of PCNF.

Essential differences between this version and [NLRR99]: (1) We add proofs to almost all the results. (2) Section 6 about the complexities is new. (3) The section about *Learning PCNF in Practice* is extended with more explanations and some complexity analysis. (4) More motivations are given.

1 Introduction

Inductive Logic Programming learns a *correct* logic formula with respect to examples. The definition of correctness depends often on how the examples are presented. If the examples are ground atoms, a definite program which implies all the positive examples but none of the negative ones is expected. If a number of interpretations are given as positive examples, then a formula with these interpretations as models should be found. In both cases a *downward* refinement operator ρ can be used to search a correct formula in a learning system. For a given formula ψ, $\rho(\psi)$ contains a finite set of formulas implied by ψ. If a searching process begins with empty clause $\top = \Box$, it should be replaced by the set of its refinements in $\rho(\top)$ because $\top$ implies all the negative examples in the first case and it has no model in the second case. A refinement $\phi \in \rho(\top)$ may have to be replaced by its refinements again because ϕ is false or some given interpretations are not its model. This process can go on until we find a correct theory w.r.t. the examples.

Refinement operators for subsumption have often been used ([S81, RD97]) to learn a correct universally quantified theory incrementally. If clause C subsumes clause D, then a *refinement chain* exists from C to D using *elementary substitutions* and adding literals. Let $C = p(x,y)$ and $D = p(x,x) \vee \neg q(f(x))$. Then a chain may be $p(x,y)$, $p(x,y) \vee \neg q(z)$, $p(x,y) \vee \neg q(f(u))$, $p(x,x) \vee \neg q(f(u))$, $p(x,x) \vee \neg q(f(x))$. If a correct universally quantified theory does exist, then a refinement chain from $\Box$ to every clause in this theory exists because $\Box$ subsumes every clause. We say in this situation that ρ is *weakly complete*.

Until now we have only considered formulas which are conjunctions of a finite number of universally quantified clauses, especially definite program clauses. However, we may want to learn a concept expressed by a formula ϕ with existential variables. To solve such problems, one can consider the universally quantified Skolem standard form ψ of ϕ. It is well known that $\psi \models \phi$ but often $\phi \not\models \psi$. For instance, let the 3-ary predicate p be interpreted in the set of real numbers $\mathbf{R}$ as $p(x,y,z)$ is true iff $xy = z$. The concept that for an arbitrary $z \in \mathbf{R}$, there are x, y such that $xy = z$ can be expressed by $\phi = \forall z \exists x \exists y \; p(x,y,z)$. A standard form of the formula ϕ is $\psi = \forall z \; p(f(z), g(z), z)$ for new function symbols f, g. A model of ϕ is also a model of ψ only when f and g are interpreted in certain ways, e.g. $f(z) = 2z$ and $g(z) = z/2$, but we would like to check the truth value of ϕ directly. Moreover, most learning systems in ILP use function-free languages with constants. The extra function symblos do not suit the syntax of such a system. In a database we may have an integrity constraint $\forall x \forall y \exists z \neg sell(x,y) \vee supply(x,y,z)$: if a shop x sells an item y, there must be a company z which supplies x with y. Of course we can define one particular supplier as $f(x,y)$ but we have to change f when we consider another supplier. Actually an existential variable z expresses that as long as there is a supplier, it does not really matter which one it is. It seems now we emphasize something less important, namely a special supplier which is defined as $f(x,y)$. That means the standard forms are sometimes unnatural. In description logic (DL) existential quantifiers apper very often in formulas. For example, let us consider a constraint of concepts in DL: $\exists Friend.Dutch \sqsubseteq \exists Friend.European$: people who have Dutch friend(s) is a subset of people who have European friend(s). Let E, D and F stand for $European$, $Dutch$ and $Friend$, respectively. This constraint is equivalent to the following first order formulas:

$\forall x(\exists y F(x,y) \wedge D(y) \rightarrow \exists z F(x,z) \wedge E(z))$

$\Leftrightarrow \forall x((\neg \exists y F(x,y) \wedge D(y)) \vee (\exists z F(x,z) \wedge E(z)))$

$\Leftrightarrow \forall x \forall y \exists z(\neg F(x,y) \vee \neg D(y) \vee F(x,z)) \wedge (\neg F(x,y) \vee \neg D(y) \vee E(z))$.

This example shows to handle existential variables is important if we want to apply ILP techniques for learning in description logic. To add expressiveness we should consider learning PCNF with existential variables in ILP.

To assure a refinement operator being complete, we need to take small steps for refinements such that refinement chains will not skip the correct solutions. Let us first consider a search space of universal quantified theories. Since every substitution is a composition of some elementary substitutions so we choose elementary substitutions to define refinements[S81, NW97]. This idea motivates us to use (elementary) substitutions to refine a PCNF. However, the usual substitutions often generalize a formula instead of specializing it because of the existential variables. Let $\phi = \forall x \exists y p(x, y)$ and $\psi = \forall x p(x, x)$. Then $\psi \models \phi$ but $\phi \not\models \psi$. Therefore we will define a new type of substitutions to specialize a PCNF. Based on our substitutions and adding literals, we can define a *downward* refinement operator ρ which is *weakly complete*: For every ϕ there is a refinement chain from $\top$ to ϕ, i.e., there is an n such that $\phi \in \rho^n(\top)$.

Generalizing and specializing a formula with existential quantifiers have also been considered in [GF96]. A formula there involves only one clause. The variables in the head of a clause are universally quantified. The variables not in the head are quantified separately in the body by existential and universal quantifiers. Some rules are given to manipulate the variables only in the body. It seems that the rules are motivated by the following principle: If the body is generalized, then the formula is specialized and vice versa. [GF96] adopts neither PCNF in general nor an uniform approach with substitutions.

In this article we begin with establishing some properties of PCNF. We then define (elementary) substitutions and prove that they specialize a formula. Based on these substitutions we can define a refinement operator ρ. We prove that ρ is weakly complete. To establish the foundation of learning PCNF with ρ, we will end with an analysis of the complexities of this refinement operator applied in different search spaces. At last we explain briefly our first step in implementation. If $I_1, I_2, \ldots, I_n$ is a set of interpretations, the system PCL finds a PCNF ϕ such that every I_j is a model of ϕ. Hence we have generalized the Claudien system[RD97] which only deals with the standard forms.

2 Prenex Conjunctive Normal Forms

In the first subsection we give some well known definitions and results in logic (see [CL73, NW97]). In the second subsection we establish two important lemmas for later use. In the last subsection we consider the effects of adding literals to a clause in a PCNF.

2.1 Preliminaries

In this article we consider a first order logical language $\mathcal{L}$ with a finite number of function and predicate symbols.

Definition 1. An *interpretation* I with domain D of a logic language $\mathcal{L}$ consists of the following: (a) Each n-ary function symbol f in $\mathcal{L}$ is assigned a mapping I_f from D^n to D. (b) Each n-ary predicate symbol p in $\mathcal{L}$ is assigned a mapping I_p from D^n to $\{true, false\}$. For simplicity, we use f instead of I_f and p instead of I_p.

Definition 2. Let I be an interpretation of the language $\mathcal{L}$ with domain D. Let V be a set of variables, then a mapping $\theta : V \to D$ is called a *variable assignment* from V. Given a variable assignment θ from the set of variables of formula ϕ, we can check if ϕ is true or false under I and θ. If ϕ is a closed formula, then the truth value of ϕ is independent of the variable assignment we choose. I is a *model* of the closed formula ϕ if ϕ is true under I. Formula ψ *logically implies* formula ϕ, denoted by $\psi \models \phi$, if every model of ψ is also a model of ϕ. Formulas ψ and ϕ are said *logically equivalent*, denoted by $\psi \Leftrightarrow \phi$, if they have the same models.

