CONSISTENT TERM MAPPINGS, TERM PARTITIONS, AND INVERSE RESOLUTION

SHAN HWEI NIENHUYS-CHENG (*) PETER A. FLACH (**)
(*) Erasmus University, POBox 1738, 3000 DR Rotterdam, Netherlands
(**) Tilburg University, POBox 90153, 5000 LE Tilburg, Netherlands. Email: flach@kub.nl.

Abstract

We formalize the notion of inverse substitution, used in the context of inverse resolution, by means of consistent term mappings. An inverse substitution from a clause to a more general clause can also be characterized by means of a term partition. We can generate clauses more general than a given clause by taking an *admissible* subset of its term occurrences, and constructing a term partition of this subset. We show that these term partitions can be partially ordered. This ordering coincides with the generality of the induced clauses. Similar partitions have been used by Muggleton and Buntine for describing their absorption operator. We show that their absorption algorithm is incomplete, and we give an alternative, complete algorithm, based on our definitions of admissible subset and term partition. We show that under certain conditions, clauses generated by absorption are incomparable with respect to generality. Finally, we relate this to a recent result about least general absorption obtained by Muggleton.

Keywords Inverse resolution, absorption, substitution.

1. Introduction

Muggleton and Buntine (1988) introduced inverse resolution in machine learning. Operators like absorption and intra-construction are used to generalize given first-order clauses, and to invent new predicates. They implement the absorption operator in a system GIGOL such that for given clause C and positive literal C_1, C_2 can be found as the resolvent of C_1 and C_2. They present also an algorithm, which we call MB-absorption, to find C_2 non-deterministically. They consider a subset TP' of the set of all term occurrences in $C \vee {\sim}C_1$. This subset is partitioned in blocks. Every block looks like

$$B=\{(r,p_1),\ldots,(r,p_n)\}\cup\{(s,q_1),\ldots,(s,q_m)\}$$

where (r,p_i) is a term occurrence in C and (s,q_j) is a term occurrence of ${\sim}C_1$. Furthermore, there is a substitution θ_1 from ${\sim}C_1$ which brings s to r. Every such block corresponds to a new variable in C_2. That means, all the terms (r,p_i) and (s,q_j) are changed to a new variable. To understand what this algorithm can do or cannot do, we give a few examples.

(*i*) Let $C_1=P(x)$, $C=Q(v,g(v))$ and $C_2=Q(v,g(v))\vee{\sim}P(h(v))$. Although C is the resolvent of C_1 and C_2, yet we cannot find C_2 with MB-absorption. A block which contains the term occurrence of x from C_1 has to correspond to a variable. It cannot be changed to h(v). Hence, MB-absorption is *incomplete*: it does not find all C_2 such that C is the resolvent of C_1 and C_2.

(*ii*) Another example of the incompleteness of MB-absorption: let $C_1=P(x,y)$, $C=Q(u,f(w))$ and $C_2=Q(u,f(w))\vee{\sim}P(u,u)$. It is clear that C is the resolvent of C_1 and C_2. On the other hand, we cannot find C_2 with MB-absorption. In that algorithm, the term occurrences (s,q_j) from C_1 in a block have to be the same term s. Here x, y from C_1 are different but they have to be in the same block in order to go to the same variable u in C_2.

(*iii*) Let $C=Q(f(a),f(b))$, $C_1=P(f(u),f(v))$, then $C_2=Q(x,y)\vee{\sim}P(x,y)$ can be constructed by MB-absorption if we define $\theta_1=\{u/a, v/b\}$. The resolvent of C_1 and C_2 is $Q(f(u),f(v))$, which is more general than C. Thus, MB-absorption is also *unsound*: It constructs C_2 such that C is not the resolvent of C_1 and C_2.

Besides the incompleteness and unsoundness of the MB-algorithm there are still the following questions:

- What kind of subsets of terms are used for partitions and which partitions are allowed for a given subset?
- Different partitions induce different clauses. Is it possible to see that one induced clause is more general than another, just by comparing the associated partitions?

To improve the MB-algorithm and to answer the two questions above we need a formal basis so as to discuss problems and prove theorems easily and more precisely. To this end, we introduce consistent term mappings in section 2. A consistent term mapping is defined on a subset of all term occurrences in a clause. A term occurrence is identified not only by the term but also by the position where this term occurs. In fact the position determines the term in the clause and we can use positions to prove several properties and theorems. A consistent term mapping has the effect of replacing a term occurrence in a clause by new ones and thus induces a new clause. We can also formulate substitutions and inverses of substitutions as special consistent term mappings and thus we have generalized these two concepts. The generalizations go beyond these aspects. For example, a substitution is defined on variables, now we can consider a consistent mapping which coincides with this substitution in variables but has a different domain. The flexibility of domains makes many mathematical formulations and proofs possible and easier. Consistent term mappings have been introduced (under the name consequent functions) in a report by the first author (Nienhuys-Cheng, 1990), which examines in more detail the properties of consistent term mappings in general.

