

# The V- and W-operators in Inverse Resolutions

*Shan-Hwei Nienhuys-Cheng*  
*Department of Computer Science*  
*Erasmus University Rotterdam*  
*Rotterdam, Netherlands*

## 1 Introduction

Resolutions are often used to conclude results from given clauses. In machine learning we need sometimes to consider some given facts as results of some clauses and we want to find these clauses. This is the background of inverse resolution. Muggleton and Buntine[1] have defined V- and W-operators for inverse resolutions in 1988. They have also implemented them in a system CIGOL. These two operators consider two different situations for inverting resolutions. [11,12] use flattening to find solutions of these two operators. [3,6] use term partitions to find solutions of V-operators. A comparison of these two approaches are given in [8]. In fact, both approaches can be considered as different ways of coding the same problems related to the trees of clauses[7]. In this article we consider these problems in different levels. We are most interested in the first and the second level. The first level (highest) considers a clause as a set of literals. The second level considers a clause as a disjunction of literals which we call a *clausal form*. The third level applies so-called *integer coding* to clausal forms to find term partitions. The fourth level, also the lowest, is the implementation of the operators. In this article we are not going to talk about the implementation level.

Finding generalizations of clauses is important for solving problems in inverse resolutions. We shall explain this problem in the first and second levels so that we get more insight about generalizations and also about levels. In the second level we consider clauses as clausal forms. Let C and D be two clausal forms. D is called a *generalization* of C if there is a substitution  $\mu$  such that  $D\mu=C$ . This definition is purely syntactic and inflexible. For example if  $C=P(a,a)\vee Q(a,a)$ , then  $D_1=P(x,y)\vee Q(x,y)$  is a generalization of C and  $D_2=Q(x,y)\vee P(x,y)$  is not a generalization of C. If we consider the same problem in the first level, then  $\{P(x,y),Q(x,y)\}$  and  $\{Q(x,y),P(x,y)\}$  do not make any difference. They represent the same set which is a generalization of  $\{P(a,a),Q(a,a)\}$ . Another problem can also be explained by an example. The two clausal forms  $C_1=P(f(a),f(a))$  and  $C_2=P(f(a),f(a))\vee P(f(a),f(a))$  represent the same clause (set form)  $\{P(f(a),f(a))\}$ . The clause  $D=P(f(x),f(y))\vee P(f(a),z)$  is a generalization of  $C_2$  but not of  $C_1$  in the second level. On the other hand  $\{P(f(x),f(y)),P(f(a),z)\}$  is a generalization of  $\{P(f(a),f(a))\}$  in the first level. For us the first level makes more sense because it has more to do with the semantic of generalizations. On the other hand, the second level is more manipulable. The best way to approach the problems in generalizations and inverse resolutions is first to formulate a problem in the highest (first) level. We translate it then to problem(s) in the lower levels and solve the problems in lower levels.

In chapter 2 we are going to define clauses (in set forms) and their corresponding clausal forms. The V- and W-operators shall also be formulated in both forms in this chapter.

In chapter 3 we are going to introduce two order relations defined by the two ways of generalizations which we have just talked about. The problems about order relation in the first level can also be translated to problems about order relation in the second level.

In chapter 4 we define minimal proper generalizations and the supremum of clausal forms w.r.t. to the order relation in the second level. For the results about minimal proper generalizations we can look up in [4]. For the supremum of compatible clausal forms we need only to generalize the results of Plotkin in [9].

In chapter 5, 6 we use the results of chapter 3, 4 to solve the problems in V-operators and W-operators respectively.

In chapter 7 finding generalizations by using the integer coding and term partitions shall be quickly explained. We look at the problems V- and W-operators once more by using this coding.

## 2 V- and W-operators

### 2.1 Preliminaries in the first order logic

In this article we use a language of first order logic. The *constants* are denoted by a, b, c, etc. The *predicates* are denoted by P, Q, R, etc. and the *variables* are denoted by u, v, w, x, y, z,  $u_1$ , etc. The letters f, g, h, etc. are used to denote *functions*. A term is either *simple*, i.e. a constant or variable, or *compound*, i.e. it has the form  $f(t_1, \dots, t_n)$  where  $t_i$  is a term and f is n-ary. An *atom* has the form  $P(t_1, \dots, t_n)$  where  $t_i$  are terms and P is n-ary. The *negation* of an atom M is denoted by  $\sim M$ . An atom or the negation of an atom is called a *literal*. A *clause* is a set of literals  $\{L_1, \dots, L_n\}$  where every  $L_i$  is a literal. A clause  $C = \{L_1, \dots, L_n\}$  can be expressed in *clausal-form*  $L_{i1} \vee L_{i2} \dots \vee L_{im}$  if  $\{L_{i1}, \dots, L_{im}\} = \{L_1, \dots, L_n\}$ . A clause is actually an equivalence class of clausal forms where two clausal forms in the same class are composed of the same literals. If two clausal forms of the same clause do not contain repeated literals, then one form can be achieved by permutation of literals in the other form of the same clause. We can extend the equivalence class even further. Two clauses are equivalent if they differ only by variable names. Thus the equivalence class of clausal forms can be extended by using this equivalence relation in the clauses. That is to say, two clausal forms belong to the same *extended equivalent class* if they represent two clauses which are equivalent. When we deal with problems about clauses, we are often satisfied with finding one clause as solution without considering the names of variables. Thus when we translate a problem into clausal forms, we are also satisfied if we can find one clausal form in an extended equivalent class.

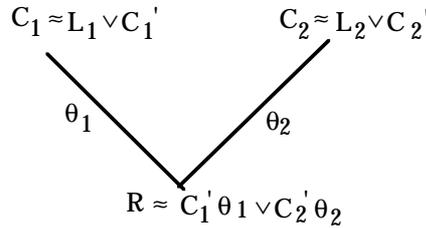
If C is a clause and D is a clausal form to represent C, we use  $C \approx D$  to denote this relation. If C is a clause and D is a clausal form made of some literals in C, then  $C - D$  is the subset of C without the literals in D. Unless explicitly stated, we usually assume that a clause or a clausal form does not contain or is not written with repeated literals.