Definition 3. A *clause* is a disjunction of a finite number of literals. A *prenex conjunctive normal form* (PCNF) ψ is a closed formula $q_1x_1q_2x_2 \dots q_nx_n(C_1 \wedge C_2 \wedge \dots \wedge C_m)$ where every q_i is a quantifier ($\exists$ or $\forall$) and every C_j is a clause. $q_1x_1q_2x_2 \dots q_nx_n$ is the *prenex* of ψ and $C_1 \wedge C_2 \wedge \dots \wedge C_m$ is the *matrix* of ψ. We denote ψ often by $q_1x_1q_2x_2 \dots q_nx_nM(x_1, \dots, x_n)$ or $Q(x_1, \dots, x_n)M(x_1, \dots, x_n)$ or $Q(\overline{x})M(\overline{x})$ or $Q(\psi)M(\psi)$ or QM. We call a variable in $Q(\psi)$ *universal* or *existential*; depending on how it is bound; the sets of existential and universal variables are denoted by $\mathrm{eVar}(\psi)$ and $\mathrm{uVar}(\psi)$, respectively. We have $\mathrm{Var}(\psi) = \mathrm{uVar}(\psi) \cup \mathrm{eVar}(\psi)$.

Note that if $\{y_1, \dots, y_m\} \supseteq \{x_1, \dots, x_n\}$, then $Q(\overline{y})M(\overline{x}) \Leftrightarrow Q(\overline{x})M(\overline{x})$.

Theorem 4. *If ϕ is a closed formula, then there exists a PCNF ψ such that ϕ and ψ are logically equivalent.*

2.2 Some properties of PCNF

We often need to check if some interpretation I is a model of a PCNF ψ. Lemma 5 in this subsection gives a necessary and sufficient condition for I to be a model of ψ. Lemma 6 tells that the truth value of a formula has often to do with the positions of variables in the prenex. We first give an example to motivate these two lemmas. Given a variable assignment θ defined on a subset of variables in ψ, we often talk about $\psi\theta$ (replacing variables by values given by θ) as if θ is a usual substitution from variables to terms (which are defined using the language itself instead of the domain of interpretation).

Example 1. Let $\psi = \exists x \forall y p(x, f(y))$ and $\phi = \forall y \exists x p(x, f(y))$. Then $\psi \models \phi$: Let I be an interpretation with domain D. Then ψ is true under $I \Leftrightarrow$ there is $d \in D$ such that for every $e \in D$, $p(d, f(e))$ is true. The choice of d does not depend on e. On the other hand, ϕ is true under $I \Leftrightarrow$ for every $e' \in D$, $\exists x p(x, f(e'))$ is true $\Leftrightarrow$ for every $e' \in D$, there is $d' \in D$ such that $p(d', f(e'))$ is true. Usually the choice of d' may depend on e' but here we can use the same d as ψ.

Lemma 5. *Let $\psi = q_1x_1 \dots q_nx_nM(x_1, \dots, x_n)$. An interpretation I with domain D is a model of ψ iff for every assignment $\sigma : \mathrm{uVar}(\psi) \to D$, there is an assignment $\gamma : \mathrm{eVar}(\psi) \to D$ such that the following two conditions are satisfied: (a) $M(\sigma \cup \gamma)$ is true under I. (b) the definition of γ on $x_i \in \mathrm{eVar}(\psi)$ depends only on how σ and γ are defined on $\{x_1, x_2, \dots, x_{i-1}\}$, i.e., an element in D can be assigned to x_i after the assignment of all $x_j, j < i$ has been done.*

Proof Let I be an interpretation with domain D. Let $\mathrm{eVar}(\psi) = \{x_{i_1}, \dots, x_{i_k}\}$ where $i_1 < \dots < i_k$. Let $\mathrm{Var}_1 = \{x_i | i < i_1\}$, $\mathrm{Var}_j = \{x_i | i_{j-1} < i < i_j\}$ for

$1 < j \leq k$ and $\mathrm{Var}_{k+1} = \{x_i | i > i_k\}$.
$\Rightarrow$: Let $\sigma : \mathrm{uVar}(\psi) \to D$ be a variable assignment. We want to find $\gamma : \mathrm{eVar}(\psi) \to D$ such that $M(\sigma \cup \gamma)$ true under I. Let $\sigma_1 = \sigma|\mathrm{Var}_1$, i.e. σ restricted to variables in Var_1. Then there is a $d_1 \in D$ such that $\psi(\sigma_1 \cup \{x_{i_1}/d_1\})$ is true. This implies for variable assignment $\sigma_2 = \sigma|\mathrm{Var}_2$ there is $d_2 \in D$ such that $M(\sigma_1 \cup \{x_{i_1}/d_1\} \cup \sigma_2 \cup \{x_{i_2}/d_2\}) = M(\sigma_1 \cup \sigma_2 \cup \{x_{i_1}/d_1, x_{i_2}/d_2\})$ is true under I. We can go on this way and say that for every $\sigma = \sigma_1 \cup \sigma_2 \cup \ldots \cup \sigma_{k+1}$, there is a variable assignment of $\mathrm{eVar}(\psi)$ to D: $\gamma = \{x_{i_1}/d_1, \ldots, x_{i_k}/d_k\}$ such that $M(\sigma \cup \gamma)$ is true. Note that x_{i_j}/d_j in γ may depend on how $x_k, k < i_j$ are assigned by σ and γ.
$\Leftarrow$: To prove I is a model of ψ we have to prove the following: For every variable assignment σ_1 of Var_1, there is a $d'_1 \in D$ (yet to be found) such that $\psi(\sigma_1 \cup \{x_{i_1}/d'_1\})$ is true under I. This means for every σ_2 of Var_2, there is a $d'_2 \in D$ (yet to be found) such that $\psi(\sigma_1 \cup \{x_{i_1}/d'_1\} \cup \sigma_2 \cup \{x_{i_2}/d'_2\})$ is true under I. We can go on this way and let the collection of σ_i be $\sigma = \sigma_1 \cup \sigma_2 \cup \ldots \cup \sigma_{k+1}$. By the given condition, for the variable assignment σ there is a $\gamma = \{x_{i_1}/d_1, \ldots, x_{i_k}/d_k\}$ such that $M(\sigma \cup \gamma)$ is true under I. The definition of γ on x_{i_j} depends on how σ and γ are defined on the variables before x_{i_j}. That means we can choose $d'_1 = d_1, d'_2 = d_2, \ldots$. Thus we have proved the sufficiency of the condition.

Lemma 6. *Let*
$$\psi = q_1 x_1 \ldots \exists x_i \ldots \forall x_j \ldots q_n x_n M, \quad \phi = q_1 x_1 \ldots \forall x_j \ldots \exists x_i \ldots q_n x_n M$$
and there is no other existential variable between x_i and x_j in the prenexes of these two formulas. Then $\psi \models \phi$.

Proof Suppose I is a model of ψ. We want to prove that I is also a model of ϕ. Let $\sigma : \mathrm{uVar}(\phi) \to D$. By Lemma 5, there is a variable assignment γ of $\mathrm{eVar}(\psi)$ such that $M(\psi)(\sigma \cup \gamma)$ is true under I. The assignment on $x_k \in \mathrm{eVar}(\psi)$ may depend on the definition of σ and γ on variables before x_k in $Q(\psi)$. Since $\mathrm{uVar}(\psi) = \mathrm{uVar}(\phi)$ and $M(\psi) = M(\phi)$, we have $M(\phi)(\sigma \cup \gamma)$ true under I. By Lemma 5, we can say I is a model of ϕ if γ in $x_k \in \mathrm{eVar}(\phi)$ depends on the assignment of variables before x_k in $Q(\phi)$. Note that $Q(\phi)$ interchanges only the order of $\exists x_i$ and $\forall x_j$ in $Q(\psi)$. The assignment of x_i in γ depends only on the assignment of σ, γ on $x_1, \ldots, x_{i-1}$ which appear before $\exists x_i$ in $Q(\phi)$ too. If $x_k \in \mathrm{eVar}(\phi)$ is another existential variable, then it is not between x_i and x_j so $x_1, \ldots, x_{k-1}$ are still before x_k in $Q(\phi)$. That means I is a model of ϕ.

