PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT.

Learning Missing Clauses by Inverse Resolution

Peter Idestam-Almquist*

Department of Computer and Systems Sciences
Stockholm University
Electrum 230, 164 40 Kista, Sweden
pi@dsv.su.se

Abstract

The incomplete theory problem has been of large interest both in explanation based learning and more recently in inductive logic programming. The problem is studied in the context of Horn clause logic, and it is assumed that there is only one clause missing for each positive example given. Previous methods have used either top down or bottom up induction. Both these induction strategies include some undesired restriction on the hypothesis space for the missing clause. To overcome these limitations a method where the different induction strategies are completely integrated is presented. The method involves a novel approach to inverse resolution by using resolution, and it implies some extensions to the framework of inverse resolution which makes it possible to uniquely determine the most specific result of an inverse resolution step.

1 Introduction

Completion of incomplete theories has been of large interest in machine learning, particularly in the area of explanation based learning, for which a complete theory is crucial [Mitchell *et al.* 1986, DeJong and Mooney 1986]. Research on augmenting an incomplete domain has been reported in [Hall 1988, Wirth 1988, Ali 1989]. A new framework for inductive learning was invented by inverting resolution [Muggelton and Buntine 1988]. Papers considering augmentation of incomplete theories in this framework are [Wirth 1989, Rouveirol and Puget 1990, Rouveirol 1990].

We only consider Horn clause logic, which is a subset of first order logic, and we follow the notation in logic programming [Lloyd 1987]. The incomplete theory problem can then be formulated as follows. Let P be a definite program (an incomplete theory) and E a definite program clause which should but does not follow from P ($P \not\models E$). Then find a definite program clause H such that:

(a) $P \cup \{E\} \not\models \overline{\overline{H}}$

(b) $P \cup \{H\} \models E$

H is an ***inductive conclusion*** according to [Genesereth and Nilsson 1987].

Let $E=(A \leftarrow B_1,...,B_n)$. Then by ***top down induction*** we mean any reasoning procedure, to infer an inductive conclusion, that starts from A. By ***bottom up induction*** we mean any inductive reasoning procedure that starts from $B_1,...,B_n$.

Most previous methods use either top down [Hall 1988, Wirth 1988, Ali 1989] or bottom up induction [Sammut and Banerji 1986, Muggelton and Buntine 1988, Rouveirol and Puget 1990]. Both these induction strategies have some undesired restrictions on the hypothesis space of H. In [Wirth 1989] a method that combines top down and bottom up induction is presented, while in this paper a method where they are completely integrated will be described. In the previous methods there are also other undesired restrictions, namely that the input clause E must be fully instantiated [Hall 1988, Wirth 1988, Wirth 1989, Ali 1989, Sammut and Banerji 1986] or a unit clause [Muggelton and Buntine 1988]. Our method works for full Horn clause logic.

Logical entailment is used as a definition of generality. Let E and F be two expressions. Then E is ***more general*** than F, if and only if E logically entails F ($E \models F$). We also say that F is ***more specific*** than E.

In the examples, predicate symbols are denoted by p, q, r, s, t and u. Variables (universally quantified) are denoted by x, y, z and w. Constants are denoted by a, b and c. Skolem functions are denoted by k.

In section 2 the inductive framework of inverse resolution is given. In section 3 some extensions to this framework, which make it possible to determine the most specific inverse resolvent, are described. In section 4 a new

* This research was supported by NUTEK, the Swedish National Board for Industrial and Technical Development.

inverse resolution method is presented, and finally in section 5 related work and contributions is discussed.

2 The Framework of Inverse Resolution

The inductive framework of inverse resolution was first presented in [Muggelton and Buntine 1988]. First, as a background, resolution will be described. Then inverse resolution will be definied, and some problems considering inverse reslution will be pointed out.

2.1 Resolution

A ***substitution*** is a finite set of the form $\{v_1/t_1,...,v_n/t_n\}$, where each v_i is a variable, each t_i is a term distinct from v_i, and the variables $v_1,...,v_n$ are distinct. Each element v_i/t_i is called a ***binding*** for v_i. A substitution is applied by simultaneously replacing each occurence of the variable v_i, in an expression, by the term t_i.

An ***expression*** is either a term, a literal, a clause or a set of clauses. (A fixed ordering of literals in clauses and a fixed ordering of clauses in sets of clauses are assumed.)

Let E be an expression and V be the set of variables occurring in E. A ***renaming substitution*** for E is a substitution $\{x_1/y_1,...,x_n/y_n\}$ such that $y_1,...,y_n$ are distinct variables and $(V-\{x_1,...,x_n\})\cap\{y_1,...,y_n\}=\emptyset$.

Let E and F be expressions. Then E is a ***variant*** of F if there exists a renaming substitution θ such that $E=F\theta$.

A ***unifier*** for two terms or literals t_1 and t_2 is a substitution θ such that $t_1\theta=t_2\theta$.

A unifier θ for t_1 and t_2 is called a ***most general unifier*** (mgu) for t_1 and t_2, if for each unifier θ' of t_1 and t_2 there exists a substitution θ'' such that $\theta'=\theta\theta''$.

Let C and D be two clauses which have no variables in common. Then the clause R is ***resolved*** from C and D, denoted $(C;D)\vdash_R R$, if the following conditions hold:

(a) A is a literal in C and B is a literal in D.
(b) θ is an mgu of A and $\overline{B}$.
(c) R is the clause $((C-\{A\})\cup(D-\{B\}))\theta$.