In section 3 we consider partitions which are defined on some subsets of term occurrences in a clause. Such a partition induces a more general clause by constructing an inverse substitution with respect to this partition. We can compare two partitions by an order relation. This order relation between partitions coincides with the generality relation between the induced clauses. The advantages of comparing term partitions instead of clauses is that we do not have to construct the induced clauses and the substitutions explicitly.

In section 4 we apply the theory in the first two sections to absorption. The problem with MB-absorption is that they consider $C \vee \sim C_1$ when they want to construct C_2 and they distinguish the term occurrences from C and from C_1. This approach is not general enough to construct all C_2's. Our approach considers first a fixed substitution θ_1 and then $C \vee \sim C_1\theta_1$ as a whole, thus we can apply the theory of section 3 about partitions which are based on one clause (i.e. $C \vee \sim C_1\theta_1$) without taking into account which term occurrences comes from C and which ones comes from C_1. Thus we establish a new algorithm. If we let θ_1 change, then we have all possible C_2's.

However, if we do consider $C \vee \sim C_1\theta_1$ as combination of C and $\sim C_1\theta_1$, we can compare C_2's with respect to different θ_1's by using C as a bridge. If C_2 is induced on the basis of $C \vee \sim C_1\theta_1$ and C_2' is induced on the basis of $C \vee \sim C_1\theta_1'$, then a substitution from C_2 to C_2' implies that $\theta_1 = \theta_1'$ under not very constraining conditions. Thus for a fixed substitution θ_1, we can build a partial ordering of C_2's on $C \vee \sim C_1\theta_1$ according to their generalities. For different θ_1's, the C_2's are incomparable. The theorem about comparing C_2's with respect to different θ_1's has as a corollary a result of (Muggleton, 1990).

For the sake of brevity, we omit most proofs of theorems; the interested reader is referred to (Nienhuys-Cheng, 1990).

2. Consistent term mappings

In this paper we use a language of first order logic. The *constants* are denoted by a, b, c,... and the variables are denoted by x, y, z, u, v, w, ... The letters P, Q, R, ... are used to denote *predicates* and the letters f, g, h, ... are used to denote *functions*. A *term* is either *simple*, i.e. a constant or variable, or *compound* which has the form of $f(t_1, t_2, ..., t_n)$ where t_i's are terms and f is n-ary. An *atom* has the form of $P(t_1,\ldots,t_n)$ where P is an n-ary predicate and t_i's are terms. The negation of an atom has the form ~M where M is an atom and we call an atom or the negation of an atom a *literal*. A *clause* has the form $L_1 \vee L_2 \vee \ldots \vee L_n$ where every L_i is a literal.

2.1 Term occurrences

Let P(x,y) be a given clause. A mapping which maps x to f(u) and y to f(u) can be used to denote the action of substituting x and y in this clause both by f(u). The result is P(f(u),f(u)). If we want to do this action reversely, the function to map f(u) to x or y is not enough and we have to specify that the first f(u) is mapped to x and the second f(u) is mapped to y. Thus we need to define positions of terms. This notation is also used in (Plotkin, 1970; Muggleton & Buntine, 1988).

Definition. A *position* is a sequence $<n_1, n_2, ..., n_j>$ of positive integers. Let X be a term, literal or a non-unit clause. We use $<>$ to denote the position of X related to itself. If $X=L_1 \vee L_2 ... \vee L_n$, $n \geq 2$ is a clause, then $<i>$ is used to denote the position of L_i in X. If $Y(t_1,...,t_n)$ is a term or a literal in X with position $<p_1,p_2,...,p_k>$, then t_i has the position $<p_1,p_2,...,p_k,i>$. A *term occurrence* in X is a pair (t,p) which is used to denote the term t found at position p in X.

For example, if $X=P(f(x),y) \vee Q(f(x))$, the position of P(f(x),y) is $<1>$, the position of y in P(f(x),y) is $<2>$ but in X is $<1,2>$.

Notice that in one term or clause the position determines the term occurrence completely. If (t,p) and (s,q) are term occurrences in X where $p=<p_1,\ldots,p_k>$ and $q=<p_1,\ldots,p_k,q_1,...,q_j>$, then in position $q'=<q_1,\ldots,q_k>$ of t we find the term s, i.e. (s,q') is a term occurrence in t. In this situation (s,q) is called a *subterm occurrence* of (t,p) and we denote the relation by (t,p)≥(s,q). We also say that p is a *subsequence* of q, and we can use q-p to denote q' and pq' to denote q. If p=q, then (t,p)=(s,q); if q is longer than p, then (s,q) is called a *proper subterm occurrence* of (t,p), denoted by (t,p)>(s,q). Notice that a variable or a constant occurrence has longest position specification because they do not have proper subterm occurrences.