2.3 Adding literals

A classic refinement step for a universal quantified PCNF extends a clause in the matrix with an extra literal containing new variables. This will also be done for our refinement operator for PCNF. For this we need the the following results which are based on the fact that the disjunctions of two formulas is true if at least one of them is true.

Lemma 7. *Let $\psi = q_1 x_1 \ldots q_n x_n C_1 \wedge C_2 \ldots \wedge C_m$ where every C_i is a clause. Let L be a literal which contains only new variables $y_1, \ldots, y_k$ w.r.t. ψ. If $\phi_j = q_1 x_1 \ldots q_n x_n \forall y_1 \ldots \forall y_k\ C_1 \wedge \ldots \wedge (C_j \vee L) \ldots \wedge C_m$, then $\psi \models \phi_j$.*

We call the action in Lemma 7 *adding a u-literal*. Similarly we can prove the following lemma which *adds an e-literal* to a formula. Notice that these lemmas are true even L contains some old variables. (Of course we should then only quantify the new variables.) However, we would like to have as few refinements as possible so we will later choose only ϕ_j using new variables in L as refinements of ψ. In fact, the truth value of the new formula does not depend on the locations of the quantification of these new variables in the prenex. We choose to place the new existential variables at leftmost and the new universal variables at rightmost of the prenex because this does not contradict our feeling that the existential variables before the universal variables are stronger formulas (see Lemma 6). Moreover, this way will make the proof of the weak completeness of the refinement operator ρ more straightforward.

Lemma 8. *Let* $\psi = q_1x_1 \ldots q_nx_nC_1 \wedge C_2 \ldots \wedge C_m$ *where every* C_i *is a clause. Let* L *be a literal which contains only new variables* $y_1, \ldots, y_k$ *w.r.t.* ψ*. If* $\phi_j = \exists y_1 \ldots \exists y_k q_1x_1 \ldots q_nx_n\ C_1 \wedge \ldots \wedge (C_j \vee L) \ldots \wedge C_m$*, then* $\psi \models \phi_j$*.*

3 Substitutions and Specializations

Usually, a substitution θ replaces some variables by terms. For a universally quantified clause C we have $C \models C\theta$. This is not valid for PCNF as the following examples show. Thus we are motivated to define a new type of substitutions which specialize PCNF.

Example 2. Consider the following implications.
$\forall p(x) \models p(a)$ and $p(a) \not\models \forall p(x)$
$p(a) \models \exists x\, p(x)$ and $\exists x p(x) \not\models p(a)$
$\exists x p(x,x) \models \exists x \exists y\, p(x,y)$ and $\exists x \exists y\, p(x,y) \not\models \exists x p(x,x)$
Moreover, a unification of two universally quantified variables does not always specialize a PCNF. Let
$\psi = \forall x \exists y \forall z p(x,y,z)$, $\phi = \forall x \exists y p(x,y,x)$, $\phi' = \exists y \forall z p(z,y,z)$,
and $I = \{p(t, f(t), t') |\, t, t' \text{ ground}\}$. Then I is a model of ψ and ϕ but not a model of ϕ'. For ϕ' true under I, we need an s such that $p(t,s,t)$ is true for every t.

3.1 Elementary substitutions for PCNF

A matrix in a PCNF can be pictured as a tree, with the root on top. At each node, number downgoing branches 1, 2, 3, etc. from left to right. Each node and the tree hanging from it is given by the path that leads to it from the top. For example, let $M = p(x,y) \wedge (p(x,x) \vee \neg q(f(x)))$. The second clause has position $\langle 2 \rangle$. $\neg q(f(x))$ has position $\langle 2,2 \rangle$, and $f(x)$ has position $\langle 2,2,1 \rangle$, etc.

Definition 9. An *substitution for a matrix M* has the form $\theta = \{(t_1/s_1, p_1), \ldots, (t_n/s_n, p_n)\}$. $M\theta$ is a matrix formed by using M and θ: for every i, the term at position p_i in M is t_i and t_i should be replaced by s_i. For example, if $M =$

$p(x,y) \wedge (p(x,x) \vee \neg q(f(x)))$ and $\theta = \{(x/f(z), \langle 1,1 \rangle), (f(x)/g(z), \langle 2,2,1 \rangle)\}$, then $M\theta = p(f(z),y) \wedge (p(x,x) \vee \neg q(g(z)))$. It is easy to see that the old definition of a substitution is a special case of the new kind of substitutions. In such a case we use the old notation where the positions are not needed.

Using the substitutions for the matrix of a PCNF, we can define the substitutions for the PCNF itself.

Definition 10. Let $\psi = q_1 x_1 \ldots q_n x_n C_1 \wedge \ldots \wedge C_m = Q(\psi)M(\psi)$ be a PCNF. There are the following 5 types of *elementary substitutions* for ψ.

The first two types have to do with universal variables. Notice that the old definitions of elementary substitutions[NW97] for a universally quantified clause are special cases for these two types.

- Let $x_i, x_j \in \mathrm{uVar}(\psi)$ and $i < j$. An *elementary u-unification* $\theta = \{x_j/x_i\}$ for ψ can be applied to ψ such that $\psi\theta = q_1 x_1 \ldots q_n x_n (M(\psi)\theta)$. For example let $\psi = \forall x \forall y p(f(x), y)$ and $\theta = \{y/x\}$. Then $\psi\theta = \forall x \forall y p(f(x), x)$ which is equivalent to $\forall x p(f(x), x)$.
- Let $x_i \in \mathrm{uVar}(\psi)$. If $t = f(y_1, \ldots, y_k)$, where $y_1, \ldots, y_k$ are new distinct variables w.r.t. ψ, then $\theta = \{x_i/t\}$ is called an *elementary u-substitution.* The new formula $\psi\theta$ is constructed as follows. All x_i-occurrences in the matrix of ψ are replaced by t simultaneously, i.e. $M(\psi\theta) = M(\psi)\theta$. Moreover, the $\forall x_i$ in the prenex of ψ is replaced by $\forall y_1 \forall y_2 \ldots \forall y_k$. For example, let $\psi = \forall x \exists y p(x,y)$ and $\theta = \{x/f(u,v)\}$. Then $\psi\theta = \forall u \forall v \exists y p(f(u,v), y)$.

The third and fourth types have to do with the existential variables:

- Let $x_i \in \mathrm{eVar}(\psi)$ and let $\{(x_i, p_1), \ldots, (x_i, p_k)\}$ be a proper subset of the x_i-occurrences in $M(\psi)$. If z is a new variable, then $\theta = \{(x_i/z, p_1), \ldots, (x_i/z, p_k)\}$ is called an *elementary e-antiunification* and $\psi\theta$ is the PCNF $q_1 x_1 \ldots \exists x_i \exists z q_{i+1} x_{i+1} \ldots q_n x_n (M\theta)$. For example, let $\psi = \exists x p(x,x)$ and $\theta = \{x/z, \langle 2 \rangle\}$. Then $\psi\theta = \exists x \exists z p(x,z)$.
- Let $t = f(x_{i_1}, \ldots, x_{i_m})$ which contains only distinct existential variables. Let $i_j = \max\{i_1, \ldots, i_m\}$. Let $\{(t, p_1), \ldots, (t, p_k)\}$ be occurrences in $M(\psi)$. If z is a new variable, then $\theta = \{(t/z, p_1), \ldots, (t/z, p_k)\}$ is called an *elementary e-substitution for* ψ. We define $\psi\theta = q_1 x_1 \ldots q_{i_j} x_{i_j} \exists z q_{i_j+1} x_{i_j+1} \ldots q_n x_n (M\theta)$. For example, let $\psi = \forall x \exists y \exists u \exists v p(x,u) \wedge (p(x,y) \vee \neg q(f(u,v)))$. If $\theta = \{(f(u,v)/z, \langle 2,2,1 \rangle)\}$, then $\psi\theta = \forall x \exists y \exists u \exists v \exists z\ p(x,u) \wedge (p(x,x) \vee \neg q(z))$.