The clause R is called a ***resolvent*** of C and D.

Since C and D have no variables in common, the mgu θ can uniquely be divided into two disjunctive parts θ_A and θ_B such that $\theta=\theta_A\cup\theta_B$ and $A\theta_A=\overline{B}\theta_B$. Consequently condition (c) can be rewritten as:

(c) R is the clause $(C-\{A\})\theta_A\cup(D-\{B\})\theta_B$, where $\theta=\theta_A\cup\theta_B$ and $A\theta_A=\overline{B}\theta_B$.

Let R_0 be a definite program clause and P a definite program. A ***linear derivation*** from R_0 and P consists of a sequence $R_0,R_1,...$ of definite program clauses and a sequence $C_1,C_2,...$ of variants of definite program clauses in P such that each R_{i+1} is resolved from C_{i+1} and R_i. A linear derivation of R_k from R_0 and P is denoted:
$(R_0;C_1)\vdash_R (R_1;C_2)\vdash_R ...\vdash_R R_k$ or for short
$(R_0;P)\vdash_R^* R_k$.

2.2 Inverse Resolution

A ***place*** within an expression is denoted by an n-tuple and defined recursively as follows. The term, literal or clause at place $<a_1>$ within $f(t_1,...,t_n)$ or $\{t_1,...,t_n\}$ is t_{a_1}. The term or literal at place $<a_1,...,a_m>$ $(m>1)$ within $f(t_1,...,t_n)$ or $\{t_1,...,t_n\}$ is the term or literal at place $<a_2,...,a_m>$ in t_{a_1}.

Let E be an expression. Then for each substitution θ there exists a unique inverse substitution θ^{-1} such that $E\theta\theta^{-1}=E$. Whereas the substitution θ maps variables in E to terms, the inverse substitution θ^{-1} maps terms in $E\theta$ to variables. An ***inverse substitution*** is a finite set of the form $\{(t_1,\{p_{1,1},...,p_{1,m_1}\})/v_1,...,(t_n,\{p_{n,1},...,p_{n,m_n}\})/v_n\}$ where each v_i is a variable distinct from the variables in E, each t_i is a term distinct from v_i, the variables $v_1,...,v_n$ are distinct, each $p_{i,j}$ is a place at which t_i is found within E and the places $p_{1,1},...,p_{n,m_n}$ are distinct. An inverse substitution is applied by replacing all t_i at places $\{p_{i,1},...,p_{i,m_i}\}$ in the expression E by v_i.

Example: If the following inverse substitution $\{(a,\{<1,1,2>,<1,2,1,1>,<2,2,1>\})/x\}$ is applied on the expression $\{(p(a,a)\leftarrow p(f(a)),(q(a)\leftarrow r(a))\}$, the expression $\{(p(a,x)\leftarrow p(f(x)),(q(a)\leftarrow r(x))\}$ is obtained.

Let R, C and D be three clauses. If R can be resolved from C and D, then D can be inverse resolved from R and C. The clause D is ***inverse resolved*** from R and C, denoted $(R;C)\vdash_{IR} D$, if the following conditions hold:

(a) A is a literal in C.
(b) θ_A is a substitution whose variables are variables that occur in A.
(c) $(C-\{A\})\theta_A$ is a subset of R.
(d) Γ is a subset of $(C-\{A\})\theta_A$.
(e) θ_B^{-1} is an inverse substitution whose terms are terms that occur in A.
(f) D is the clause $((R-\Gamma)\cup\{\overline{A}\}\theta_A)\theta_B^{-1}$.

The clause D is called an ***inverse resolvent*** of R and C.

Given R and C there are four sources of indeterminacy for D, namely: A, θ_A, Γ and θ_B^{-1}.

If A is a positive literal then D is ***forwardly inverse resolved***, and if A is a negative literal then D is ***backwardly inverse resolved***.

Example: Suppose we have R=(s(a,z)←q(a),r(b)), C=(p(a,x)←q(a),r(x)) and D=(s(y,z)←p(y,b)). The clause D can be forwardly inversed resolved from R and C, (R;C) $\vdash_{IR}$ D, if A=p(a,x), θ_A={x/b}, Γ=(←q(a),r(b)) and θ_B^{-1}={(a,{<1,1>,<2,1>})/y}. The clause C can be backwardly inverse resolved from R and D, (R;D) $\vdash_{IR}$ C, if A=¬p(y,b), θ_A={y/a}, Γ=(s(a,z)←) and θ_B^{-1}= {(b,{<1,2>,<3,1>})/x}. (It is assumed that the positive literal is first in the ordering of literals in a clause.)

Let D_0 be a definite program clause and P a definite program. An ***inverse linear derivation*** from D_0 and P consists of a sequence $D_0,D_1,...$ of definite program clauses and a sequence $C_1,C_2,...$ of variants of definite program clauses in P such that each D_{i+1} is inverse resolved from C_{i+1} and D_i. An inverse linear derivation of D_k from D_0 and P is denoted:
$(D_0;C_1) \vdash_{IR} (D_1;C_2) \vdash_{IR} ... \vdash_{IR} D_k$ or for short
$(D_0;P) \vdash_{IR}^* D_k$.

A ***backward inverse linear derivation*** is an inverse linear derivation where each D_i is backwardly inverse resolved, and a ***forward inverse linear derivation*** is an inverse linear derivation where each D_i is forwardly inverse resolved.