2.2 Consistent term mappings

If a clause C is given, it is easy to construct the set T(C) of all term occurrences of C. We can ask the following reverse question: what kind of set K of pairs of term and position (t,p) can be used to construct a clause C which has K as a subset of T(C)? For example, the set K={(f(x,g(y)),<1>), (h(y),<2>), (g(y),<1,2>} can be used to construct a clause P(f(x,g(y)),h(y)) for a 2-ary predicate P. A set K'={(f(x,g(y)),<1>), (h(y),<2>), (k(y),<1,2>)} cannot be used to construct a clause because <1> and <1,2> are nested but in position <2> of f(x,g(y)) is not k(y). In a way we can say a new clause can be constructed only if we can glue the terms together so that the terms coincide if the positions coincide. For a given clause C, we can also replace some term occurrences by new term occurrences and hence construct a new clause. For this purpose we define consistent term mappings.

Definition. An *abstract term occurrence* is a pair of term and position (t,p) which is not yet associated to a special clause. For a given clause C, a mapping θ from a subset of T(C) to a set of abstract term occurrences is called *consistent term mapping* (abbreviated as *CTM*) if the following condition is satisfied:

1) For every (t,p) in the domain of θ, (t,p)θ=(s,p). That is to say θ preserves positions.
2) If (t,p) and (s,q) are in the domain and (t,p)≥(s,q), then (t,p)θ≥(s,q)θ. That is, if (t,p)θ=(t',p) and (s,q)θ=(s',q), then in t' we find s' in position q-p.

We say that a CTM has *minimal set* as domain, if for every two different (t,p), (s,q) in the domain, p is not a subsequence of q and q is not a subsequence of p. In other words, one is not a subterm occurrence of the other. If we have a CTM θ defined on $\{(t_1,p_1),...,(t_n,p_n)\}$ and $(t_i,p_i)\theta=(t_i',p_i)$ for all i, we can denote this mapping also by $\{(t_1/t_1',p_1),...,(t_n/t_n',p_n)\}$. Such a CTM with minimal domain can be used to construct a new clause. We just replace every (t_i,p_i) in the original clause by (t_i',p_i). Because p_i is not a subsequence of p_j for different i, j, the replacement of such term occurrences do not interfere with each other. We can consider construction of new clauses also for more general CTM's. For example, let C=P(f(g(u),v),g(u)) and the CTM be {(f(g(u),v)/k(x,y),<1>)}, then the new clause is C'=P(k(x,y),g(u)). We can also consider C' to be the induced clause by a CTM with bigger domain, namely, {(f(g(u),v)/k(x,y),<1>), (g(u)/x,<1,1>)} because in <1,1> of C' is x and in <1,1> of C is g(u). A CTM {(g(u)/x,<1,1>)} induces a different clause P(f(x,v),g(u)).

Theorem 1. Let θ be a CTM defined on a subset T of T(C). Let $T\theta$ be the set of images of θ. Then there is a subset S of T which is minimal and θ restricted to S induces a clause C' such that $T(C') \supset T\theta$.

Proof. Let S be the subset of T which consists of occurrences with shortest position specification, i.e. $(t,p) \in S$ iff there is no other (t',p') in T such that (t',p')>(t,p). For every (t,p) in S, we replace (t,p) in C by $(t,p)\theta$. The result is a clause C'. The proof proceeds by showing that every $(t,p)\theta$ for (t,p) in T is a term occurrence in C'.

From now on we use $C\theta$ for the clause C' defined as in this theorem and we say it is induced by θ. Notice that the inverse θ^{-1} of a CTM θ is also a CTM. Thus, if $C\theta=C'$, then $C=C'\theta^{-1}$. This theorem tells us every CTM can be reduced to a CTM with minimal domain. Why not define CTM's with the restriction of minimal domains? In following sections we compare two clauses induced by different mappings. There we need to consider CTM's with bigger domains. Although we can derive many properties about CTM's in general (Nienhuys-Cheng, 1990), here we pay attention to two special kinds of CTM's: substitutions and inverse substitutions, and CTM's which induce the same clauses as them.