The last type is related to interchanging the positions of an existential variable and an universal variable in $Q(\psi)$:

- Suppose $x_i \in \mathrm{eVar}(\psi)$, $x_j \in \mathrm{uVar}(\psi)$ and $i < j$. If there is no other existential variable between x_i and x_j in $Q(\psi)$, then $\{(x_i, x_j)\}$ denotes an *elementary eu-substitution.* It interchanges the positions of x_i and x_j in the prenex $Q(\psi)$. For example, let $\theta = \{(x,y)\}$. Then $(\exists x \forall y p(x,y))\theta = \forall y \exists x p(x,y)$.

3.2 Specializations from substitutions

For every elementary substitutions θ for ψ we can prove that $\psi \models \psi\theta$.

Lemma 11. *For a PCNF $\psi = q_1x_1 \ldots q_nx_nM(\psi)$, let $x_i, x_j \in \mathrm{uVar}(\psi), i < j$ and let $\theta = \{x_j/x_i\}$ be an elementary u-unification. Then $\psi \models \psi\theta$.*

Proof Let $\phi = \psi\theta$ and I be a model of ψ with domain D. We want to use Lemma 5 to prove that I is also a model of ϕ. Let σ be a variable assignment of $\mathrm{uVar}(\phi)$. Then σ induces a variable assignment σ' of $\mathrm{uVar}(\psi)$ because we can let $\sigma'(x_j) = \sigma(x_i)$. By Lemma 5, for σ' we can find a γ defined on $\mathrm{eVar}(\psi) = \mathrm{eVar}(\phi)$ such that $\psi(\sigma' \cup \gamma)$ true under I. It is clear that $M(\psi)(\sigma' \cup \gamma) = M(\phi)(\sigma \cup \gamma)$. That means $M(\phi)(\sigma \cup \gamma)$ is true under I. If $x \in \mathrm{eVar}(\psi) = \mathrm{eVar}(\phi)$, then the definition of γ on x depends only on the assignment of σ' and γ on variables before x in $Q(\psi)$. These variables are before x too in $Q(\phi)$. By applying the sufficient condition of Lemma 5 again we have I is a model of ϕ.

Lemma 12. *Let $x \in uVar(\psi)$ and $\theta = \{x/t\}$ be an elementary u-substitution. Then $\psi \models \psi\theta$.*

Proof Let I be a model of ψ with domain D. Let $\mathrm{uVar}(\psi) = \{x, y_1, \ldots, y_k\}$ and $\mathrm{eVar}(\psi) = \{z_1, \ldots, z_m\}$. Let $\{u_1, \ldots, u_l\}$ be the set of (new) variables in t. To apply Lemma 5, we need to prove first that for every variable assignment $\sigma = \{y_1/d_1, \ldots, y_k/d_k\} \cup \{u_1/d_{k+1}, \ldots, u_l/d_{k+l}\}$ from $\mathrm{uVar}(\psi\theta)$ to D, there is a variable assignment γ of $z_1, \ldots, z_m$ such that $(M(\psi)\theta)(\sigma \cup \gamma)$ is true under I. Let $\mu = \{u_1/d_{k+1}, \ldots, u_l/d_{k+l}\}$. Let $\sigma' = \{y_1/d_1, \ldots, y_k/d_k\} \cup \{x/(t\mu)\}$. Then $(M(\psi)\theta)\sigma = M(\psi)\sigma'$. Since I is a model of ψ so there is a variable assignment γ of $z_1, \ldots, z_m$ such that $M(\sigma' \cup \gamma)$ is true by Lemma 5. That means $(M(\psi)\theta)(\sigma \cup \gamma)$ is true under I. Moreover, γ on z_i depends only on the variables before it in $Q(\psi)$. If x is before some z_i in $Q(\psi)$, then all u_j are before z_i in $Q(\psi\theta)$. The assignment of these u_j determines the assignment of x. Since the other variables before a z_i stays before z_i in $\psi\theta$ so we can say that z_i depends on all the variables (including u_j) in $P(\psi\theta)$ before z_i. We can apply the sufficiency in Lemma 5 to say that I is a model of $\psi\theta$.

Lemma 13. *Let $\psi = Q(\psi)M(\psi)$ be a PCNF. Let $\theta = \{(t/z, p_1), \ldots, (t/z, p_m)\}$ be an elementary e-substitution. Then $\psi \models \psi\theta$.*

Proof Let I be a model of ψ. Let t contain only the existential variables $x_{i_1}, \ldots, x_{i_k}$ where $i_1 < i_2 \ldots < i_k$. Then $\psi\theta = q_1x_1 \ldots q_{i_k}x_{i_k}\exists z q_{i_k+1}x_{i_k+1} \ldots q_nx_nM\theta$. $M(\psi)$ differs from $M(\psi\theta)$ only at $p_1, \ldots, p_k$. Clearly, $\mathrm{uVar}(\psi\theta) = \mathrm{uVar}(\psi)$ and $\mathrm{eVar}(\psi\theta) = \mathrm{eVar}(\psi) \cup \{z\}$. We want to prove for every variable assignment σ of $\mathrm{uVar}(\psi\theta)$ to D, there is a variable assignment γ of $\mathrm{eVar}(\psi\theta)$ to D such that $M(\psi)\theta(\sigma \cup \gamma)$ is true under I. Since I is a model of ψ, and the assignment σ is also an assignment of $\mathrm{uVar}(\psi)$, by Lemma 5 there is a variable assignment γ of $\mathrm{eVar}(\psi)$ to D such that $M(\psi)(\sigma \cup \gamma)$ is true under I. Moreover, if $x_i \in \mathrm{eVar}(\psi)$, then $\gamma(x_i)$ depends only on how σ and γ behave on variables before x_i in $Q(\psi)$. Notice that at every position p_j in $M(\psi)(\sigma \cup \gamma)$ we have in fact $t\gamma$. We consider the subsitution $\gamma' = \gamma \cup \{z/t\gamma\}$. Then $M(\psi)(\sigma \cup \gamma) = M(\psi)\theta(\sigma \cup \gamma')$.

Notice z is behind all variables $x_1, \ldots, x_{i_k}$ in $Q(\psi\theta)$. Thus z depends also only on how σ and γ' defined on the variables before it. By applying Lemma 5, we have I is a model of $\psi\theta$.

In a similar way we can prove $\psi \models \psi\theta$ for an elementary e-antiunification θ. By the lemmas in this subsection and Lemma 6, we have the following definition and theorem.

Definition 14. Let ψ be a PCNF. Suppose θ_1 is an elementary substitution w.r.t. ψ and θ_i is elementary w.r.t. $(\ldots(\psi\theta_1)\theta_2\ldots)\theta_{i-1}$ for every $i = 2, \ldots, n$. Let $\theta = \theta_1 \ldots \theta_n$ be the composition of these θ_i, i.e. $\psi\theta = (\ldots(\psi\theta_1)\theta_2)\ldots)\theta_n$. Then θ is called a *substitution* w.r.t. ψ.

Theorem 15. *For a PCNF ψ and a substitution θ defined as above, $\psi \models \psi\theta$.*

4 A Refinement Operator

Let the search space S be the set of all PCNF of a first order logical language $\mathcal{L}$ with a finite number of predicates and function symbols. Let the top $\top \in S$ be the conjunction of a positive number of empty clauses, i.e. $\top = \Box \wedge \Box \ldots \wedge \Box$. A *refinement operator* on S is a function $\rho : S \to 2^S$ (the set of all subsetes of S). A refinement operator ρ is *downward* if for every ψ and $\phi \in \rho(\psi)$, we have $\psi \models \phi$. A *refinement chain* from ψ to ϕ is a sequence $\psi_0, \psi_1, \ldots, \psi_k$ in S such that $\psi_0 \Leftrightarrow \psi$, $\psi_k \Leftrightarrow \phi$ and $\psi_i \in \rho(\psi_{i-1})$.