2.3 Some Problems

Consider the definition of inverse resolved. The substitution θ_A can be divided into two disjunctive parts, θ_{A1} including the variables that occur both in A and (C−{A}), and θ_{A2} including the variables that only occur in A ($\theta_A=\theta_{A1}\cup\theta_{A2}$). Then, to determine an inverse resolvent D, we have to choose A, θ_{A1}, θ_{A2}, Γ and θ_B^{-1}. Only in some special cases there are more than one alternative for A and θ_{A1}.

Example: Let R=(p←q(a),r(b)) and C=(p←q(x),r(x)). Then we have either A=¬q(x) and θ_{A1}={x/b}, or A=¬r(x) and θ_{A1}={x/a}.

For Γ and θ_B^{-1} there are limited numbers of alternatives, but for θ_{A2} there is not. The terms in θ_{A2} can be any possible terms. Consequently, it is hard to choose θ_{A2}. Unfortunately there are examples when the choice of θ_{A2} is crucial.

Example: Let R=(r←q), C_1=(p(x)←q), C_2=(s←p(a)) and D=(r←s). Then there is a linear derivation of R from D and $\{C_1,C_2\}$:
$(D;C_2) \vdash_R ((r←p(a));C_1) \vdash_R (r←q)$.
Consequently, there is an inverse linear derivation of D from R and $\{C_1,C_2\}$:
$(R;C_1) \vdash_{IR} ((r←p(a));C_2) \vdash_R (r←s)$.
In the first inverse resolution step θ_{A2} is chosen as {x/a}. With any other choice of θ_{A2} the inverse linear derivation of D would not have been possible.

If R, C, A and θ_{A1} are given, then it is desirable that a unique most specific inverse resolvent can be determined. Unfortunately, in Horn clause logic, it is not possible due to the substitution θ_{A2}.

Example: Let R=(r←q) and C=(p(x)←q). If we seek the most specific clause D such that (R;C) $\vdash_{IR}$ D, then we let $\Gamma=\varnothing$ and $\theta_B^{-1}=\varnothing$ but what should θ_{A2} be? If we let $\theta_{A2}=\varnothing$, the clause D_1=(r←p(x),q) is obtained. For example the clauses D_2=(r←p(a),q) and D_3=(r←p(b),q) are more specific than D_1, but neither D_2 nor D_3 is more specific than the other. Consequently, there is no unique most specific inverse resolvent.

3 Extended Inverse Resolution

Our inverse resolution method (see section 4) implies some extensions to the framework of inverse resolution. After these extensions the choices of θ_{A2}, Γ and θ_B^{-1} in inverse linear derivations can be postponed, and the most specific inverse resolvent can be determined.

3.1 Existentially Quantified Variables

To postpone the choice of θ_{A2}, existentially quantified variables will temporarily be introduced. Any sentence, in which the existentially quantified variables are replaced by Skolem functions, is equal to the original sentence with respect to satisfiability [Genesereth and Nilsson 1987]. Therefore the existentially quantified variables will be represented by Skolem functions. As a consequence of the introduction of existentially quantified variables (Skolem functions), some additional types of substitutions are needed.

A ***Skolem function*** is a term $f(x_1,...,x_n)$ where f is a new function symbol and $x_1,...,x_n$ are the variables associated with the enclosing universal quantifiers.

A ***Skolem substitution*** is a finite set of the form $\{v_1/k_1,...,v_n/k_n\}$, where each v_i is a variable, each k_i is a Skolem function, and the variables $v_1,...,v_n$ are distinct.

An ***inverse Skolem substitution*** is a finite set of the form $\{k_1/v_1,...,k_n/v_n\}$, where each k_i is a Skolem function, each v_i is a new variable, and the Skolem functions $k_1,...,k_n$ are distinct.

Let $\sigma=\{x_1/k_1,...,x_n/k_n\}$ be a Skolem substitution and $\sigma^{-1}=\{k_1/y_1,...,k_n/y_n\}$ an inverse Skolem substitution such that the Skolem functions in σ and σ^{-1} are exactly the same. Then the ***composition*** $\sigma\sigma^{-1}$ of σ and σ^{-1} is a renaming substitution $\{x_1/y_1,...,x_n/y_n\}$ for any expression E.

An ***existential substitution*** is a finite set of the form $\{k_1/t_1,...,k_n/t_n\}$, where each k_i is a Skolem function (existentially quantified variable), each t_i is a term (possibly a Skolem function) distinct from k_i, and the Skolem functions $k_1,...,k_n$ are distinct. While a substitution or a Skolem substitution corresponds to a specialization an existential substitution corresponds to a generalization.

As an inverse substitution, an inverse existential substitution is specified with respect to an expression E. An ***inverse existential substitution*** is a finite set of the form $\{(t_1,\{p_{1,1},...,p_{1,m_1}\})/k_1,...,(t_n,\{p_{n,1},...,p_{n,m_n}\})/k_n\}$ where each k_i is a Skolem function distinct from the Skolem functions in E, each t_i is a term distinct from k_i, the Skolem functions $k_1,...,k_n$ are distinct, each $p_{i,j}$ is a place at which t_i is found within E and the places $p_{1,1},...,p_{n,m_n}$ are distinct. An inverse existential substitution is applied by replacing all t_i at places $\{p_{i,1},...,p_{i,m_i}\}$ in E by k_i.

Let $\sigma=\{v_1/k_1,...,v_n/k_n\}$ be a Skolem substitution and $\eta=\{k_1/t_1,...,k_n/t_n\}$ an existential substitution such that the Skolem functions in σ and η are exactly the same. Then the ***composition*** $\sigma\eta$ of σ and η is the substitution $\{v_1/t_1,...,v_n/t_n\}$. In this way Skolem substitutions and existential substitutions can be used to postpone the choice of θ_{A2}.