2.3 Substitutions and inverse substitutions

Let C be a clause. A *substitution* θ from C is a CTM defined on the set of all variable occurrences which maps the same variable to the same term. That is to say: if $(v,p)\theta=(t,p)$ and $(v,q)\theta=(t',q)$, then t=t'. A substitution induces a mapping defined on the set of all variables. For convenience we use θ also for this mapping and we write $(v,p)\theta=(v\theta,p)$. We define substitution with domain on all variable occurrences for the convenience of term partition in the following section. Under this definition a variable can also be mapped to itself. We use often $\{v_1/t_1, v_2/t_2,.., v_n/t_n\}$ to denote a substitution where v/v can be omitted if we want. If θ is a substitution, then the inverse θ^{-1} of θ is called *inverse substitution*. We can define inverse substitution without first considering the existence of a substitution. A CTM σ defined on a subset of T(C) for a clause C is an inverse substitution iff the following conditions are satisfied: the domain is minimal; the images are variable occurrences; if $(t,p)\sigma=(v,p)$ and $(t',q)\sigma=(v,q)$, then t=t'; for every variable occurrence (w,q) of C, there is a (t,p) in the domain of σ such that $(t,p)\geq(w,q)$. The last condition guarantees that the inverse σ^{-1} of σ is defined on all variable occurrences, to ensure that the inverse of an inverse substitution is a substitution. Notice that both substitutions and inverse substitutions have minimal domains. A substitution θ from C can be extended to a CTM $\underline{\theta}$ with maximal domain, i.e. T(C). We define (t,p)$\underline{\theta}$=(t',p) where t' is obtained by replacing all variable occurrences in t by their images. An inverse substitution σ from C can also be extended to a CTM $\underline{\sigma}$ with a maximal domain. If (t,p) is in T(C) and there is a (s,q) in the domain of σ such that (t,p)≥(s,q), then define (t,p)$\underline{\sigma}$=(t',p) where t' is obtained by replacing all subterm occurrences in (t,p) which are also in the domain by their image variable occurrences. If (t,p) in T(C) contains no element from the domain of σ as subterm occurrence and is also not a subterm occurrence of such an element, then (t,p)$\underline{\sigma}$=(t,p).

There are still other extensions of a substitution which have domains between the maximal domain and the original domain. All such extensions induce the same clause as the original substitution. In fact these are not the only CTM's which induce the same clause. For example, consider C=P(g(f(x)),y) and a substitution θ={x/h(u,v)}. It induces the clause C'=P(g(f(h(u,v)),y). A CTM defined by {f(x)/f(h(u,v)),<1,1>} induces the same clause. With these ideas in mind we can prove theorem 2 and 3. Theorem 3 is used to prove theorem 5.

Theorem 2. Let μ be a substitution from C to Cμ and $\underline{\mu}$ be the maximal extension of μ defined on T(C). Let θ be another CTM on a subset T of T(C) which is the same as $\underline{\mu}$ restricted to T. Furthermore, suppose that for every variable occurrence (v,q) in T(C), there is a (t,p) in T such that (t,p)≥(v,q), then θ induces also Cμ, i.e. Cμ=Cθ.

Theorem 3. Let μ be an inverse substitution from C and it induces Cμ. Let $\underline{\mu}$ be the maximal extension of μ. If θ is a CTM, defined on a subset T of T(C) which is the same as $\underline{\mu}$ restricted to T, and for every (s,q) in the domain of μ there is a (t,p) in T such that (t,p)≥(s,q), then Cμ=Cθ.

3. Term partitions and their comparisons

In this paper the role of inverse substitutions is important because we want to generalize clauses. We can divide the domain of an inverse substitution into a partition according to the the image variables. For example, for P(f(x),g(f(x)),h(x)) we can define inverse substitution {(f(x)/v,<1>), (f(x)/v,<2,1>), (h(x)/w,<3>)} and it induces P(v,g(v),w). Thus we have a partition {(f(x),<1>), (f(x),<2,1>)} and {(h(x),<3>)} of the domain which corresponds to the variables v and w.

Let C be a clause and μ be an inverse substitution defined on T. We can define a partition Π in T by dividing T in blocks. A block B defined by the variable v is the set

$$B=\{(t,p)\in T|(t,p)\mu=(v,p)\}.$$

We use B/v to denote that B is defined by v.

Let μ and ∂ be two inverse substitutions which define the same partition Π. Then the clauses $C\mu$ and $C\partial$ differ only in the name of variables. If we are only interested in the structure of the induced clauses without concern for the names of variables, then we can use $C(\Pi)$ to denote one of such clauses. We want to define a partial ordering in partitions $\geq$ such that $\Pi\geq\Omega$ iff $C(\Pi)\geq C(\Omega)$, i.e. there is a substitution σ from $C(\Pi)$ to $C(\Omega)$. If $C_1\geq C_2$, then for every (w,q) variable in C_2, there must be a (v,p) in C_1 such that $(v,p)\sigma$ contains (w,q) as subterm. In this situation w has relative position q-p in $(v,p)\sigma$. If there is also (v,p') in C_1, then $(v,p')\sigma$ contains also a variable w in the position q-p(=q'-p'). We try to translate such concepts to relations between partitions. For example,