Definition 16. Let $\psi = q_1x_1 \ldots q_nx_nC_1 \wedge C_2 \ldots \wedge C_m$. Let ρ be a refinement operator on S defined by the following 7 items. The first three items have to do with elementary substitutions w.r.t. universal variables and adding u-literals. The next three items have to do with elementary substitutions w.r.t. existential variables and adding e-literals. The last item has to do with elementary eu-substitutions. Note that there are only a finite number of non-alphabetical variants in $\rho(\psi)$. Hence ρ is *locally finite* (see [Laird 88, NW97]).

1. For an *elementary u-unification* $\theta = \{y/x\}$, where $x, y \in \mathrm{uVar}(\psi)$, let $\psi\theta \in \rho(\psi)$.
2. For $x \in \mathrm{uVar}(\psi)$ and an *elementary u-substitution* $\theta = \{x/f(y_1, \ldots, y_k)\}$, let $\psi\theta \in \rho(\psi)$.
3. Let $L = p(y_1, \ldots, y_k)$ or $\neg p(y_1, \ldots, y_k)$, where $y_1, \ldots, y_k$ are new distinct variables w.r.t. ψ. For an arbitrary $j = 1, \ldots, m$, let ϕ_j be defined by *adding a u-literal* to ψ: $\phi_j = q_1x_1 \ldots q_nx_n\forall y_1 \ldots \forall y_k\ C_1 \wedge \ldots \wedge (C_j \vee L) \wedge \ldots \wedge C_m$. Then $\phi_j \in \rho(\psi)$.
4. Let $x \in \mathrm{eVar}(\psi)$ and $\{(x, p_1), \ldots, (x, p_k)\}$ be some (not all) x-occurrences in $M(\psi)$. For a new variable y and an *elementary e-antiunification* $\theta = \{(x/y, p_1), \ldots, (x/y, p_k)\}$, let $\psi\theta \in \rho(\psi)$.

5. Let $t = f(x_{i_1}, \ldots, x_{i_k})$ where every $x_{i_j} \in \mathrm{eVar}(\psi)$. Suppose all x_{i_j} are distinct and $\{(t,p_1), \ldots, (t,p_k)\}$ is a set of t-occurrences in $M(\psi)$. Then for the *elementary e-substitution* $\theta = \{(t/y,p_1), \ldots, (t/y,p_k)\}$, let $\psi\theta \in \rho(\psi)$.
6. Let $L = p(y, \ldots, y)$ or $L = \neg p(y, \ldots, y)$ be a literal with new variable y. Then for $j = 1, \ldots, m$, let $\phi_j \in \rho(\psi)$ be defined by *adding an e-literal*: $\phi_j = \exists y q_1 x_1 \ldots q_n x_n\ C_1 \wedge \ldots \wedge (C_j \vee L) \wedge \ldots \wedge C_m$.
7. Let $x \in \mathrm{eVar}(\psi)$ $y \in \mathrm{uVar}(\psi)$. Suppose x comes before y in $Q(\psi)$ and there is no other existential variable between x and y. Then for the *elementary eu-substitution* $\theta = \{(x,y)\}$, let $\psi\theta \in \rho(\psi)$.

By Theorem 15, Lemma 7 and Lemma 8, we have the following theorem.

Theorem 17. *If $\phi \in \rho(\psi)$, then $\psi \models \phi$, i.e. ρ is a downward refinement operator.*

5 The Operator ρ is Weakly Complete

In this section we will show that the refinement operator ρ is *weakly complete* in S. That is to say, for every ϕ in S, there is a finite refinement chain from $\top$ to ϕ. We show this by the following steps: $\top \longrightarrow \psi \longrightarrow \phi' \longrightarrow \phi$. Example 3 will illustrate these steps more concretely.

1. Replace every existential variable in $M(\phi)$ by a new constant not in ϕ and remove all the existential variables from $Q(\phi)$. Let the new PCNF be ψ. Then the variables in ψ are universally quantified. Let $M(\psi) = C_1 \wedge C_2 \ldots \wedge C_m$. Then $\Box$ subsumes C_i ($\Box \succeq C_i$) for every i.
2. Similar to a result about the classic refinement refinement operator [S81, Laird 88, LN94, NW97], we can prove that there is a chain from $\Box$ to every C_i. The combination of these chains will give a chain from $\top$ to ψ.
3. By using the elementary e-substitutions (item 4 of ρ) we can change the constant occurrences in ψ back to existential variables. This establishes a refinement chain from ψ to ϕ' which looks almost like ϕ but all existential variables appear before the universal variables.
4. Using eu-substitutions (item 7 of ρ) we can move the existential variables to the right and place them in good positions in the prenex. This means there is a chain from ϕ' to ϕ. Thus we have the weak completeness of ρ.

Example 3. We will give an example to show how a concrete finite chain from $\top$ to a given $\phi = \forall x \exists y((\neg p(x) \vee q(f(x)) \vee q(y)) \wedge r(y,a))$ looks like. Note that the chain is not unique. Such a chain exists for a general ϕ (Theorem 21) because of the following lemmas. We use an arrow $\xrightarrow{n}$ to denote a refinement step which uses the n-th item of ρ.

$\Box \xrightarrow{3} \forall x \neg p(x) \xrightarrow{3,3,3}$

$\forall x \forall u \forall v \forall w \forall w'((\neg p(x) \vee p(u) \vee q(v)) \wedge r(w,w') \xrightarrow{2}$

$\forall x \forall u_1 \forall v \forall w \forall w'(\neg p(x) \vee p(f(u_1) \vee q(v)) \wedge r(w,w')) \xrightarrow{1,1}$

$$\begin{aligned}
&\forall x\forall v\forall w'((\neg p(x) \vee p(f(x)) \vee q(v)) \wedge r(v, w')) \xrightarrow{2,2} \\
&\psi = \forall x((\neg p(x) \vee p(f(x)) \vee q(b)) \wedge r(b, a)) \xrightarrow{5} \\
&\phi' = \exists y\forall x((\neg p(x) \vee p(f(x)) \vee q(y)) \wedge r(y, a)) \xrightarrow{7} \\
&\phi = \forall x\exists y((\neg p(x) \vee p(f(x)) \vee q(y)) \wedge r(y, a))
\end{aligned}$$

Lemma 18. *Let C be a universally quantified clause. Then there is a finite chain of refinements from $\Box$ to C.*

Proof See [LN94] and subsection 17.4, [NW97]

Every universally quantified clause can be reached by a ρ-chain and hence a universally quantified PCNF can be reached by a ρ-chain.

Lemma 19. *Let $\psi = Q(\psi)M(\psi)$ be a universally quantified PCNF. Then there is a finite ρ-chain from $\top$ to ψ.*

Lemma 20. *Let $\phi = \exists y_1 \dots \exists y_n \forall x_1 \dots \forall x_m M(\phi)$ whose existential variables in the prenex appear before the universal variables. Let $b_1, \dots, b_n$ be different constants which do not appear in ϕ. Let the universally quantified ψ be ϕ after replacing variable y_i by b_i. Then there is a finite ρ-chain from ψ to ϕ.*

Proof Let the b_i-occurrences in ψ be $(b_i, p_{i_1}), \dots, (b_i, p_{i_k})$. For every $i = 1, \dots, n$, let $\theta_i = \{(b_i/y_i, p_{i_1}), \dots, (b_i/y_i, p_{i_k})\}$ which is an elementary e-substitution. Then $\psi\theta_n \dots \theta_1 = \phi$. These substitutions $\theta_i, i = 1, \dots, n$ correspond with item 5 of ρ.