3.2 Most Specific Inverse Resolution

To postpone the choice of Γ, the notion of ***optional literals*** will be used. A clause $\{B_1,...,B_k,B_{k+1},...,B_n\}$, in which the literals $\{B_{k+1},...,B_n\}$ are optional, is denoted $C[c]=\{B_1,...,B_k,[B_{k+1},...,B_n]\}$ where $C=\{B_1,...,B_k\}$ and $c=\{B_{k+1},...,B_n\}$. Consequently, if $c=\emptyset$ then $C[c]=C$.

Example: Let $R=(p\leftarrow q,r,s)$ and $C=(t\leftarrow q,r,s)$ be two clauses. Then $(R;C)\vdash_{IR} D$, where $D=(p\leftarrow t,q,r,s)-\Gamma$ and $\Gamma\subseteq\{\neg q,\neg r,\neg s\}$. All these alternatives for D can be described in a compact way by using optional literals. Thus, $D[d]=(p\leftarrow t,[q,r,s])$.

The definition of inverse resolved can now be modified in such a way that the choices of θ_{A2}, Γ and θ_B^{-1} are postponed. The clause D is ***most specific inverse resolved*** from R[r] (which may include Skolem functions) and C, denoted $(R[r];C)\vdash\downarrow_{IR} D$, if the following conditions hold:

(a) A is a literal in C.
(b) θ_{A1} is a substitution whose variables are variables that occur both in A and $(C-\{A\})$.
(c) η^{-1} is an inverse existential substitution whose terms are terms that occur in $(C-\{A\})$.
(d) $(C-\{A\})\theta_{A1}\eta^{-1}$ is a subset of R[r].
(e) σ is a Skolem substitution whose variables are all the variables that only occur in A.
(f) D[d] is the clause $D=((R-\Gamma)\cup\{\bar{A}\}\theta_{A1}\sigma)$, $d=r\cup\Gamma$, where $\Gamma=(C-\{A\})\theta_{A1}\eta^{-1}$.

The clause $D\cup d$ is called a ***most specific inverse resolvent*** of R and C.

Given R and C, there are only two sources of indeterminacy, namely: A and θ_{A1}. Consequently, given R, C, A and θ_{A1} there is a unique most specific inverse resolvent $D\cup d$.

Example: Let $R=(r\leftarrow q)$ and $C=(p(x)\leftarrow q)$. Then the unique most specific inverse resolvent of R and C is the clause $D\cup d=(r\leftarrow p(k),q)$ where k is a Skolem functions (representing an existentially quantified variable). This is true, since $\forall x(r\leftarrow p(x),q)\models(r\leftarrow p(t),q)$, and $(r\leftarrow p(t),q)\models\exists x(r\leftarrow p(x),q)$ for any term t.

Let $D_0[d_0]$ a be definite program clause and P a definite program. A ***most specific inverse linear derivation*** from $D_0[d_0]$ and P consists of a sequence $D_0[d_0],D_1[d_1],...$ of definite program clauses and a sequence $C_1,C_2,...$ of variants of definite program clauses in P such that each $D_{i+1}[d_{i+1}]$ is most specific inverse resolved from C_{i+1} and $D_i[d_i]$. A most specific inverse linear derivation of $D_k[d_k]$ from $D_0[d_0]$ and P is denoted:
$(D_0[d_0];C_1)\vdash\downarrow_{IR}(D_1[d_1];C_2)\vdash\downarrow_{IR}...\vdash\downarrow_{IR} D_k[d_k]$
or for short $(D_0[d_0];P)\vdash\downarrow_{IR}^{*} D_k[d_k]$.

Each result of an inverse linear derivation can be obtained from the result of some most specific inverse linear derivation, if we apply an inverse substitution, an existential substitution, and drop a subset of the optional literals.

Example: Suppose we have the following clauses $R=(r\leftarrow q)$, $C_1=(p(x)\leftarrow q)$, $C_2=(s\leftarrow p(a))$, $C_3=(t(b)\leftarrow p(b))$, $D=(r\leftarrow s,t(x),p(c))$ and $D'[d']=(r\leftarrow s,t(b),[p(k),q])$. Then
$(R;\{C_1,C_2,C_3\}) -_{IR}{}^* D$ and
$(R;\{C_1,C_2,C_3\}) \vdash\downarrow_{IR}{}^* D'[d']$.
The clause D can be obtained from D'[d'] by application of the inverse substitution $\{(b,\{<3,1>\})/x\}$ and the existential substitution $\{k/c\}$, and by dropping the optional literal q.
The most specific inverse linear derivation of D'[d'] looks as follows:
$(R;C_1) \vdash\downarrow_{IR} ((r\leftarrow p(k),[q]);C_2) \vdash\downarrow_{IR}$
$((r\leftarrow s,[p(k),q]);C_3) \vdash\downarrow_{IR} (r\leftarrow s,t(b),[p(k),q])$.
That η^{-1}, in the two last steps are $\{(a,<1,1>)/k\}$ and $\{(b,<1,1>)/k\}$, and that k then can be replaced by a third term c, may seem inconsistent, but it is not. Consider the corresponding inverse linear derivation of D from R and $\{C_1,C_2,C_3\}$:
$((r\leftarrow q);C_1) \vdash_{IR} ((r\leftarrow q,p(c));C_1) \vdash_{IR}$
$((r\leftarrow q,p(b),p(c));C_1) \vdash_{IR} ((r\leftarrow p(a),p(b),p(c));C_2) \vdash_{IR}$
$((r\leftarrow s,p(b),p(c)),C_3) \vdash_{IR} (r\leftarrow s,t(x),p(c))$.
Note that since k has been used as three different terms (a, b and c) in the most specific inverse linear derivation, three inverse resolution steps are needed to compensate for the step where k is introduced. Note also that $\theta_{A2}=\{x/c\}$ in the first, $\theta_{A2}=\{x/b\}$ in the second and $\theta_{A2}=\{x/a\}$ in the third inverse resolution step. To choose exactly those substitutions is hard, but in a most specific inverse linear derivation it is not necessary.