Let two clauses  $C_1$  and  $C_2$  be given and  $L_1 \in C_1$  and  $L_2 \in C_2$ . If there exists a pair of substitutions  $(\theta_1, \theta_2)$  from  $C_1$  and  $C_2$  respectively such that  $L_1\theta_1 = \sim L_2\theta_2$ , then  $C = (C_1 - L_1)\theta_1 \cup (C_2 - L_2)\theta_2$  is called the *resolvent* of  $C_1$  and  $C_2$ , and  $L_1$  and  $L_2$  are the literals to be *resolved on*. Notice that we use a unifier  $(\theta_1, \theta_2)$  of  $L_1$  and  $L_2$  instead of using a most general unifier for the definition of resolution. We can also express the resolution with clausal forms. Let the clause  $C_1$  be represented by  $L_1 \vee C_1'$  and  $C_2$  be represented by  $L_2 \vee C_2'$  where  $C_1 - C_1' = L_1$  and  $C_2 - C_2' = L_2$ , then  $C_1'\theta_1 \vee C_2'\theta_2$  represents a *resolvent* of  $C_1$  and  $C_2$ . We call  $C_1'\theta_1 \vee C_2'\theta_2$  also a *resolvent*. Notice that even if we write  $C_1'$  and  $C_2'$  not with repeated literals,  $C_1'\theta_1 \vee C_2'\theta_2$  can contain some literal more than once.

### 2.2 V-operator

Let two clauses  $C_1 = \{L_1\} \cup C_1'$  and R be given. A *V-operator* is an algorithm which finds clauses  $C_2 = \{L_2\} \cup C_2'$  such that R is a resolvent of  $C_1$  and  $C_2$  and  $L_1$  and  $L_2$  are the literals to be resolved

on. If  $L_1$  is a positive literal, then we say that this V-operator is an *absorption*. This V-operator is *complete* if it can find all such  $C_2$ . We can also use the clausal forms to express the V-operators. If a clause  $C_1$  is represented by  $L_1 \vee C_1'$  with no repeated literals. A V-operator is then an algorithm to construct a clause  $C_2$  represented by  $L_2 \vee C_2'$  such that the resolvent of  $L_1 \vee C_1'$  and  $L_2 \vee C_2'$ , with  $L_1, L_2$  the literals being resolved on, represents the clause  $R$ . Thus a V-operator is complete if for every extend equivalence class of clausal forms of  $C_2$  we can find one such  $L_2 \vee C_2'$ .



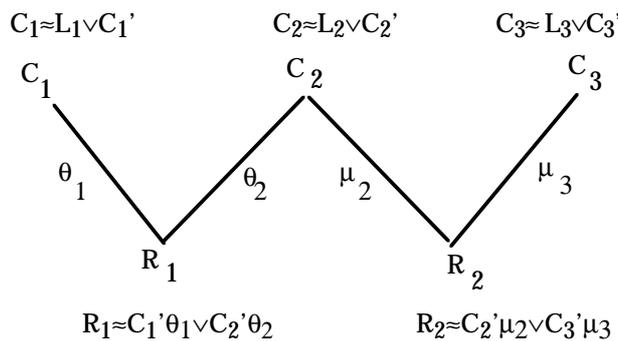
The simplest situation of V-operator is when  $C_1'$  is empty. For given  $\theta_1$  we have  $R \approx C_2' \theta_2$ . Thus

$$\{\sim L_1 \theta_1\} \cup R \approx \sim L_1 \theta_1 \vee C_2' \theta_2 = L_2 \theta_2 \vee C_2' \theta_2 = (L_2 \vee C_2') \theta_2 \approx C_2 \theta_2.$$

Thus we should find  $C_2$  such that  $C_2 \theta_2 = \{\sim L_1 \theta_1\} \cup R$ , i.e. the generalizations of  $\{\sim L_1 \theta_1\} \cup R$ . If we define an order relation by substitution, then  $\{\sim L_1 \theta_1\} \cup R$  is clearly the minimum of all  $C_2$  w.r.t the same  $\theta_1$ . If we have found a  $C_2$ , then it is plausible that we want to find other  $C_2$  which are higher w.r.t. this order relation, but not too high. All these questions can be translated to questions in clausal forms. We will discuss an order relation and the minimal proper generalizations on the clausal forms in chapter 3, 4.

### 2.3 W-operator

Given clauses  $R_1, R_2$ , a *W-operator* is a construction of  $C_1, C_2$  and  $C_3$  and literals  $L_1 \in C_1, L_2 \in C_2$  and  $L_3 \in C_3$  with the property that  $R_1$  is a resolvent of  $C_1$  and  $C_2$  resolved on  $L_1$  and  $L_2$  and  $R_2$  is a resolvent of  $C_2$  and  $C_3$  resolved on  $L_2$  and  $L_3$ . A W-operator is *complete* if for every pair  $(R_1, R_2)$  it can find all such triple  $(C_1, C_2, C_3)$  and related  $L_1 \in C_1, L_2 \in C_2$  and  $L_3 \in C_3$ . We can also formulate this in the language of clausal forms. For given clauses  $R_1$  and  $R_2$ , a W-operator constructs clausal forms  $C_1', C_2', C_3', L_1, L_2, L_3$  such that there are substitutions  $\theta_1, \theta_2$  and  $\mu_2, \mu_3$  with the property  $L_1 \theta_1 = \sim L_2 \theta_2$  and  $L_2 \mu_2 = \sim L_3 \mu_3$  and that  $R_1 \approx C_1' \theta_1 \vee C_2' \theta_2, R_2 \approx C_2' \mu_2 \vee C_3' \mu_3$ . A W-operator for clausal forms is *complete* if for every clauses  $R_1$  and  $R_2$  and solution  $(C_1, C_2, C_3)$  with  $L_1 \in C_1, L_2 \in C_2$  and  $L_3 \in C_3$ , defined above we can find representatives  $L_1 \vee C_1', L_2 \vee C_2', L_3 \vee C_3'$  in the extended equivalence classes related to  $C_1, C_2, C_3$ , respectively such that a resolvent of  $L_1 \vee C_1'$  and  $L_2 \vee C_2'$  represents  $R_1$  and a resolvent of  $L_2 \vee C_2'$  and  $L_3 \vee C_3'$  represents  $R_2$ .



Notice that  $C_2'$  is higher than  $C_2' \theta_2$  and  $C_2' \mu_2$  in the order relation defined by substitutions. The last two clausal forms represent subsets of  $R_1$  and  $R_2$ , respectively. Thus the best way to solve this problem is to begin with choosing subsets of  $R_1$  and  $R_2$  and finding the clausal forms of these two subsets. For every two such suitable clausal forms we need then to find the supremum of the

two clausal forms. For every  $C_2'$ , there is a substitution from it to such a supremum. Thus to find  $C_2'$  means to find more general clausal forms than such a supremum. So we treat the problem about supremum also in chapter 4.

### 3 Partial order relations on clauses and clausal forms

In this chapter we define two partial order relations with respect to substitutions. The order relation  $\geq$  has to do with clausal forms and the order relation  $\succ=$  has to do with clauses. We want to compare them and we want to translate problems related to  $\succ=$  to problems related to  $\geq$ .

#### 3.1 Partial order relations defined on clauses

**3.1.1 Definition.** Let  $C, D$  be a given clauses. If there is a substitution  $\mu$  such that  $D\mu=C$ , then we say  $D \succ= C$ .  $D$  is called a generalization of  $C$ .