$C=P(f(g(h(x),y))), g(h(x),y),k(a,h(x)))$

$C_2=P(f(g(w,y)), g(w,y), k(a,w))$

$C_1=P(f(u),u,v)$

To find C_2, we need the following partition Ω:

$D_1=\{(h(x),<1,1,1>), (h(x),<2,1>), (h(x), <3,2>)\}$, D_1/w;

$D_2=\{(y,<1,1,2>),(y,<2,2>)\}$,$D_2/y$

To find C_1, we need the following partition Π:

$B_1=\{(g(h(x),y),<1,1>), (g(h(x),y),<2>)\}$, B_1/u;

$B_2=\{(k(a,h(x)),<3>)\}$, B_2/v.

Notice that $(u,<1,1>)\sigma=(g(w,y),<1,1>)$ and $(u,<2>)\sigma=(g(w,y),<2>)$ and $(v,<3>)\sigma=(k(a,w),<3>)$. The first two elements in D_1 are related to B_1. For $(h(x),<1,1,1>)$ in D_1 there is $(g(h(x)),<1,1>)$ in B_1 to contain it as subterm in position $<1>$ and for $(h(x),<2,1>)$ in D_1 there is $(g(h(x)),<2>)$ in B_1 to contain it as subterm in position $<1>$. This is also the position of w in $v\sigma$. For $(h(x),<3,2>)$ in D_1 there is $(k(a,h(x)),<3>)$ in B_2 which contains it as subterm in position $<2>$. This is also the position of y in $v\sigma$. We can find similar relation between elements in D_2 and elements in B_1. Thus, first we want to define a partition without an explicit inverse substitution and then define the partial order relation $\geq$ for partitions:

Definition. Let C be a given clause. An *admissible* subset T of T(C) satisfies the following conditions:

1) T is minimal.
2) If (w,q) is an variable occurrence in C, then there is a (t,p) in T such that $(t,p)\geq(w,q)$.

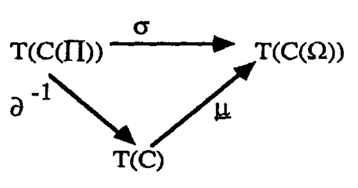
A *term partition* of an admissible subset T is a set of disjoint non-empty subsets $B_1,.., B_k$ such that $B_1 \cup ... \cup B_k = T$, and every *block* B_i contains occurrences of only one term. Notice that every partition defined by an inverse substitution is also a term partition. On the other hand, we can define an inverse substitution μ from T such that Π is also the partition induced by μ. We just define $(t,p)\mu=(v_i,p)$ if (t,p_i) is in B_i. Thus we can call the partition induced by an inverse substitution also term partition.

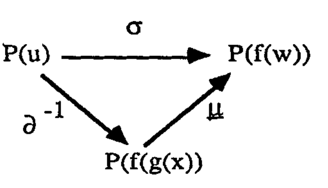
Definition. Let C be a given clause and T, S be admissible subsets of T(C). Let Π be a term partition defined on T and Ω be a term partition defined on S. We say Π≥Ω if

1) For every (s,q) in S, there is a (t,p) in T such that (t,p)≥(s,q).
2) Let (t,p) be in a block B of Π and (s,q) be in a block D in Ω. If (t,p)≥(s,q), and $B=\{(t,p_1),...,(t,p_n)\}$, $D=\{(s,q_1),..,(s,q_m)\}$, then m≥n and by reordering the indices, we have $p_1=p$, $q_1=q$ and $q_i-p_i=q-p$ for every i=1,...,n.

Theorem 4. Let C be a given clause. Let ∂ and μ be two inverse substitutions which induce term partitions Π and Ω on T and S, admissible subsets of T(C), respectively. If there is a substitution from C∂ to Cμ, then Π≥Ω.

To prove that Π≥Ω implies also C(Π)≥C(Ω), we use theorem 3 of the last section which tells when a CTM induces the same clause as the inverse substitution. As an example, let C=P(f(g(x))). Let C(Π)=P(u) and ∂ be the inverse substitution from C to C(Π). Let C(Ω)=P(f(w)) and let μ be the inverse substitution from C to C(Ω). Let $\underline{\mu}$ be also the maximal extension of μ and ∂^{-1} be the substitution which is the inverse of ∂. We can use the composition of ∂^{-1}: u → f(g(x)), $\underline{\mu}$: f(g(x)) → f(w) to define the composition σ: u → f(w). This CTM induces a clause based on C(Π) and we can prove it is just C(Ω). That means σ is the substitution which we are looking for. The following diagram illustrates the situation. In the right diagram we left the letter T out to make things look more transparant.