3.3 Truncation Generalization

A clause C_1 ***θη-subsumes*** a clause C_2 if there exists a substitution θ and an existential substitution η such that $C_1\theta\subseteq C_2\eta$. If C_1 θη-subsumes C_2 then $C_1 \models C_2$.

To perform a ***θη-truncation*** is to apply some arbitrary existential substitution η, apply some arbitrary inverse substitution θ^{-1}, and drop some arbitrary literals. The generalizarion technique θη-truncation corresponds to θη-subsumption.

Let P be a definite program (an incomplete theory) and E a definite program clause which should but does not follow from P ($P \not\models E$), let $\mathbb{D}$ be the set of definite program clauses D such that $(E;P) \vdash_{IR}{}^* D$, and let $\mathbb{H}$ be the set of definite program clauses H such that $P\cup\{H\} \models E$. Since resolution is not complete [Rob65] $\mathbb{D}$ is a subset of $\mathbb{H}$ ($\mathbb{D} \subseteq \mathbb{H}$). In particular each definite program clause D' that θη-subsumes some clause D, where $D \in \mathbb{D}$, will be in $\mathbb{H}$. This is true since $P\cup\{D\} \models E$, and $D' \models D$, gives us $P\cup\{D'\} \models E$. Consequently, we can perform any θη-truncation on the result D of a most specific inverse linear derivation and still have an inductive conclusion.

4 The Method

In this section a method, which in an easy way realizes inverse linear derivations, will be described. Instead of performing an inverse linear derivation from the example clause E, a variant of ordinary resolution derivation is performed from the complement $\overline{E}$ of E.

4.1 Complement

A ***definite program clause complement set*** (dpcc-set) is set of clauses containing exactly one unit goal and a number of unit clauses.

Let C be a definite program clause $(A\leftarrow B_1,...,B_n)$, $\sigma_S{}^{-1}$ an inverse Skolem substitution including all Skolem functions in C, and σ_S a Skolem substitution including all the universally quantified variables in C. Then the ***complement*** $\overline{C}$ of C is the definite program clause complement set $\{(\leftarrow A),(B_1\leftarrow),...,(B_n\leftarrow)\}\sigma_S{}^{-1}\sigma_C$. Let S be a dpcc-set $\{(\leftarrow A),(B_1\leftarrow),...,(B_n\leftarrow)\}$, $\sigma_C{}^{-1}$ an inverse Skolem substitution including all Skolem functions in S, and σ_S a Skolem substitution including all the universally quantified variables in S. Then the complement $\overline{S}$ of S is the definite program clause $(A\leftarrow B_1,...,B_n)\sigma_C{}^{-1}\sigma_S$. Thus, the complement of a dpcc-set is a definite program clause and vice versa.

Example: Let C be the clause $(p(a,x)\leftarrow q(k,x,y))$. Then the complement $\overline{C}$ of C is the definite program clause complement set $\{(\leftarrow p(a,k_x)),(q(x_k,k_x,k_y)\leftarrow)\}$, which is obtained by application of the inverse Skolem substitution $\{k/x_k\}$ and the Skolem substitution $\{x/k_x,y/k_y\}$ on the set of clauses $\{(\leftarrow p(a,x)),(q(k,x,y)\leftarrow)\}$. The complement C' of $\overline{C}$ is the definite program clause $(p(a,x')\leftarrow q(k',x',y'))$, which is obtained by application of the inverse Skolem substitution $\{k_x/x',k_y/y'\}$ and the Skolem substitution

$\{x_k/k'\}$ on the clause $(p(a,k_x)\leftarrow q(x_k,k_x,k_y))$. The clause C' is a variant of C, since $C'=C\theta$ where θ is the renaming substitution $\{x/x',y/y'\}$.

4.2 Clause Set Resolution

The notion of ***optional clauses*** will be used a similar same way as optional literals. A set of clauses $\{C_1,\ldots,C_k,C_{k+1},\ldots,C_n\}$, in which the clauses $\{C_{k+1},\ldots,C_n\}$ are optional, is denoted $S[s]=\{C_1,\ldots,C_k,[C_{k+1},\ldots,C_n]\}$ where $S=\{C_1,\ldots,C_k\}$ and $s=\{C_{k+1},\ldots,C_n\}$. Consequently, if $s=\varnothing$ then $S[s]=S$.

An ***elementary clause set*** Σ is a set of clauses containing at most one clause, that is $\Sigma=\varnothing$ or $\Sigma=\{C\}$ where C is a clause.