Theorem 21. *Given a PCNF $\phi = Q(\phi)M(\phi)$, there is a finite ρ-chain from $\top$ to ϕ.*

Proof Suppose uVar$(\phi) = \{x_1, \dots, x_m\}$ and eVar$(\phi) = \{y_1, \dots, y_n\}$. Let $i_1, i_2, \dots, i_n$ where $i_1 < i_2 \dots < i_n$ be the places of $y_1, y_2 \dots, y_n$ in $Q(\psi)$, respectively. We divide the proof in three steps:

1. Let the universally quantified ψ be constructed as follows. Suppose $b_1, \dots, b_n$ are different constants not in ϕ. $Q(\psi)$ is $Q(\phi)$ by removing the existential variables and $M(\psi)$ is $M(\phi)$ by replacing every y_i by the constant b_i. Then there is a finite refinement chain from $\top$ to ψ by Lemma 19.
2. Let $\phi' = \exists y_1 \dots \exists y_n \forall x_1 \dots \forall x_m M(\phi)$. Then there is a finite refinement chain from ψ to ϕ' by Lemma 20.
3. Now ϕ' is almost the same as ϕ except the order of the existential variables and universal variables in the prenex. We can use elementary eu-substitution (i.e. item 7 of ρ) to find a refinement chain from ϕ' to ϕ in the following way. We move first $\exists y_n$ to place i_n in the prenex by interchanging it and the universal variables $\forall x_1 \forall x_2$, etc. stepwisely. We then move $\exists y_{n-1}$ to place i_{n-1} by interchanging its place and its right neighbors in the prenex stepwisely. These movements can be done without problem because $i_{n-1} < i_n$. Eventually we move $\exists y_1$ to place i_1 in the prenex. Now we have the formula ϕ after finite refining steps.

In [LN94, LN98, NW97] some properties of refinement operators are investigated, in particular *ideal refinement operators.* A refinement operator is ideal if it is locally finite, complete and proper. An ideal refinement operator does not exist in the search space of all universal quantified theories ordered by implication. We can consider our refinement operator restricted to the universal theories. It is surely not ideal because it contradicts the old results. We can also directly see that using item 3 will give sometimes equivalent theories. For example, $\forall x \forall y p(x,y) \Leftrightarrow \forall x \forall y \forall u \forall v p(x,y) \vee p(u,v)$. Therefore, ρ is not proper.

6 The Complexity of the Refinement Operator

In this article we investigate some complexity problems related to ρ. These complexities about the number of refinements of a formula will be expressed by some upper bounds. In fact, we can not have smaller complexities than these upper-bounds because we can find examples which have indeed so many refinements. We will first consider function-free logical languages because ILP often uses this kind of languages in implementations. We will call a function *exponential* w.r.t. n if it can be majorized by $n \to 2^{p(n)}$ where $p(n)$ is a polynomial of n.

6.1 $\mathcal{L}$ is function-free

Consider a *function-free* logical language $\mathcal{L}$, i.e. the only functions are constants.

1. The following constants are used to specify a search space S of PCNF.
   - $P =$ the number of predicate symbols in $\mathcal{L}$.
   - $F =$ the number of constants in $\mathcal{L}$.
   - $K =$ the maximal number of arguments of a predicate in $\mathcal{L}$.
   - $U =$ the maximal number of universal variables allowed for $\psi \in S$.
   - $E =$ the maximal number of existential variables allowed for $\psi \in S$.
   - $M =$ the maximal number of clauses allowed for $\psi \in S$.
   - $L =$ the maximal number of literals allowed for a clause in $\psi \in S$.

   Let $N = \max\{P, F, U, E, M, L, K\}$.
2. Let $\psi = Q(\psi)M(\psi)$ be a PCNF. We use the following notations in the computations of upper bounds: $Q(\psi)$ can be divided into a sequence of blocks of existential and universal variables: $e_1, u_1, e_2, u_2 \ldots, e_k, u_k$. That means: $Q(\psi)$ begins with e_1 existential quantifiers and variables, followed by u_1 universal quantifiers and variables, etc. Let $|u_i|$ or $|e_i|$ be the number of variables in the i-th block.

The number of elements in $\rho(\psi)$ for function-free $\mathcal{L}$: for $i = 1, 2, \ldots, 7$, the i-th case below finds an upper bound of the number of refinements defined by the i-th item in ρ.

1. The number of u-unifications is bounded by $\binom{U}{2} = \frac{U(U-1)}{2} \leq N^2$.

2. The number of u-substitutions is bounded by $U \cdot F \leq N^2$.
3. For every clause in $M(\psi)$ there are $2P$ ways to add u-literals. The total number of ways for ψ is bounded by $2MP \leq 2N^2$.
4. A clause can contain at most LK occurrences of variables and the total number of variable occurrences in $M(\psi)$ is bounded by LKM. The number of subsets of variable occurrences is bounded by 2^{LKM}. Hence the total number of elementary e-antiunifications is bounded $2^{LMK} \leq 2^{N^3}$.
5. The number of constant occurrences in a clause is at most LK and the total number of constant occurrences in $M(\psi)$ is bounded by LKM. The number of subsets of constant occurrences is bounded by 2^{LKM}. Hence the number of elementary e-substitutions is bounded by $2^{LMK} \leq 2^{N^3}$. (In fact, we can reason one upper bound for case 4 and case 5 together and say that 2^{N^3} is their upper bound.)
6. The number of ways to add an e-literal is bounded by $2MP \leq 2N^2$.
7. For every $i = 1, \ldots, k$, only the last existential variable in block e_i may exchange its position with one of the variables in u_i because we may move an existential variable to the right only when it does not jump over other existential variables. Hence there are only $|u_i|$ ways. In total there are $|u_1| + |u_2| + \ldots + |u_k| \leq U \leq N$ ways to perform elementary eu-substitutions.

For an upper bound of $|\rho(\psi)|$ we can add these numbers given above.
$|\rho(\psi)| \leq N^2 + N^2 + 2N^2 + 2^{N^3} + 2^{N^3} + 2N^2 + N$.

Theorem 22. *Consider a function-free logical language $\mathcal{L}$. The number of refinements of a PCNF in S is bounded by an exponential function of N.*

6.2 $\mathcal{L}$ is not function-free

Let us now consider a logical language $\mathcal{L}$ which is not function-free. The complexity of refinements will increase enormously as we may expect. Now let F be the number of function symbols in $\mathcal{L}$. Let K be the maximum number of arguments allowed for a predicate or a function symbol in $\mathcal{L}$ and let D be the maximal number of nestings (depth of a term or the length of the path to a term in the tree of the matrix of a PCNF) allowed in $\psi \in S$. The same definitions will be used for P, E, U, etc. Moreover, let N be the maximum of all these constants. We can use similar arguments to compute the upper bounds.

For types of 1, 2, 3, 6, 7 of $\rho(\psi)$ the arguments are identical as in the function-free situation. For type 4 we have K^D instead of K: A clause can contain at most LK^D occurrences of variables, in particular, existential variables. The number of subsets of variable occurrence in $M(\psi)$ is bounded by 2^{MLK^D}. Hence the number of elementary e-antiunifications is bounded by $2^{MLK^D} \leq 2^{N^{N+2}}$. Likewise, the total number of e-substitutions defined by type 5 in $\rho(\psi)$ is bounded by $2^{MLK^D} \leq 2^{N^{N+2}}$.

Theorem 23. *Consider a logical language $\mathcal{L}$ which is not function-free. The number of refinements of a PCNF in S has an upper bound which is double exponential w.r.t. N.*

This theorem tells us the number of refinements can grow extremely fast when the nestings of functions in formulas increase. This discourages one to implement learning systems on PCNF which are not function-free.

6.3 The complexity of ρ for universally quantified PCNF

For comparison we investigate the complexities of refinement operators in classical ILP systems, i.e. ρ restricted to a search space with universally quantified PCNF, (denoted by ρ_L in [NW97].)

The number of elements in $\rho(\psi)$: If ψ does not contain existential variables, then cases 4, 5, 6 and 7 in ρ will not be used. That means the number $|\rho(\psi)| \leq N^2 + N^2 + 2N^2 = 4N^2$ which is a polynomial function of N.