Theorem 5. Let C be a given clause and let Π and Ω be two term partitions defined on S and T, admissible subsets of T(C), respectively. Let C(Π) and C(Ω) be two clauses induced by Π and Ω, respectively. If Π≥Ω, then there is a substitution σ from C(Π) to C(Ω).

For the given clause C, the relation $\geq$ forms a partial ordering on all term partitions which are defined on subsets of T(C). The minimal term partition under this ordering induces the clause C itself. The ordering coincides with the generality ordering on clauses, which allows us to compare clauses without actually building them. The absorption algorithm, discussed in the next section, is based on such term partitions. A related problem is the construction of minimal generalizations of a given clause, and of the supremum of clauses (Plotkin, 1970; Reynolds, 1970). In (Nienhuys-Cheng, 1991) we consider all partitions based on C and we give algorithms for building the least higher partitions (w.r.t. $\geq$) for a given partition and the supremum of two partitions.

4. Absorption

We briefly review the basic concepts related to resolution. Let L_1 and L_2 be two literals. A *unifier* of the L_1 and L_2 is a pair of substitutions (θ_1,θ_2) such that θ_1 is defined on all variable occurrences of L_1 and θ_2 is defined on all variable occurrences of L_2 and $L_1\theta_1=L_2\theta_2$. A unifier (θ_1,θ_2) is called a *most general unifier* (*mgu*) if for any unifier (σ_1,σ_2) for L_1, L_2 there is a substitution γ such that $L_1\theta_1\gamma=L_2\theta_2\gamma=L_1\sigma_1=L_2\sigma_2$ where $L_i\theta_i\gamma$ is the clause induced by γ based on $L_i\theta_i$.

To define the resolution principle we need to know first how to extend a substitution from a literal to a clause which contains this literal. If C is a clause such that $C=C'\vee L$ where L is a literal, then a substitution θ on L can be extended to a substitution on the entire clause C. If $v\theta=t$, then for every (v,p) in C we can define $(v,p)\theta=(t,p)$. Let $C_1=C_1'\vee L_1$, $C_2=C_2'\vee L_2$ be two clauses. If (θ_1,θ_2) is a mgu of $\sim L_1$ and L_2, then the resolution principle allows to infer $C_1'\theta_1\vee C_2'\theta_2$. This is called a *resolvent* of C_1 and C_2.

4.1 MB-absorption and a new algorithm

In the introduction, we demonstrated the incompleteness and unsoundness of MB-absorption. On the other hand, (Muggleton, 1990) demonstrates that for any C_2 constructed by MB-absorption from C_1 and C, there are substitutions θ_1 and θ_2 such that $C_2\theta_2=C\vee\sim C_1\theta_1$. Thus, the resolvent of C_1 and C_2 is either C, or some clause more general than C. Because in machine learning we are looking for generalizations, we may take this as an alternative soundness condition. We have a sound and complete absorption algorithm, if it can construct all, and only those, C_2's such that $C_2\theta_2$=C∨~$C_1\theta_1$. Essentially, such an algorithm first constructs θ_1 from C_1 and then constructs an inverse substitution θ_2^{-1} from an admissible subset of T(C∨~$C_1\theta_1$) by means of a term partition.

Algorithm. *A non-deterministic, sound and complete absorption algorithm.*
Input: clauses C and C_1, where C_1 is a positive literal.
Output: C_2 such that $C_2\theta_2$=C∨~$C_1\theta_1$ for some θ_1 and θ_2.
Construct a substitution θ_1 from C_1;
Construct an admissible subset T of T(C∨~$C_1\theta_1$);
Construct a term partition of T;
Construct an inverse substitution and an induced clause C_2 from this partition.

To find an admissible subset T, we can begin with considering a set S of all variable occurrences plus some constant occurrences (optional). Initially, T:=∅. For every (s,q) in S, find a (t,p) in T(C∨~$C_1\theta_1$) such that q contains p as a subsequence. If this (t,p) is not already in T, and there is no (t',p') in T with the property that p and p' have subsequence relationship, let T:=T∪{(t,p)}. When all elements in S have been considered, we have an admissible set. We define the partition in the following way. Let (t,p) in T, find some elements in T such that they are occurrences of the same term t. Define a block B by including these elements and (t,p). We repeat the same process for elements in T:=T-B. The partition is ready when there is no element in T left.

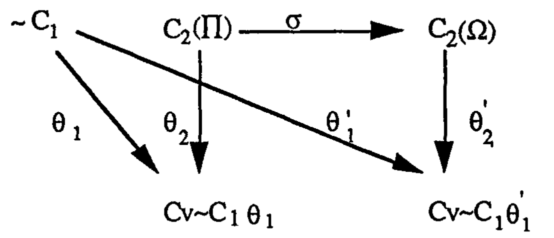
C_2's based on the same θ_1 can be compared by means of the term partitions of section 3. Notice that C∨~$C_1\theta_1$ is the least general C_2 which can be constructed by using the same θ_1; it will be called an *LG-absorption*. The algorithm above is not directed and therefore inefficient. This can partly be remedied by constructing partitions which are least higher (w.r.t. ≥) compared to a given partition based on C∨~$C_1\theta_1$, and constructing the supremum of some partitions based on C∨~$C_1\theta_1$ (Nienhuys-Cheng, 1991).