**3.1.2 Proposition.** Given clauses  $D$  with  $m$  elements, and  $C$  with  $n$  elements. If there is a substitution  $\mu$  such that  $C=D\mu$ , then  $m \geq n$ .

**3.1.3 Proposition.** Given  $C=\{L_1, L_2, \dots, L_n\}$  where  $L_i \neq L_j$  if  $i \neq j$ , and  $D$  with  $m$  elements where  $m \geq n$ . If there is a substitution  $\mu$  such that  $C=D\mu$ , then there is a subset  $D'$  of  $D$  with  $n$  elements such that  $D'\mu=C$ .

**Proof.** For every  $L_i$  choose a  $M_i$  such that  $L_i=M_i\mu$ . We have  $M_i \neq M_j$  because  $M_i\mu \neq M_j\mu$ . Define  $D'=\{M_1, \dots, M_n\}$ .

**3.1.4 Remark.** Given a clause  $C$ , there can be infinite many clauses  $D$  such that  $D \succ= C$ . For example,  $C=P(x,x)$ , then  $D_1=\{P(x_1, x_2)\}$ ,  $D_2=\{P(x_1, x_2), P(x_2, x_1)\}$ ,  $D_3=\{P(x_1, x_2), P(x_2, x_3), P(x_3, x_1)\}$ , etc. are all generalizations of  $C$ . Two  $D_i$ 's are not equivalent under the definition of Plotkin[9]. Plotkin defines an equivalence relation  $\sim$  by using the concept of *subsume*. A clause  $D$  *subsumes*  $C$  if there is a substitution  $\mu$  such that  $C \supseteq D\mu$ . We say  $D \sim C$  if  $C$  subsumes  $D$  and  $D$  subsumes  $C$ , e.g.  $\{P(x,y)\} \sim \{P(x,y), P(z,u)\}$ . If  $j > i$ , then  $D_i$  subsumes  $D_j$ , but  $D_j$  does not subsume  $D_i$ . The size of the  $D_i$  can also not be reduced. Plotkin defines *reduced* as follows.  $D$  is *reduced* if for any subset  $C$  of  $D$  such that  $D \sim C$ , then  $D=C$ . We can not reduce a  $D_i$  because it is impossible to find a  $\mu$  such that  $D_i\mu$  is a proper subset of  $D_i$ . Thus for a given clause  $C$ , there is usually no way to find all the generalizations  $D$ , not even all the reduced  $D$ . However, if we require beforehand the size of  $D$  not exceeding certain limit, then the choice of  $D$  is finite. This shall be more clear when we come to order relation defined on clausal forms in 3.2.

#### 3.2 Partial order relation defined on clausal forms

**3.2.1 Definition.** Let  $D=L_1 \vee L_2 \dots \vee L_n$  and  $C=M_1 \vee M_2 \dots \vee M_n$ . We say  $D \geq C$  if there is a substitution  $\mu$  such that  $L_i\mu=M_i$  for  $i=1, \dots, n$ . In this situation  $D$  is also called a *generalization* of  $C$  (in clausal form).

**3.2.2 Proposition.** Let  $C$  be a clause with  $n$  literals and  $D=\{L_1, L_2, \dots, L_m\}$  be a clause with  $m$  literals. If  $D \geq C$ , then we can represent  $C$  by a clausal form of  $m$  literals, i.e.  $C \approx M_1 \vee M_2 \dots \vee M_m$  for some  $M_i$  such that  $(L_1 \vee L_2 \dots \vee L_m)\mu=M_1 \vee M_2 \dots \vee M_m$ .

**Proof.** Suppose  $D=\{L_1, \dots, L_m\}$  and there is a substitution  $\mu$  such that  $D\mu=C$ . Then define  $M_i=L_i\mu \in C$  for  $i=1, \dots, m$ . The clausal form  $M_1 \vee M_2 \dots \vee M_m$  represents  $C$ .

**3.2.3 Remark.** Given a clause  $C$  with  $n$  literals. We can find all the generalizations of  $C$  with  $m$  literals or less where  $m \geq n$  in the following way. Consider all the clausal representations  $C'$  of  $C$  in  $m$  literals, i.e.  $C'$  is the disjunction of all literals in  $C$  and  $C'$  may contain the same literal more than once. Then a generalization of  $C'$  represents a generalization of  $C$ . This is finite up to different variable names. We can use term partitions to make this statement more clear in chapter 7.

## 4 Minimal proper generalizations and supremum of clausal forms

### 4.1 Minimal proper generalizations

**4.1.1 Definition.** For a clausal form  $C$ ,  $D$  is called a *proper generalization* if  $D \geq C$  but  $C \geq D$  is not true. We use  $D > C$  to denote this relation.  $D$  is called a *minimal proper generalization* of  $C$  if  $D$  is a proper generalization and for any other proper generalization  $D'$  of  $C$  we have either  $D' \geq D$  or  $D'$  and  $D$  are incomparable.

**4.1.2** We can either generalize the proofs of Reynolds[10] for literals or we use term partitions[4] to prove that there are three kinds of minimal proper generalizations of a clausal form. For detailed formulation of these generalizations, see [4] and also chapter 7.

**Minimal proper generalizations of the first kind.** We can replace some constant occurrences of the same constant in  $C$  by a new variable to obtain a minimal proper generalization. For example, let  $C = P(f(x), g(x), a, g(a))$ . Then

$$P(f(x), g(x), y, g(a)), P(f(x), g(x), y, g(y)), P(f(x), g(x), a, g(y))$$

are minimal generalizations of  $C$ .

**Minimal proper generalizations of the second kind.** Another way of finding minimal proper generalizations is to change some occurrences of a certain variable with a new variable name. For example, let  $C = P(x, f(y), g(x), g(x))$ . Then

$$P(x_1, f(y), g(x), g(x)), P(x_1, f(y), g(x_1), g(x)), P(x, f(y), g(x_1), g(x_1))$$

are minimal generalizations.

**Minimal proper generalizations of the third kind.** We can construct minimal proper generalizations by replacing some kind of compound terms with only variable as arguments by a new variable. For example, let  $C = P(f(g(x, y), h(g(x, y))), a)$ . Then  $P(f(z), g(z), a)$  is a minimal generalization.

It is also proved in [4,10] that if two clausal forms  $C$  and  $C'$  are given with  $C' > C$ , then we can construct a finite sequence of clauses  $C_0 = C, C_1, C_2, \dots, C_n = C'$  such that  $C_i$  is minimal generalization of  $C_{i-1}$  where  $i = 1, \dots, n$ .

### 4.2 Supremum of clausal forms

**Definition.** Let  $S$  be a partially ordered set with some partial order denoted by  $\geq$ . Let  $T$  be a non-empty subset of  $S$ . An element  $D \in S$  is *supremum* of  $T$  if it satisfies the following conditions

- 1)  $D \geq C$  for any  $C \in T$ .
- 2) If  $D' \geq C$  for any  $C \in T$ , then  $D' \geq D$ .