Let $S_i[s_i]$ be a clause set and Σ an elementary clause set. Then $S_{i+1}[s_{i+1}]$ is ***clause set resolved*** from $S_i[s_i]$ and Σ, denoted $(S_i[s_i];\Sigma) \vdash_{CSR} S_{i+1}[s_{i+1}]$, if the following conditions hold:

(a) C' is a variant of a clause C in $S_i[s_i]\cup\Sigma$.
(b) D is a clause in $S_i[s_i]$.
(c) R is a resolvent of C' and D.
(d) Δ is the elementary clause set of unit clauses in {C,D}.
(e) $S_{i+1}[s_{i+1}]$ is the clause set $S_{i+1}=(S_i-\{C,D\})\cup\{R\}$, $s_{i+1}=s_i\cup\Delta$.

If D is a definite goal then R will also be a definite goal, and we say that $S_{i+1}[s_{i+1}]$ is ***backwardly clause set resolved*** from $S_i[s_i]$ and Σ. If both C and D are definite program clauses then R will also be a definite program clause, and we say that $S_{i+1}[s_{i+1}]$ is ***forwardly clause set resolved*** from $S_i[s_i]$ and Σ.

Let $S_0[s_0]$ be a clause set and P a definite program. A ***clause set derivation*** from $S_0[s_0]$ and P consists of a sequence $S_0[s_0],S_1[s_1],\ldots$ of clause sets, and a sequence $\Sigma_1,\Sigma_2,\ldots$ of elementary clause sets, such that each Σ_i is a subset of P and each clause set $S_{i+1}[s_{i+1}]$ is clause set resolved from $S_i[s_i]$ and Σ_{i+1}. A clause set derivation of $S_k[s_k]$ from $S_0[s_0]$ and P is denoted:
$(S_0[s_0];\Sigma_1) \vdash_{CSR} (S_1[s_1];\Sigma_2) \vdash_{CSR} \ldots \vdash_{CSR} S_k[s_k]$ or
$(S_0[s_0];P) \vdash_{CSR}^* S_k[s_k]$.

A ***backward clause set derivation*** is a clause set derivation where each $S_i[s_i]$ is backwardly clause set resolved, and a ***forward clause set derivation*** is a clause set derivation where each $S_i[s_i]$ is forwardly clause set resolved.

Example: Let $S_0=\{(\leftarrow p(k)),(q(k)\leftarrow),(r(k)\leftarrow)\}$ and $C=(p(x)\leftarrow r(x),s(x))$. Then we have the following backward clause set derivation:
$(S_0;\{C\}) \vdash_{CSR} (\{(\leftarrow r(k),s(k)),(q(k)\leftarrow),(r(k)\leftarrow)\};\varnothing) \vdash_{CSR} \{(\leftarrow s(k)),(q(k)\leftarrow),[(r(k)\leftarrow)]\}$.

4.3 The Algorithm

Let P be a definite program and E a definite program clause which should but does not follow from P ($P \not\models E$). Our algorithm to produce an inductive conclusion H looks as follows.

Completion of Refutation Proof Algorithm:

1. Compute the complement $\overline{E}$ of E, which is a dpcc-set.
2. Perform a clause set derivation from P and $\overline{E}$ of a dpcc-set $\overline{H'[h']}$.
3. Compute the complement H'[h'] of $\overline{H'[h']}$, which is a definite program clause.
4. Perform a $\theta\eta$-truncation of H'[h'] to obtain H.

The generalization performed in steps 1-3, is called a ***reformulation generalization***, which in fact is equivalent to performing a most specific inverse linear derivation.

Reconsider the definition of most specific inverse linear resolved in section 2. Let $\{A_1,\ldots,A_m\}=C-\{A\}$ and $\{B_1,\ldots,B_n\}=R-\{A_1,\ldots,A_m\}\theta_{A1}\eta^{-1}$. Then the clause $D[d]=\{B_1,\ldots,B_n\}\cup\{\overline{A}\}\theta_{A1}\sigma_{S2}\cup[\{A_1,\ldots,A_m\}\theta_{A1}\eta^{-1}]$ is most specific inverse resolved from
$R=\{A_1,\ldots,A_m\}\theta_{A1}\eta^{-1}\cup\{B_1,\ldots,B_n\}$ and
$C=\{A\}\cup\{A_1,\ldots,A_m\}$.

The corresponding reformulation generalization looks as follows:

1. The complement $\overline{R}$ of R is the dpcc-set
$(\{\{\overline{A}_1\},\ldots,\{\overline{A}_m\}\}\theta_{A1}\eta^{-1}\cup\{\{\overline{B}_1\},\ldots,\{\overline{B}_n\}\})\sigma_{S1}^{-1}\sigma_R$
where σ_{S1}^{-1} is an inverse Skolem substitution including all Skolem functions in R and σ_R is a Skolem substitution including all universally quantified variables in R.
2. The following clause set derivation is performed:
$(\overline{R};\{C\}) \vdash_{CSR}^* \overline{D[d]}$ where
$\overline{D}=(\{\{\overline{B}_1\},\ldots,\{\overline{B}_n\}\}\cup\{\{A\}\theta_{A1}\}$ and
$\overline{d}=[\{\{\overline{A}_1\},\ldots,\{\overline{A}_m\}\}\theta_{A1}\eta^{-1}])\sigma_{S1}^{-1}\sigma_R$.
3. The complement D[d] of $\overline{D[d]}$ is the definite program clause
$(\{B_1,\ldots,B_n\}\cup\{\overline{A}\}\theta_{A1}\sigma_{S2}\cup[\{A_1,\ldots,A_m\}\theta_{A1}\eta^{-1}])\theta$
where σ_{S2} is a Skolem substitution including all universally quantified variables in $\overline{D[d]}\sigma_{S1}$ and θ is the renaming substitution $\theta=\sigma_{S1}^{-1}\sigma_R\sigma_R^{-1}\sigma_{S1}$.