4.2 Comparison of C_2 induced by different θ_1's

Let C=P(f(x)) and C_1=Q(y) and θ_1={y/f(x)}, then we have C∨~$C_1\theta_1$=P(f(x))∨~Q(f(x)). For some term partition Π we have C_2(Π)=P(u)∨~Q(u). For any other θ_1', we have C∨~$C_1\theta_1$'=P(f(x))∨~Q(X), with X an unknown term. Suppose there is a term partition Ω on a subset of T(C∨~$C_1\theta_1$') such that C_2(Ω)=P(f(v))∨~Q(Y) and a substitution from C_2(Π) to C_2(Ω), then it must bring u to f(v); hence Y=f(v). The variable v determines a block in the term partition Ω, and from C we know (x,<1,1,1>) must be in the block. Therefore, X=f(x) and $\theta_1=\theta_1'$. In general, C_2's are incomparable if they are built on different θ_1's which satisfy a certain condition.

Lemma. Let C be a clause and C_1 be a literal. Consider two substitutions θ_1, θ_1' from C_1. Let Π be a term partition defined on a subset of $T(Cv{\sim}C_1\theta_1)$, and let Ω be a term partition defined on a subset of $T(Cv{\sim}C_1\theta_1')$. Let θ_2 and θ_2' be the substitution from $C_2(\Pi)$ to $Cv{\sim}C_1\theta_1$ and $C_2(\Omega)$ to $Cv{\sim}C_1\theta_1'$, respectively. Suppose there is a substitution σ from $C_2(\Pi)$ to $C_2(\Omega)$ and suppose a block B, B/v of Π contains both terms from C and ${\sim}C_1\theta_1$, then if (t,p), a term occurrence in $T(Cv{\sim}C_1\theta_1)$, is in B, then (t,p) is also in $T(Cv{\sim}C_1\theta_1')$ and $(t,p)=(v\sigma,p)\underline{\theta}_2'$ where $\underline{\theta}_2'$ is the maximal extension of θ_2'.

Proof. The relations between different mappings can be seen in the following diagram:



Let us consider the following block of Π: $B=\{(t,p_1),\ldots,(t,p_n),(t,q_1),\ldots,(t,q_m)\}$, B/v where (t,p_i) are term occurrences of C and (t,q_j) are term occurrences of ${\sim}C_1\theta_1$. From the given condition about a block of Π we know m>0 and n>0. We consider the set

$$B'=\{(v\sigma\underline{\theta}_2',p_1),\ldots,(v\sigma\underline{\theta}_2',p_n),(v\sigma\underline{\theta}_2',q_1),\ldots,(v\sigma\underline{\theta}_2',q_m)\}$$

B is a subset of $T(Cv{\sim}C_1\theta_1)$ and B' is a subset of $T(Cv{\sim}C_1\theta_1')$. Furthermore, (t,p_i) is the p_i-th term of C and so is $(v\sigma\underline{\theta}_2',p_i)$. Thus if the set of (t,p_i)'s in B is not empty, then $t=v\sigma\underline{\theta}_2'$. Thus (t,q_j) is also a term occurence in $Cv{\sim}C_1\theta_1'$.

Theorem 6. Let C, C_1, θ_1, θ_1', C_2, C_2', Π, Ω, θ_2 and θ_2' be defined as in the lemma. Suppose there is a substitution σ from $C_2(\Pi)$ to $C_2(\Omega)$ and every block in Π which contains term occurrences from ${\sim}C_1\theta_1$ contains also term occurrences from C. Then for every variable w in C_1, we have $w\theta_1=w\theta_1'$.

The proof goes as follows. If w is a variable such that there is a (t,q_j) in block B such that $(t,q_j)\geq(w,q)\theta_1$, then $(w,q)\theta_1'$ is the same subterm of (t,q_j) with position $q\text{-}q_j$ because θ_1' preserves positions and (t,q_j) is also in $T(Cv{\sim}C_1\theta_1')$ from the lemma. If $(w,q)\theta_1$ contains

$(t_1,p_1),...,(t_k,p_k)$ which belong to blocks $B_1,...,B_k$, then (t_i,p_i) are also in $T(C\vee\sim C_1\theta_1')$ from the lemma and they are subterms of $(w,q)\theta_1'$ and in fact $(w,q)\theta_1=(w,q)\theta_1'$.