**Definition.** A *word* is a literal or a term. Two words are *compatible* iff they are both terms or they have the same predicate and sign. Let  $K$  be a set of words and  $W_1, W_2 \in K$ , then  $W_1 \geq W_2$  iff there is a substitution  $\mu$  such that  $W_1 \mu = W_2$ . (Plotkin uses  $\leq$  instead of  $\geq$ ). Let  $D = L_1 \vee L_2 \dots \vee L_n$  and

$C=M_1 \vee M_2 \dots \vee M_m$ . Then  $C$  and  $D$  are called *compatible* if  $n=m$  and  $L_i$  and  $M_i$  are compatible for  $i=1, \dots, n$ .

In fact, a clausal form can be considered as a generalized word. For example, we can consider  $L_1 \vee L_2 \dots \vee L_n$  as a term or literal in the form of  $\vee(L_1, \dots, L_n)$ . Thus a finite set of compatible clausal forms has a supremum by generalizing the following theorem.

**Theorem (Plotkin).** Every non-empty, finite set  $K=\{W_1, \dots, W_n\}$  of words has a supremum iff all words in the set are compatible. Furthermore, if  $W$  is a supremum of  $K$  and  $\mu_i, i=1, \dots, n$  are the substitutions such that  $W\mu_i=W_i, i=1, \dots, n$ , then

- 1) If  $t$  is a term in  $W$  then  $t$  is a supremum of  $\{t\mu_1, \dots, t\mu_n\}$ .
- 2) If  $x, y$  are variables in  $W$  and  $x\mu_i=y\mu_i, i=1, \dots, n$ , then  $x=y$ .

**Theorem.** Every non-empty, finite set  $K=\{W_1, \dots, W_n\}$  of clausal forms has a supremum iff all clausal forms in the set are compatible. Furthermore, if  $W$  is a supremum of  $K$  and  $\mu_i, i=1, \dots, n$  are the substitutions such that  $W\mu_i=W_i, i=1, \dots, n$ , then

- 1) If  $t$  is a word in  $W$  then  $t$  is a supremum of  $\{t\mu_1, \dots, t\mu_n\}$ .
- 2) If  $x, y$  are variables in  $W$  and  $x\mu_i=y\mu_i, i=1, \dots, n$ , then  $x=y$ .

**Example.** Given  $W_1=P(x, g(x)) \vee Q(x, g(f(x)))$  and  $W_2=P(g(y), g(y)) \vee Q(g(y), z)$ . Then the supremum  $W$  of  $W_1$  and  $W_2$  is  $P(x, g(u)) \vee Q(x, z)$ .

## 5 Completeness of V-operators

Let  $R$  and  $C_1$  be given clauses and  $L_1 \in C_1$ , we want to find  $C_2$  such that  $R$  is a resolvent of  $C_1$  and  $C_2$  with  $L_1$  a literal to be resolved on. Let  $C_1'$  be the clausal form which is the disjunction of the literals in  $C_1 - L_1$  in some order. If  $C_2$  is the disjoint union of  $L_2, C_2'$  (clausal form) and there are  $\theta_1, \theta_2$  such that  $L_1\theta_1 = \sim L_2\theta_2$ , then a necessary condition to have  $R$  being a resolvent is  $R \supseteq C_1'\theta_1$  and  $R \supseteq C_2'\theta_2$ . Furthermore,  $C_1'\theta_1$  and  $C_2'\theta_2$  can have non-empty intersection. Thus  $C_2\theta_2 = \{\sim L_1\theta_1\} \cup R'$  where  $R'$  is a subset of  $R$  which contains  $R - C_1'\theta_1$ . Thus for fixed  $R', C_1'$  and  $\theta_1$ , we consider a clausal form  $C' \approx R'$  and we try to find generalizations of the clausal forms of  $\sim L_1\theta_1 \vee C'$ . We can formulate our algorithm in the following way:

### Algorithm for V-operator

- (1) For given clause  $C_1$  and  $R$  where  $L_1 \in C_1$ , consider a clausal form  $C_1'$  of  $C_1 - L_1$ .
- (2) Find a substitution  $\theta_1$  such that  $R \supseteq C_1'\theta_1$ .
- (3) Consider a subset  $R'$  of  $R$  such that  $R' \supseteq R - C_1'\theta_1$ .
- (4) Find a clausal form  $C'$  such that  $C' \approx R'$ .
- (5) Find a  $C_2'$  and  $L_2$  such that  $(L_2 \vee C_2')\theta_2 = \sim L_1\theta_1 \vee C'$ .
- (6)  $C_2' \vee L_2$  is a clausal form of  $C_2$ .

Notice that we can write  $C'$  in different ways, namely any literal in  $R'$  can appear more than one time. Suppose  $R'$  has  $n$  elements and suppose we want to find  $C_2$  with  $k$  literals where  $k$  is between  $n$  and a given  $m$ , i.e.  $m \geq k \geq n$ . Then we can write  $C'$  in  $m$  literals in different ways by letting different literals repeat a different number of times. This algorithm is *complete* in the sense that for a given  $m, m \geq n$ , we can find all  $C_2$  with  $k$  literals such that  $m \geq k \geq n$ .

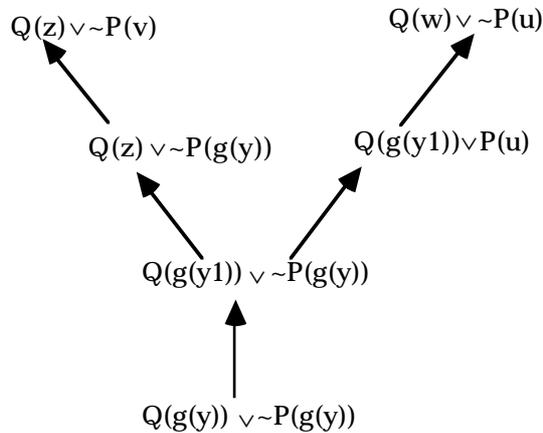
Among all the generalizations of  $\sim L_1\theta_1 \vee C'$  is  $\sim L_1\theta_1 \vee C'$  the minimum. In chapter 7 we tell how we use term partitions to find all the generalizations of clausal forms. This makes step (5) more concrete. If we want to find generalizations systematically, we should find them by every step

taking minimum proper generalization. This can also be done by considering the corresponding least higher term partitions. We are also going to explain it in chapter 7.