Consequently, a most specific inverse linear derivation
$(D_0[d_0];P) \vdash_{\downarrow IR}^* D_k[d_k]$
is equivalent to the clause set derivation
$(S_0[s_0];P) \vdash_{CSR}^* S_k[s_k]$,
where $S_0[s_0] = \overline{D_0[d_0]}$, $D_k[d_k] = \overline{S_k[s_k]}\theta$ and θ is a renaming substitution.

Example: Let $R=(r(a,z)\leftarrow q(z))$, $C_1=(p(x,y)\leftarrow q(y))$, $C_2=(s(w,y)\leftarrow p(b,y))$, $D_1=(r(x,z)\leftarrow s(c,z))$, $D_2=(r(x,z)\leftarrow)$ and $D[d]=(r(a,z)\leftarrow s(k_w,z),[p(k_x,z),q(z)])$ Then
$(R;\{C_1,C_2\}) \vdash_{\downarrow IR}^* D[d]$, and
$(R;\{C_1,C_2\}) \vdash_{IR}^* D_1$.
Although D_2 is not inverse linear derivable, it is still an inductive conclusion, since $\{C_1,C_2,D_2\} \models R$.
With our algorithm D'[d'], D_1' and D_2', which are equal to D[d], D_1 and D_2 up to variable renaming, are constructed in the following way:
1. The complement $\overline{R}$ of R is the dpcc-set
$\{(\leftarrow r(a,k_z)),(q(k_z)\leftarrow)\}$.
2. The following clause set derivation is performed:
$(\overline{R};\{C_1\}) \vdash_{CSR}$
$(\{(\leftarrow r(a,k_z)),(p(x,k_y)\leftarrow),[(q(k_z)\leftarrow)]\};\{C_2\}) \vdash_{CSR} \overline{D[d]}$,
where
$\overline{D[d]}=\{(\leftarrow r(a,k_z)),(s(w,k_z)\leftarrow),[(p(x,k_y)\leftarrow),(q(k_z)\leftarrow)]\}$.
3. The complement D'[d'] of $\overline{D[d]}$ is the definite program clause
$(r(a,z')\leftarrow s(k_w,z'),[p(k_x,z'),q(z')])$.
4. By application of the inverse substitution $\{(a,\{<1,1>\})/x\}$ and the existential substitution $\{k_w/c\}$ and by dropping the optional literals, $D_1'=(r(x,z')\leftarrow s(c,z'))$ is obtained.
If the last negative literal in D_1' also is dropped then $D_2'=(r(x,z')\leftarrow)$ is obtained.

Steps 2 and 4 in the completion of refutation proof algorithm are indeterministic. The use of a preference bias can make them deterministic. Such a preference bias must specify which clause set is the most preferable result of the clause set derivation (reformulation bias), and which generalization should be done in the $\theta\eta$-truncation (truncation bias).

The algorithm is implemented in a system, called CRP1, in which a depth first search is used to find the best dpcc-set $\overline{H'[h']}$ according to some given preference bias.

4.4 Integrating Top down and Bottom up Induction

Backward inverse linear derivations correspond to top down induction, and forward inverse linear derivations correspond to bottom up induction. In our method, backward clause set derivations correspond to top down induction, and forward clause set derivations correspond to bottom up induction. Each step in a clause set derivation can be either backwardly or forwardly clause set resolved. Consequently, in our method (and in the system CRP1) top down and bottom up induction are completely integrated.

Example: Let $E=(p\leftarrow q,t,u)$ and $P=\{(p\leftarrow q,r),(s\leftarrow t,u)\}$. Then the inductive conclusion $H_1=(r\leftarrow t,u)$ is inferable by top down induction (backward inverse linear derivation), but not by bottom up induction (forward inverse linear derivation). The inductive conclusion $H_2=(p\leftarrow q,s)$ is inferable by bottom up induction, but not by top down induction. The inductive conclusion $H_3=(r\leftarrow s)$ can only be inferred by a method that combines top down and bottom up induction. With our algorithm the clause H_3 is constructed as follows:
1. The complement $\overline{E}$ of E is the dpcc-set $\{(\leftarrow p),(q\leftarrow),(t\leftarrow),(u\leftarrow)\}$.
2. The following clause set derivation is performed:
$(\overline{E};\{(p\leftarrow q,r)\}) \vdash_{CSR}$
$(\{(\leftarrow q,r),(q\leftarrow),(t\leftarrow),(u\leftarrow)\};\varnothing) \vdash_{CSR}$
$(\{(\leftarrow r),(t\leftarrow),(u\leftarrow),[(q\leftarrow)]\};\{(s\leftarrow t,u)\}) \vdash_{CSR}$
$(\{(\leftarrow r),(s\leftarrow u),(u\leftarrow),[(t\leftarrow),(q\leftarrow)]\};\varnothing) \vdash_{CSR} \overline{H_3[h_3]}$
where $\overline{H_3[h_3]}=\{(\leftarrow r),(s\leftarrow),[(u\leftarrow),(t\leftarrow),(q\leftarrow)]\}$
3. The complement $H_3[h_3]$ of $\overline{H_3[h_3]}$ is the definite program clause $(r\leftarrow s,[u,t,q])$.

4. By dropping the optional literals $H_3=(r\leftarrow s)$ is obtained.
The first two steps in the clause set derivation are backwardly clause set resolved (top down induction) and the last two steps are forwardly clause set resolved (bottom up induction).