The total number of ρ-steps needed to get to $\rho^d(\top)$: We have seen that for some fixed $\phi \in \rho^d(\top)$, the number of universal PCNF in $\rho(\phi)$ is bounded by a polynomial of N. We want also to know how many ρ-steps have to be used to find $\rho^d(\top)$ if we start from $\top$. That means the complexity of $\Sigma_{i=1}^{d-1}|\rho^i(\top)|$. For example, let a be a constant and f be a 1-ary function symbol. Let p, q be a 2-ary and a 1-ary predicates, respectively. We will omit the universal quantifiers in the following computation so that the formulas look neater. Then

$\rho(\top) = \{p(x,y), q(x), \neg p(x,y), \neg q(x)\}$,
$\rho(p(x,y)) = \{p(x,x), p(x,a), p(a,y), p(f(z),y), p(x,f(z)),$
$\qquad p(x,y) \vee q(z), p(x,y) \vee \neg q(z)\}$,
$\rho(q(x)) = \{q(a), q(f(z)), q(x) \vee p(u,v), q(x) \vee \neg p(u,v)\}, \quad \ldots \;,$

This means we need 1 ρ-step to find $\rho(\top)$, $5(=1+4)$ ρ-steps to find $\rho^2(\top)$, and $27(=5+22)$ ρ-steps to find $\rho^3(\top)$, etc. To compute the number of ρ-steps needed for $\rho^d(\top)$ from $\top$, we can consider a tree with $d-1$ layers and at most $4N^2$ outgoing branches for every node. If $X = 4N^2$ then the total number of nodes is bounded by $1+X+X^2+X^3+\ldots+X^{d-1} \leq (X-1)(1+X+\ldots+X^{d-1}) = X^d-1$. If $d \leq N$, then we have an exponential function of N. This explains perhaps why most learning systems use function-free languages even for universal PCNF.

Theorem 24. *Let S be restricted to universal PCNF. Let $\rho^1(\top) = \rho(\top)$ and $\rho^d(\top) = \bigcup\{\rho(\phi) |\, \phi \in \rho^{d-1}(\top)\}$. For some fixed $\phi \in \rho^d(\top)$, the number of universal PCNF in $\rho(\phi)$ is bounded by a ploynomial of N. The total number of ρ-steps needed to compute $\bigcup_{d\leq N} \rho^d(\top)$ is bounded by an exponential function of N.*

7 Learning PCNF in Practice

Based on the refinement operator given in the previous sections, we have extended a simple version of the Claudien system (see [RD97]) to a learning system PCL (abbreviation of *PCNF Claudien*) and implemented it in Prolog. Just like Claudien, learning by interpretations[RD97] is also used in PCL. Claudien learns a universal clausal theory w.r.t. a set of positive examples (interpretations) such that each example is a model of the theory. This clausal theory can be considered as a set of regularities or integrity constraints satisfied by these examples. Of course, the purpose of PCL is to learn more expressive PCNF with existential variables.

Consider a finite set of scenes (*Bongard interpretations*) as positive examples. A scene contains several figures: each figure has properties like *shape, size,...* and these figures are related to each other indicated by *in, above, left_of,....* Claudien is able to find clauses like:

$\forall x \forall y(\text{shape}(y, \text{circle}) \leftarrow \text{figure}(x), \text{figure}(y), \text{in}(x, y))$,

i.e., for every figure x and y in a scene, when x is inside y, then y must be a circle. The following rule cannot be found by Claudien, but PCL can:

$\forall x \exists y((\text{in}(x, y) \leftarrow \text{shape}(x, \text{triangle})) \wedge \text{shape}(y, \text{circle}))$,

i.e., each triangle is in at least one circle.

To extend Claudien towards PCL, several issues have to be addressed. In the following subsections we discuss the most important ones.

7.1 Testing interpretation in a search space

In Claudien, a clause $head \leftarrow body$ is true for an interpretation (positive example) if the Prolog query $? - body, not(head)$ fails for that example. This works only if the clauses are *range restricted*, meaning that all variables in the head should also occur in the body. Indeed, consider a Herbrand interpretation $I = \{p(a, b), q(a)\}$ and $\phi = \forall x \forall y(p(x, y) \leftarrow q(x))$. ϕ is false because $p(a, a) \leftarrow q(a)$ is false. On the other hand, we get *No* as the answer of the query ?- $q(x), not(p(x, y))$ because $q(a), not(p(a, b))$ is false. For PCL, we also consider range restricted PCNFs. The following definition can be found in [N82]: a PCNF ϕ in S is *range restricted* iff

- If $x \in \text{uVar}(\phi)$ and x is in a positive literal of a clause C (in $head(C)$) in $M(\phi)$, then x must also appear in a negative literal of C (in $body(C)$).
- If $x \in \text{eVar}(\phi)$ and x is in a negative literal of a clause C (in $body(C)$) in $M(\phi)$, then there is a clause D in $M(\phi)$ with only positive literals (no body) such that x appears in every literal in D (in every atom of $head(D)$).

Intuitively, a range restricted formula is structured in such a way that for an interpretation, the range of a variable is restricted to the elements defined by some other relations in the same formula. For example, $\forall x \exists y(q(x) \leftarrow p(x, y))$ is not range restricted, but $\phi = \forall x \exists y(r(y) \wedge (q(x) \leftarrow p(x, y)))$ is range restricted. Let us consider interpretation $I = \{q(a), r(a), p(b, a)\}$. To check if I is a model of ϕ, we need only to consider y where $r(y)$ is true, i.e. $\{y/a\}$.

A search space S for PCL consists of range restricted and function free PCNFs. Moreover, there is an upper bound N for the number of clauses in a PCNF, the number of literals in each clause, etc. The search space is further restricted by some language bias, e.g., the types of the arguments of each predicate must be declared, a declaration is also necessary for each type for which constants must be generated. It is also allowed to specify some types where no constants should be generated. (See Section 8 for some examples of the language bias.)

We can show that the number of steps needed to verify if the given Herbrand interpretation I is a model of some ϕ is exponential w.r.t. N. We sketch the proof here. Let L be the maximal number of literals in a clause of ϕ, M the maximal number of clauses, K the maximal number of arguments in a predicate, F the maximal number of constants allowed in I, and G the maximal number of ground literals which are true in I. Then we can say:

1. The range restriction means that only values occuring in I need to be tested for a variable. There are at most F possibliities for a variable. For a literal there are at most F^K ground instances. There are at most LM literals in ϕ. The total number of possible ground literals in a clause is therefore bounded by LMF^K.
2. Every ground instance of a literal in ϕ should be compared with the truth values of literals in the interpretation I. There are at most G true ones in I. Thus we should have at most $GLMF^K$ comparisons.
3. Suppose the maximum of all these constants K, G, M, etc. is N. We can say now that PCL has an exponential time complexity w.r.t. N.

7.2 The downward refinement operator

For the implementation of the refinement operator ρ, several issues must be addressed.

First, for efficiency it is necessary to optimize the refinement operator. We should try to avoid deriving several equivalent PCNF (and then refine these PCNF, doing the same work several times). One way to do this is to define an order in which literals have to occur in each clause and an order of the clauses. This removes equivalencies obtained by applying assiociativity and commutativity rules. For Instance, we can obtain $false \leftarrow p \wedge q$ in two ways. We can start from the empty clause $false$, first add p to obtain $false \leftarrow p$ and then add q to obtain $false \leftarrow p \wedge q$. We can also first add q to obtain $false \leftarrow q$ and then add p to obtain $false \leftarrow q \wedge p$. The latter is not allowed because p comes alphabetically before q. Such orders are also considered in Claudien using the DLAB language bias.

Second, we need to address the following problem. Let M be the maximal number of clauses in a PCNF in the search space S. Then S can be divided into M subsets $S_1, S_2, \ldots, S_M$ such that every PCNF in S_i contains i clauses. To search S_i, we start from the conjunction of i empty clauses. Thus we have to search M trees. The computational cost of searching in the i-th tree grows very fast as i increases. We can solve this problem partially by reusing the formulas we have found in S_{i-1} to speed up the search in S_i. If $q_1 x_1 ... q_k x_k C_1 \wedge .. \wedge C_{i-1} \wedge C_i$ is a PCNF with i clauses which is true for the given examples, then $q_1 x_1 ... q_k x_k C_1 \wedge .. \wedge C_{i-1}$ must also be true in these examples. To find a good PCNF with i clauses, we can start from good PCNF containg $i-1$ clauses with an extra empty clause added. The refinement operator can be applied again and again until good PCNFs are found for all S_i.