**Example.** Let  $C_1 = \{P(x), \sim Q(f(x))\}$ ,  $L_1 = P(x)$  and  $R = \{Q(g(y)), \sim Q(f(g(y)))\}$ . There is only one  $\theta_1$  possible, namely  $\{x/g(y)\}$ . We demonstrate only the situation where  $C' \approx R - C_1 \theta_1 = \{Q(g(y))\}$  and  $C'$  does not contain repeated literals, i.e.

$$C' \vee \sim L_1 \theta = Q(g(y)) \vee \sim P(g(y))$$

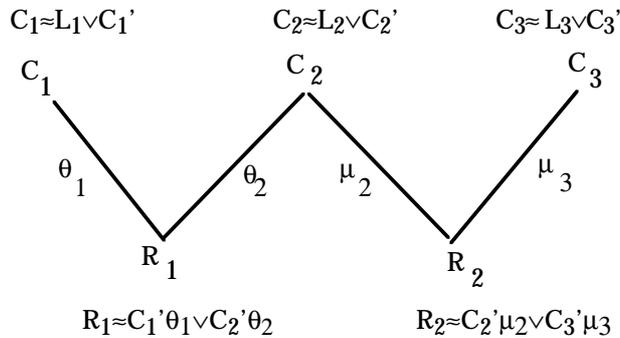
The following diagram tells what all the  $C_2$  are in this situation. For every step we take a minimal proper generalization.



It is interesting to know the order relation between different  $C_2$  which are induced by different  $\theta_1$ . In [6] it is proved by term partitions that different  $C_2$  induced by different  $\theta_1$  are essential incomparable.

## 6 Completeness of W-operators

The following diagram and the clauses in this diagram have the same meaning as in 2.3.



Notice that  $C_2'$  is higher than  $C_2' \theta_2$  and  $C_2' \mu_2$  in the order relation defined by substitutions. The last two compatible clausal forms represent subsets of  $R_1$  and  $R_2$ , respectively. If we can find two such clausal forms, we can construct the supremum of these two clauses. All  $C_2'$  which induce the same  $C_2' \theta_2$  and  $C_2' \mu_2$  are higher (w.r.t.  $\geq$ ) than this supremum. We summarize these facts as follows.

- 1)  $C_2' \theta_2$  and  $C_2' \mu_2$  are compatible because  $C_2'$  is a generalization of both clauses.
- 2) Given  $R_1$  and  $R_2$  and let  $C_2'$  be a clausal form which satisfies the condition that  $C_2' \theta_2$  and  $C_2' \mu_2$  represent subsets of  $R_1$  and  $R_2$ , respectively. If we use  $S$  to denote the supremum of  $C_2' \theta_2$  and  $C_2' \mu_2$ , then  $C_2' \geq S$ .

- 2) For every choice of  $C_2'\theta_2$ , a  $C_1$  can be considered as a generalization of  $\sim L_2\theta_2\vee C'$  where  $C'$  is a clausal form containing  $R_1-C_2'\theta_2$ . Similarly for  $C_3$ . This part is similar to (3)-(6) of V-operator in chapter 5.

The construction of  $C_1$ ,  $C_2$  and  $C_3$  is as follows. We can prove that this construction is complete, i.e. it finds all the  $C_2$  which has less than and equal to  $n+1$  elements for given  $n$ .

### Construction of $C_1$ , $C_2$ and $C_3$ :

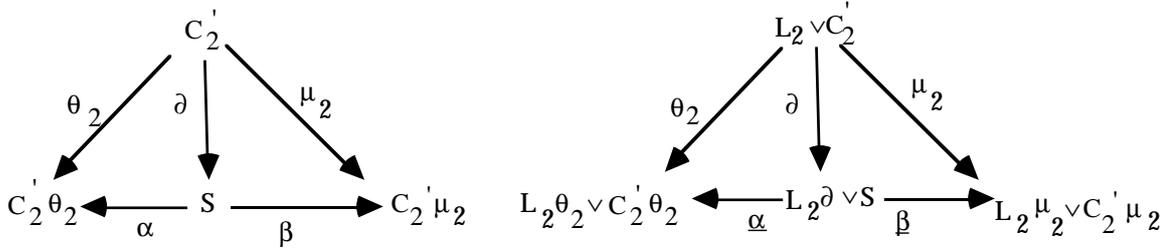
- (1) Given clauses  $R_1$  and  $R_2$ . Consider a non-empty subset of  $R_1$  and a non-empty subset of  $R_2$  such that these two subsets can be represented by the compatible clausal forms  

$$C=M_1\vee M_2\dots\vee M_n, D=N_1\vee N_2\dots\vee N_n.$$
- (2) Find the supremum  $S$  of  $C$  and  $D$  w.r.t. the order relation  $\geq$ . Let  $\alpha$  and  $\beta$  be the substitution from  $S$  to  $C$  and  $D$ , respectively.
- (3) Define a literal  $L_2$ .
- (4) Let  $M=L_2\vee S$ . Find all generalizations of  $M$ . Every generalization is a  $C_2$ .
- (5) Let  $R_1'$  be a subset of  $R_1$  containing  $R_1-M_1\vee M_2\dots\vee M_n$ . Let  $\underline{\alpha}$  be an extension of  $\alpha$  on variables of  $M$ .  $\underline{\alpha}$  may be defined on some variables in  $L_2$  which are not  $S$ . The generalizations defined on clausal forms representing  $R_1'\cup\{\sim L_2\underline{\alpha}\}$  give different  $C_1$ . We can also consider a limit for the number of elements in  $C_1$  just as we do with V-operator. Similarly, we can find  $C_3$ .
- (6) We can let  $C'$  change. A triple  $(C_1, C_2, C_3)$  is a combination of  $C_2$  from (4) and  $C_1, C_3$  from (5).
- (7) If there are permutations of  $(1,2,\dots,n)$  which is represented by  $(i_1,\dots,i_n)$  and  $N_{ij}$  and  $M_i$  are still compatible for  $i=1,\dots,n$ , then we can consider a new  $M$  which is the supremum of  

$$M_1\vee M_2\dots\vee M_n, N_{i_1}\vee N_{i_2}\dots\vee N_{i_n}.$$
Then we can again find the supremum  $S$  of these two clausal forms and the generalizations of  $S$ . Repeat (3), (4), we can again find different  $C_2$ .
- (8) We can let different literals in  $C$  and  $D$  repeat and thus we have new  $C$  and  $D$ . Do (2)-(7).
- (9) If we let the subsets of  $R_1$  and  $R_2$  in (1) change, then we find all  $(C_1, C_2, C_3)$  with  $C_2$  containing  $k$  literals where  $k\leq n+1$ .