5 Concluding Remarks

Some extensions to the inverse resolution framework and a new inverse resolution method have been presented. This method subsumes the previous methods based on inverse resolution and completely integrates top down and bottom up induction.

Reconsider the definition of inverse resolved in section 2. If we let A be a positive literal, $\theta_{A2}=\varnothing$ and $\Gamma=(C-\{A\})\theta_A$ then it is a definition of the absorption operator [Muggelton and Buntine 1988]. If we let A be a positive literal, $\theta_{A2}=\varnothing$, $\Gamma=\varnothing$ and $\theta_B{}^{-1}=\varnothing$ then it is a definition of elementary saturation [Rouveirol and Puget 1990]. The saturation operator [Rouveirol and Puget 1990] is equal to an exhaustive forward inverse linear derivation, in which each step is restricted according to elementary saturation. If we let A be a positive literal, $\theta_{A2}=\varnothing$, $\Gamma=(C-\{A\})\theta_A$ and $\theta_B{}^{-1}=\varnothing$ then it is a definition of the learning procedure called generalize in [Banerji 1991]. If we let A be a negative literal, $\theta_{A2}=\varnothing$ and $\Gamma=(C-\{A\})\theta_A$ then it is a definition of the identification operator [Muggelton and Buntine 1988].

Since our method performs inverse linear derivations without any restrictions on A, θ_{A2}, Γ or $\theta_B{}^{-1}$, all the methods mentioned above can be seen as special cases of our method.

Our notion of optional literals is the same as in [Rouveirol and Puget 1990]. Our $\theta\eta$-truncation is similar to the truncation generalization in [Rouveirol and Puget 1990] and the truncation operator in [Muggelton and Buntine 1988], which both correspond to θ-subsumption.

Wirth [Wirth 1989] and Rouveirol [Rouveirol 1991] have both pointed out the advantages of combining top down and bottom up induction. In [Wirth 1989], a system called LFP2, which uses both top down and bottom up induction is presented. However, the different induction strategies are separated into different parts of the system. The first part (top down) is based on completion of partial proof trees, while the second part (bottom up) is based on operators performing inverse resolution. The second part uses the result from the first part, and different types of bias are used in the different parts. Our method has the major advantage that the two different induction strategies are completely integrated, which not only eliminates the restrictions that they imply when separated, but also makes possible the use of an overall preference bias.

The main contributions of this research are:

1. A complete integration of top down and bottom up induction.
2. Introduction of existentially quantified variables, which makes it possible to uniquely determine the most specific inverse resolvent.
3. A method to perform inverse resolution for full Horn clause logic by using resolution.

References

[Ali 1989] K. M. Ali, "Augmenting Domain Theory for Explanation Based Generalization" in *Proceedings of the 6th International Workshop on Machine Learning*, Morgan Kaufmann, 1989.

[Banerji 1991] R. B. Banerji, "Learning Theoretical Terms" in *Proceedings of International Workshop on Inductive Logic Programming*, 1991.

[DeJong and Mooney 1986] G. DeJong and Mooney, "Explanation-Based Learning: An Alternative View" in *Machine Learning* 1: 145-176, 1986.

[Genesereth and Nilsson 1987] Nilsson and Genesereth, *Logic Foundations of Artificial Intelligence*, Morgan Kaufmann, 1987.

[Hall 1988] R. J. Hall, "Learning by Failing to Explain: Using Partial Explanations to Learn in Incomplete or Intractable Domains" in *Machine Learning* 3: 45-77, 1988.

[Lloyd 1987] J. W. Lloyd, *Foundations of Logic Programming* (second edition), Springer-Verlag, 1987.

[Mitchell *et al.* 1986] T. M. Mitchell, S. Kedar-Cabelli and R. Keller, "Explanation-Based Generalization: A Unifying View" in *Machine Learning* 1: 47-80, 1986.

[Muggleton and Buntine 1988] S. Muggleton and W. Buntine, "Machine Invention of First-order Predicates by Inverting Resolution" in *Proceedings of the 5th International Conference on Machine Learning*, Morgan Kaufmann, 1988.

[Robinson 1965] J. Robinson, "A Machine-oriented Logic Based on the Resolution Principle" in *Journal of ACM* 12(1), 1965.

[Rouveirol 1990] C. Rouveirol, "Saturation: Postponing Choices when Inverting Resolution" in *Proceedings of the 9th European Conference on Artificial Intelligence*, Pitman, 1990.

[Rouveirol 1991] Céline Rouveirol, "ITOU: Induction of First Order Theories" in *Proceedings of International Workshop on Inductive Logic Programming*, 1991.

[Rouveirol and Puget 1990] C. Rouveirol and J. F. Puget, "Beyond Inversion of Resolution" in *Proceedings of the 7th International Conference on Machine Learning*, Morgan Kaufmann, 1990.

[Sammut and Banerji 1986] C. Sammut and R. Banerji, "Learning concepts by asking questions" in Michalski, Carbonell and Mitchell (eds), *Machine Learning: an artificial intelligence approach* volume 2, Morgan Kaufmann, 1986.

[Wirth 1988] R. Wirth, "Learning by Failure to Prove" in *Proceedings of the 3rd European Working Session on Learning*, Pitman, 1988.

[Wirth 1989] R. Wirth, "Completing Logic Programs by Inverse of Resolution" in *Proceedings of the 4th European Working Session on Learning*, Pitman, 1989.