Let us define two conditions, V: $C_1\theta_1$ *should contain only variables occurring in* C and W*: if a block* B *in a partition to define* C_2 *contains a term in* $\sim C_1\theta_1$ *then it contains also term occurrences in* C. If for some θ_1 there is a C_2 which satisfies W, then θ_1 satisfies V because every variable occurrence in $\sim C_1\theta_1$ is contained in a term occurrence in a block and the same term occurs also in C. On the other hand, if θ_1 satisfies V, then we can take the trivial $C\vee\sim C_1\theta_1$ as a C_2 which satisfies W. Thus, V for θ_1 is equivalent with the existence of a C_2 satisfying W.

Consider the set of all C_2's satisfying W, based on some θ_1 satisfying V: this set is partially ordered, but C_2's based on different θ_1's cannot be compared. In fact, we can prove that $C\vee\sim C_1\theta_1$ for θ_1 satisfying V is least general, i.e. there exists no substitution from it to a C_2 based on another θ_1 (not necessarily satisfying V). If θ_1 does not satisfy condition V, then it may result in a C_2 which is not least general.

4.3 Related work

Muggleton (1990) investigates how to construct a least general C_2 if C_1 is not a literal. Let $C_1=C_1'\vee L_1$, $C_2=C_2'\vee L_2$, L_1 (positive) and L_2 (negative) are the literals resolved upon and (θ_1,θ_2) is the mgu. Muggleton argues that $C_2'\theta_2$ should contain every literal in $C_1'\theta_1$, hence $(C_2'\vee L_2)\theta_2=C\vee\sim L_1\theta_1$. Furthermore, θ_1 can partly be derived by comparing literals in C_1' and C, because $C_1'\theta_1$ must be a part of C. Therefore, he requires the variables in the head of C_1 to be in its body, to assure $C\vee\sim L_1\theta_1$ is least general. On the other hand, for us to construct such a C_2 means to construct a term partition on $C\vee\sim L_1\theta_1$, and the condition concerning the variables in C_1 implies θ_1 satisfies V. Thus, his result is a corollary of our theory. We allow in addition C_1 to be a unit clause.

Unlike what is implicitly suggested in (Muggleton, 1990), such a θ_1 is not necessarily unique. For instance, let $C=P(x,y)\vee\sim R(x)\vee\sim R(y)$ and $C_1=Q(z)\vee\sim R(z)$, then we can take $\theta_1=\{z/x\}$ resulting in $C_2=P(x,y)\vee\sim R(x)\vee\sim R(y)\vee\sim Q(x)$, but also $\theta_1'=\{z/y\}$ which yields $C_2'=P(x,y)\vee\sim R(x)\vee\sim R(y)\vee\sim Q(y)$. Both C_2 and C_2' are least general.

(Rouveirol & Puget, 1989) present an approach to inverse resolution, based on a representation change. Before applying an inverse resolution step to clauses, they are *flattened* to clauses containing no function symbols (the functions are transformed to predicates). This simplifies the process of inverse resolution. For instance, an inverse substitution on a flattened clause amounts to dropping some literals from the clause. Within our framework, the completeness of their Absorption operator could be analysed as well.

5. Conclusions

In this paper, we have formalized the language for discussing problems about inverse resolutions by using consistent term mappings; we compare the clauses by comparing the partitions and we have improved the absorption algorithm. Finally, we have extended Muggleton's result about least general absorption by allowing C_1 to be a unit clause.

References

Stephen Muggleton. Inductive Logic Programming. First Conference on Algorithmic Learning Theory, Ohmsha, Tokyo, October 1990.

Stephen Muggleton & Wray Buntine. Machine Invention of First-order Predicates by Inverting Resolution. Proceedings of the 5th International Conference on Machine Learning, Morgan Kaufmann, pp. 339-351, 1988.

Shan-Hwei Nienhuys-Cheng. Consequent Functions and Inverse Resolutions. Report Eur-CS-90-03, Erasmus University, Rotterdam, Netherlands, May 1990.

Shan-Hwei Nienhuys-Cheng. Term Partitions and Minimal Generalizations of Clauses. Report, Erasmus University, Rotterdam, Netherlands, 1991.

Gordon D. Plotkin. A Note on Inductive Generalisation. *Machine Intelligence 5*, B. Meltzer & D. Michie (eds.), Edinburgh University Press, 1970.

John C. Reynolds. Transformational Systems and the Algebraic Structure of Atomic Formulas. *Machine Intelligence 5*, B. Meltzer & D. Michie (eds.), Edinburgh University Press, 1970.

Céline Rouveirol & Jean-Francois Puget. A Simple Solution for Inverting Resolution. EWSL-89, Pitman, London, pp. 201-210, 1989.