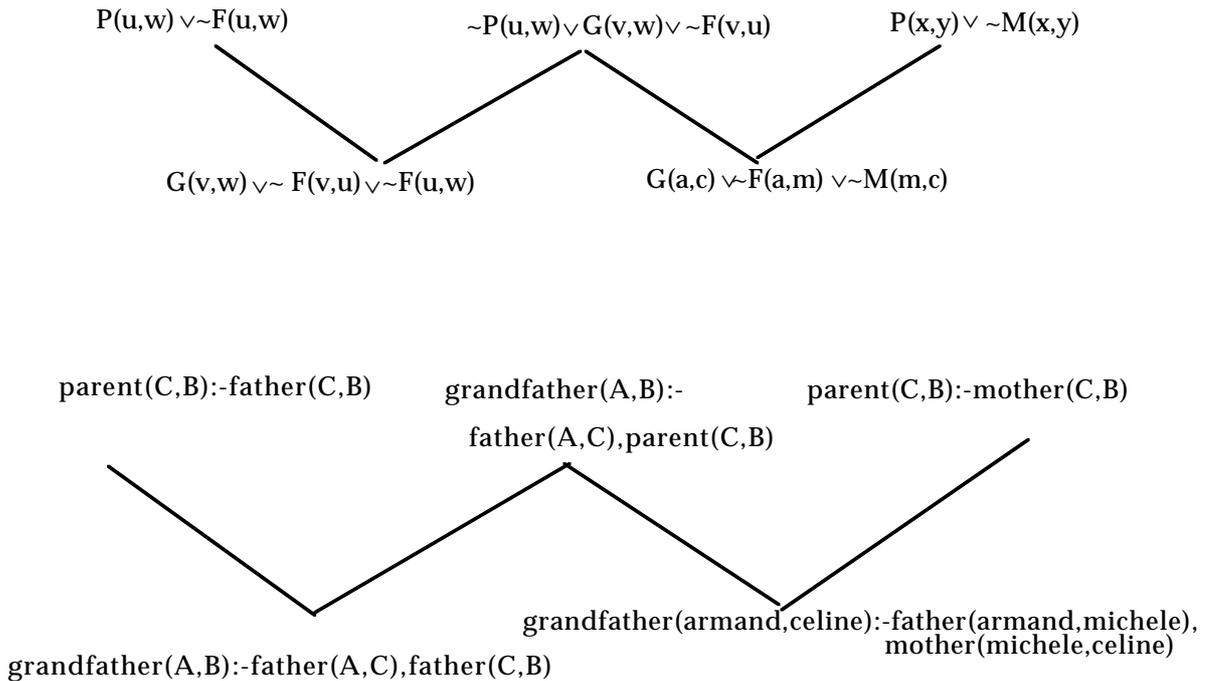
**Theorem.** A W-operator defined by the above algorithm is complete.

**Proof.** Given  $C_1=L_1\vee C_1'$ ,  $C_2=L_2\vee C_2'$  and  $C_3=L_3\vee C_3'$ . Suppose also  $L_1\theta_1=\sim L_2\theta_2$  and  $L_2\mu_2=\sim L_3\mu_3$  and that  $R_1=C_1'\theta_1\vee C_2'\theta_2$ ,  $R_2=C_2'\mu_2\vee C_3'\mu_3$ . From the compatibility between  $C_2'\theta_2$  and  $C_2'\mu_2$  we can construct the supremum  $S$  of these two clausal forms. Let  $\alpha$  and  $\beta$  be the substitutions from  $S$  to  $C_2'\theta_2$  and  $C_2'\mu_2$ , respectively. Then there is a substitution  $\partial$  from  $C_2'$  to  $S$  such that  $\partial\alpha=\theta_2$  and  $\partial\beta=\mu_2$  because  $S$  is the supremum. We can assume  $S$  and  $C_2'$  have different variable names. Let us consider the clausal form  $L_2\partial\vee S$ . It is easy to see that  $L_2\vee C_2'$  is a generalization of  $L_2\partial\vee S$  because  $L_2\partial\vee S=(L_2\vee C_2')\partial$ . We can consider  $\partial$  as a substitution from all variables in  $L_2\vee C_2'$ . We want now to find extensions  $\underline{\alpha}$ ,  $\underline{\beta}$  of  $\alpha$ ,  $\beta$ , respectively such that  $\partial\underline{\alpha}=\theta_2$  and  $\partial\underline{\beta}=\mu_2$ . We need to prove this because we can then say  $L_2\partial=L_2$  and  $(L_2\vee C_2')\theta_2=(L_2\vee C_2')\partial\underline{\alpha}=(L_2\vee S)\underline{\alpha}$ ,  $(L_2\vee C_2')\mu_2=(L_2\vee C_2')\partial\underline{\beta}=(L_2\vee S)\underline{\beta}$ . Thus step (5) above makes sense.



If we consider a variable  $x$  in  $L_2$  which are not in  $C_2'$ , then  $L_2\delta$  has made no change in this variable. We can define  $x\alpha = x\theta_2$ . Thus  $\delta\alpha = \theta_2$ . Thus  $L_2\delta v C_2'$  can be achieved by finding a generalization on  $L_2 v S$ .  $C_1$  can be achieved by considering  $R_1' \cup \{\sim L_2\alpha\} = R_1' \cup \{\sim L_2\delta\alpha\}$  where  $R_1' \approx R_1 - C_2'\theta_2$ . Similarly for  $C_3$ .

**Example.** Let  $R_1 \approx G(v,w) \vee \sim F(v,u) \vee \sim F(u,w)$  and  $R_2 \approx G(a,c) \vee \sim F(a,m) \vee \sim M(m,c)$ . Consider  $G(v,w) \vee \sim F(v,u)$  and  $G(a,c) \vee \sim F(a,m)$  as the compatible clausal forms, representing subsets of  $R_1$  and  $R_2$ , respectively. The supremum  $S$  of these two clausal forms is  $G(v,w) \vee \sim F(v,u)$  and  $\alpha = \{ \}$  and  $\beta = \{v/a, w/c, u/m\}$ . Let us consider a literal  $L_2 = \sim P(u,w)$ . The minimum generalization of  $L_2 v S = \sim P(u,w) \vee G(v,w) \vee \sim F(v,u)$  is itself. Let  $\underline{\alpha} = \alpha$  and  $\underline{\beta} = \beta$ . Consider  $R_1' = \{ \sim F(u,w) \}$  and  $R_2' \approx \{ \sim M(m,c) \}$ . One possibility of  $C_1$  is  $P(u,w) \vee \sim F(u,w)$ . A possibility of  $C_3$  is  $P(x,y) \vee \sim M(x,y)$  which is a generalization of  $P(m,c) \vee \sim M(m,c)$ . The relations between different clausal forms can be seen in the first W-diagram below. If we translate the first W-diagram into more concrete Prolog clauses of the example[11] related to the family situation, then we have the second the W-diagram.