8 Experiments

In this section we present some experiments to illustrate that PCL can learn rules which can not be learned by existing systems that only learn universal quantified clauses.

Experiment 1 We considered a set of undirected graphs as positive examples. One of the examples contains the following true ground atoms:

$r(a,b)$. $r(b,a)$. $r(a,c)$. $r(c,a)$.
$r(a,d)$. $r(d,a)$. $r(e,a)$. $r(a,e)$.

Every graph has the property that each point is connected to at least one other point in the same graph. PCL found this and also some other rules:

$\forall x \forall y (r(x,y) \leftarrow r(y,x))$,
$\forall x \exists y (r(x,y) \leftarrow \text{point}(x))$

Note that the last PCNF is made range restricted by adding a predicate *point.* PCL automatically makes PCNF formulas range restricted by using the language bias and the definitions of the domain predicates. The following language bias are used:

$\text{type}(r(\text{point\_type}, \text{point\_type}))$.
$\text{domain}(\text{point\_type}, X, \text{point}(X))$.

The first declaration means that r is a predicate with two arguments which are of the type *point_type*. The *domain* declaration says that if a variable X of type point_type is not range-restricted, it can be made range-restricted by adding a literal *point*(X) in the formula.

Experiment 2 We also did an experiment on some *bongard-like* examples mentioned at the beginning of last section. These are scenes of figures (in this case triangles and circles) which are related to each another. One example is

$\text{figure}(a)$. $\text{figure}(b)$. $\text{figure}(c)$. $\text{figure}(d)$,
$\text{in}(a,b)$. $\text{in}(c,b)$,
$\text{shape}(a, \text{triangle})$. $\text{shape}(c, \text{triangle})$. $\text{shape}(b, \text{circle})$. $\text{shape}(d, \text{circle})$.

The search space is large and many correct formulas are found. Even more than the case of Claudien, many trivial formulas are given by PCL which do not add new knowledge:

$\exists x(\text{shape}(x, \text{triangle}))$, $\exists x \exists y(\text{in}(x,y))$,
$\exists y(\text{figure}(y) \wedge (false \leftarrow \text{shape}(y, \text{triangle})))$.

After the search has continued for some time, more interesting results are given, such as: $\forall x \exists y(\text{shape}(y, \text{circle}) \wedge (\text{in}(x,y) \leftarrow \text{shape}(x, \text{triangle}))$.

Finally, we add a remark about the language bias. The following bias has been used in this experiment. The two *constant* declarations mean that *triangle* and *circle* are constants which should be tried for arguments of type *shape_type*. In fact, constants are only generated if such declarations are present. For example, the literal *shape*$(X, triangle)$ occurs in one of the clauses found, but PCL never generates clauses containing a literal such as $figure(a)$ since we did not specify constants for the type *name*.

$\text{type}(\text{shape}(\text{name}, \text{shape\_type}))$.

```
type(in(name, name)).
domain(name, X, figure(X)).
constant(shape_type, triangle).
constant(shape_type, circle).
```

9 Conclusion and Future Work

As we know, every formula is equivalent to a PCNF but not necessarily its Skolem standard form. Until now we consider in ILP almost exclusively formulas which are conjunctions of a finite number of universally quantified clauses, especially Horn clauses. To add expressiveness we should consider PCNF in general.

If we want to extend refinement operators to PCNF, we should first extend substitutions to PCNF. In this article we have defined the substitutions which specialize a given PCNF. Elementary substitutions and adding literals can be used to define a refinement operator ρ which is weakly complete. In section 6 we have also analyse the complexities of ρ in function-free or more general logical languages. Unfortunately the results are not encouraging. We have also considered search spaces with universally quantified PCNF. The total number of elements in a $\rho(\phi)$ for a fixed ϕ is polynomial but the total number of ρ-steps needed to find $\rho^d(\top)$ starting from $\top$ is still exponential w.r.t. the depth d of formulas.

This article lays a theoretical foundation for systems to learn PCNF. In fact, a simple system PCL is already implemented by us. PCL deals with a finite search space of function free and range restricted PCNF. If a set of interpretations are given as positive examples, then PCL finds some PCNF such that these examples are models of these formulas.

Notice that we have not used items 4 and 6 in ρ for the weak completeness. In a set of formulas ordered by some kind of generalization, a refinement operator is *complete* if there is a refinement chain from ψ to ϕ whenever ψ is more general than ϕ. For example, item 4 is needed when we consider $\psi = \exists x p(x, x)$ and $\phi = \exists x \exists y p(x, y)$. We would like to know more about the role of item 4 and 6 in completeness. Furthermore, in the set of PCNF there is no well known proof procedure corresponding to resolutions in universal theories. It is also not clear what is the counterpart of subsume for clauses with existential variables. These are interesting research subjects.

We are not only interested in extending our implementation but also in more theoretical applications. As we have mentioned in the introduction, existential variables are used even in very simple constraints in declarative languages. If we want to find more connections between description logic and first order logic, it seems natural to investigate the connections between constraints and PCNF. For example, T-box and A-box are the most elementary ideas in describing a knowledge base using description logic. We can consider a subset of an A-box as a set of positive examples and a T-box as some background knowledge. It may be possible to extend the T-box to a new T-box such that the new T-box implies the set of positive examples. We would like to do such kind of research.

Acknowledgements. Our thanks go to Ronald de Wolf for his comments. Jan Ramon is supported by the Flemish Institute for the Promotion of Science and Technological Research in Industry (IWT).

References

[CL73] C. L. Chang and R. C. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, San Diego, CA, 1973.

[RD97] L. De Raedt and L. Dehaspe, Clausal discovery *Machine Learning*, 26:99-146, 1997.

[GF96] M. Goncalves and C. Froidevaux, A new formalism to integrate quantification in inductive processes, *Proceedings of ILP96*, S. Muggleton (ed.) Vol. 1314 of LNAI series, 1997, Springer, Berlin.

[Laird 88] P. D. Laird, Learning from Good and Bad Data, Kluwer Academic Publishers, Boston, MA, 1988.

[LN94] P. van der Laag and S. H. Nienhuys-Cheng, Existence and nonexistence of complete refinement operators *Proceedings of ECML94*, Vol. 784 of LNAI series, F. Bergadano and L. De Raedt (eds.). Springer-Verlag, Berlin, 1994.

[LN98] P. van der Laag and S. H. Nienhuys-Cheng, Completeness and properness of refinement operators in inductive logic programming. *Journal of Inductive logic programming*, p201-226, vol. 34, nr. 3, 1998, Elsivier.

[N82] J. M. Nicolas, Logic for Improving Integrity Checking in Relational Data Bases, *Informatica*, 1982, Springer-Verlag.

[NLRR99] S. H. Nienhuys-Cheng, W. V. Laer, J. Ramon, L. De Raedt, *Generalizing Refinement operators to learn Prenex Conjunctive Normal Forms*, to appear in the Proceedings of ILP99, LNAI, Springer-Verlag, June 1999, Bled, Slovenia.

[NW97] S. H. Nienhuys-Cheng and R. de Wolf, *Foundations of Inductive Logic Programming*, LNAI Tutorial 1228, Springer-Verlag, 1997.

[S81] E. Y. Shapiro, *Inductive inference of theories from facts.* Research Report 192, Yale University, 1981.

This article was processed using the LaTeX macro package with LLNCS style

---

[*] ERIM Research Programs:

LIS	Business Processes, Logistics and Information Systems
ORG	Organizing for Performance
MKT	Decision Making in Marketing Management
F&A	Financial Decision Making and Accounting
STR	Strategic Renewal and the Dynamics of Firms, Networks and Industries

An NPV and AC analysis of a stochastic inventory system with joint manufacturing and remanufacturing
Erwin van der Laan
ERS-2000-38-LIS

Generalizing Refinement Operators to Learn Prenex Conjunctive Normal Forms
Shan-Hwei Nienhuys-Cheng, Wim Van Laer, Jan N Ramon & Luc De Raedt
ERS-2000-39-LIS