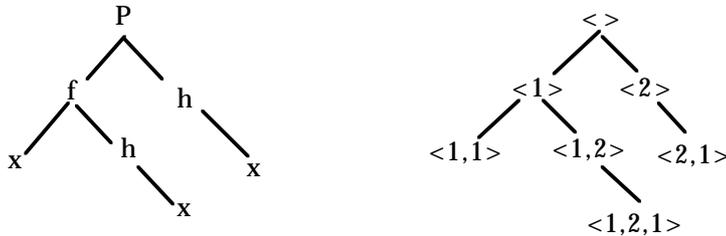
## 7 A survey of using term partitions in inversting resolutions

We have talked about the V- and W-operators on the first and second levels. The third level uses codings of trees (or clauses) so that we can discuss the questions more concretely and clearly.

Integer codings and variable codings[7] have been used for this purpose. Term partitions[3,6] come from integer coding and flattening[11,12] has to do with variable coding. These two ways are used for finding generalizations of clauses and thus inverse resolutions. We are going to give a survey of integer coding and term partition for finding generalizations here.

## 7.1 Term occurrences

A clause like  $P(f(x,h(x)),h(x))$  can be considered as a labeled tree (left below). If we only consider the structure of the tree, then we have an unlabeled tree. The nodes of an unlabeled tree can be coded by natural number sequences (right below).

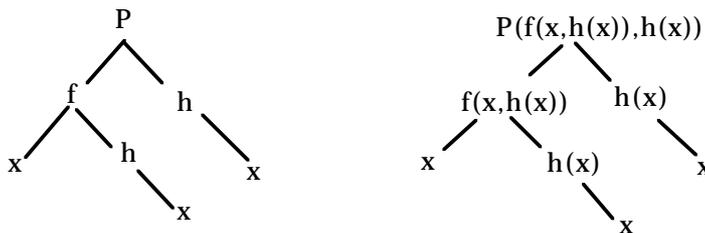


The labeled tree can be expressed by a function from the unlabeled tree:

$$\{(P, \langle \rangle), (f, \langle 1 \rangle), (h, \langle 2 \rangle), (x, \langle 1, 1 \rangle), (h, \langle 1, 2 \rangle), (x, \langle 1, 2, 1 \rangle), (x, \langle 2, 1 \rangle)\}.$$

The second coordinate of a pair is an original of the function and the first coordinate (*label*) is the image of the second coordinate.

A labeled tree defined by a clause  $C$  induces a labeled tree with trees as labels. These tree labels come from the subtrees of the original tree. For example, the right tree below is induced by the left one. We use  $f(x,h(x))$ , etc. to denote such labels.



We use the set of pairs which contains elements like  $(f(x,h(x)), \langle 1 \rangle)$ ,  $(h(x), \langle 1, 2 \rangle)$  to code this tree. Such a pair is called a *term occurrence* if the first coordinate is a term like  $f(x,h(x))$ . This coding can be extended to clauses with more than one literal, e.g.  $(y, \langle 1, 2, 1 \rangle)$  is a term occurrence of  $C = P(x, f(y, z)) \vee \sim Q(x)$ . As we know two subtrees of a tree have either no nodes in common or one is a subtree of the other. This property can be translated directly into the coding system. For example,  $f(y, z)$  in  $C = P(g(x, f(y, z)))$  has position  $\langle 1, 2 \rangle$  and  $y$  has position  $\langle 1, 2, 1 \rangle$ . The sequence  $\langle 1, 2 \rangle$  is a *subsequence* (first part) of  $\langle 1, 2, 1 \rangle$ . This subsequence relationship characterizes the subtree relationship. If  $p$  is a subsequence of  $q$ , then we use  $q-p$  to denote the rest of  $q$  after removing  $p$ . For example,  $\langle 1, 2, 1 \rangle - \langle 1, 2 \rangle = \langle 1 \rangle$ . Consider two term occurrences  $(t, p)$  and  $(s, q)$ , if  $q$  contains  $p$  as a subsequence then  $q-p$  denotes the relative position of the subterm occurrence  $(s, q)$  in  $(t, p)$ .

## 2.2 Term partitions and generalizations

Finding the generalization  $C_1 = P(f(x, z), z)$  of the clause  $C = P(f(x, h(x)), h(x))$  can be considered as changing two subtrees of  $C$  by  $z$ . The subtrees of  $C$  which correspond to variables in  $C_1$  are denoted by the following set of term occurrences

$$T = \{(x, \langle 1, 1 \rangle), (h(x), \langle 1, 2 \rangle), (h(x), \langle 3 \rangle)\}$$

T can be divided into two blocks of disjoint subsets which correspond with the variables x and z in  $C_1$ :

$$B_x = \{(x, \langle 1, 1 \rangle)\}$$

$$B_z = \{(h(x), \langle 1, 2 \rangle), (h(x), \langle 3 \rangle)\}$$

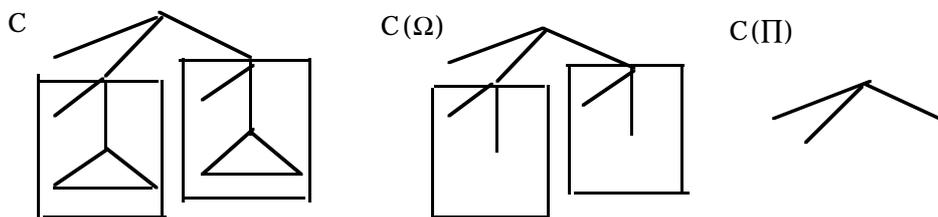
This partition of blocks is called a *term partition* on C. Generally, we can define term partitions with two properties. Firstly, the chosen set T should satisfy some *minimal* condition, i.e. no second coordinates of two pairs in T have subsequence relationship. Indeed, if we intend to change one subtree (s,p) to a variable, then we should not consider a subtree (t,q) of (s,p) also as a candidate for a variable. Secondly, the labels in the same block should be the same because they are the images of the same variable of a substitution from  $C_1$ .

Let C be a given clause and  $C_1$  be a clause defined by the inverse of a substitution. Then there is a term partition defined on C. On the other hand, if there is a term partition on C, then we can define a clause  $C_1$  which is more general than C. A term partition on C induces many  $C_1$ , but these differ only by variable names. We can find all generalizations of C by considering all the term partitions on C. Using partitions have the following advantage: a term partition is defined on a subset of term occurrences on C and the structure of  $C_1$  is known without having to construct  $C_1$  concretely.

### 7.3 Order relations on term partitions

We have seen that the set of all term partitions determine all the generalizations of a clause. For a V-operator we need to find the generalizations of clauses with the form of  $\sim L_1 \theta_1 \vee C'$ . For a W-operator we need to find the generalizations of the supremum of two clausal forms. Thus using term partitions solves also the problems of V- and W-operators. Finding all the term partitions is very much work. We have to do our work efficiently. If we have a clause  $C_1$  which is a generalization of a clause C, we need perhaps to find a clause  $C_2$  which is one step more generalized, i.e.  $C_2$  is a minimal proper generalization of  $C_1$ . Even more, if we know  $C_2$  is a generalization of C and it is too general, we want then to find a minimal generalization  $C_1$  of  $C_2$  such that  $C_2 \geq C_1$ . If we consider the term partitions on C instead of the actual generalizations of C, then we need to define a compatible order relation on term partitions. The following definitions of order relations on term partitions and related theorems can be found in [4].

Let C be a given clause and let  $\Pi, \Omega$  be two term partitions on C. In the following diagram one block of  $\Pi$  is illustrated by two rectangles and one block of  $\Omega$  is illustrated by two triangles. Notice that the relative positions of the two triangles in rectangles are the same. Notice also that  $C(\Pi)$  is more general than  $C(\Omega)$ . The order relation in the term partitions can be defined by comparing the relative positions of term occurrences in the blocks of the two term partitions.



Let us consider  $C = P(x, f(x, g(y, z)), f(x, g(y, z)))$  as a concrete example. The rectangles correspond with the block B in  $\Pi$ :

$$B = \{(f(x, g(y, z)), \langle 2 \rangle), (f(x, g(y, z)), \langle 3 \rangle)\}$$

and the triangles correspond with the block D in  $\Omega$ :

$$D = \{(g(y, z), \langle 2, 2 \rangle), (g(y, z), \langle 3, 2 \rangle)\}$$

The relative position of  $\langle 2,2 \rangle$  in  $\langle 2 \rangle$  is  $\langle 2 \rangle$  and so is the relative position of  $\langle 3,2 \rangle$  in  $\langle 3 \rangle$ . The definition of  $\Pi \geq \Omega$  can be formulated as follows:

**Definition.** Let  $C$  be given clause and  $\Pi, \Omega$  be partitions defined on  $T, S$ , subsets of term occurrences of  $C$ , respectively. We say  $\Pi$  is *higher than or equal to*  $\Omega$ , denoted by  $\Pi \geq \Omega$ , if

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

Condition 2) can be interpreted as:  $D$  contains the set of all subterm occurrences of occurrences in  $B$  at position  $q-p$ .

**Remark.** To prove the compatibility between the order relation in the term partitions on  $C$  and in the generalizations of the clause  $C$ , we need the concept of *consistent term mappings*. [3,6]. This concept is a generalization of the concept of substitutions and inverse substitutions. We can also apply this concept of mappings and term partitions to  $V$ -operators. For a fixed  $\theta_1$  we need to consider the term partitions on the clausal form  $\sim L_1 \theta_1 \vee C'$  to find  $C_2$  as we have seen before. If  $C_1 = L_1$ , then it is proved in [6] that different  $C_2$  (under some restriction) constructed via different  $\theta_1$  (under some restriction) are incomparable. Thus the partial ordering on different  $\sim L_1 \theta_1 \vee C'$  for different  $\theta_1$  have essentially nothing to do with each other.

Consider the set of all partitions on a clause  $C$ . We can define the *least higher partition* of a partition on this set in an analogous way as we define minimal proper generalization. The following three least higher partitions correspond with minimal proper generalizations given in 4.1. Some examples for corresponding generalizations have also been given in 4.1.

#### Least higher partition of the first kind

The following theorem can be translated to the language of clauses as follows: by replacing some constants in  $C$  by a new variable we obtain a minimal generalization of  $C$ .

**Theorem.** Let  $B = \{(a,q_1), \dots, (a,q_n)\} \neq \emptyset$  be a set of  $a$ -occurrences in a clause  $C$  and  $\Omega$  be the trivial partition defined by variables in  $C$ . Then the partition  $\Pi$  defined by  $\Pi = \Omega \cup \{B\}$  is least higher than  $\Omega$ .

#### Least higher partition of the second kind

Minimal generalization can be obtained by changing some occurrences of a variable to a new variable name.

**Theorem.** Let  $\Omega$  be the trivial partition on  $C$  and  $D$  be the block in  $\Omega$  defining a variable  $v$ . If  $B_1$  is a proper subset of  $D$  and  $B_2 = D - B_1$ , then  $\Pi = (\Omega - \{D\}) \cup \{B_1, B_2\}$  is least higher than  $\Omega$ .

#### Least higher partition of the third kind

We can construct a minimal generalization of a clause by replacing some kind of compound term occurrences by a new variable.

**Theorem.** Let  $\Omega$  be the trivial partition on  $C$  and  $t = f(v_1, \dots, v_m)$  such that  $v_i \neq v_j$  if  $i \neq j$ . Let  $B = \{(t,p_1), \dots, (t,p_n)\}$  be the set of all  $t$ -occurrences and let  $D_1, \dots, D_m$  be the blocks defined by  $v_1, \dots, v_m$ . If every  $v_i$ -occurrence is always a subterm occurrence of such a  $(t,p_i)$ , then the partition  $\Pi = (\Omega - \{D_1, \dots, D_m\}) \cup B$  is least higher than  $\Omega$ .

**Theorem.** Let  $C$  be a given clause and  $\Omega, \Pi$  be partitions on  $C$  such that  $\Pi > \Omega$ . We can find a finite sequence of partitions  $\Omega_0, \Omega_1, \dots, \Omega_n$  such that  $\Omega_0 = \Omega$ ,  $\Omega_n = \Pi$  and  $\Omega_{i+1}$  is least higher than  $\Omega_i$  for every  $i$ .

**Example.** Let  $C, \Omega$  and  $\Pi$  be given as below, we want to construct a sequence of partitions as in the theorem. For simplicity, we write  $p$  instead of  $(t,p)$ .

$C=P(f(g(u,a,a),u), g(u,a,a))$

$\Pi: E_1=\{<1,1>, <2>\}, E_2=\{<1,2>\}$ .

$\Omega_0=\Omega: D_1=\{<1,1,1>, <1,2>, <2,1>\}, D_a=\{<1,1,2>, <1,1,3>, <2,2>, <2,3>\}$ . We consider first  $E_1$ .

$\Omega_1$ : Because  $E_1 \geq D_a$ , let  $B_1=\{<1,1,2>, <2,2>\}$ . Then  $\Omega_1=\Omega \cup \{B_1\}=\{B_1, D_1\}$  from (1). The constant block of  $\Omega_1$  is  $B_a=D_a-B_1=\{<1,1,3>, <2,3>\}$ .

$\Omega_2$ : Because  $E_1 \geq B_a$ , let  $B_2=\{<1,1,3>, <2,3>\}$ . Then  $\Omega_2=\Omega_1 \cup \{B_2\}=\{B_1, B_2, D_1\}$  from (1).

$\Omega_3$ : Compare  $E_1$  with  $D_1$ , we have  $D_1=B_3 \cup B_4$  where  $B_3=\{<1,1,1>, <2,1>\}$  and  $B_4=D_1-B_3=\{<1,2>\}$ . Thus  $\Omega_3=(\Omega_2-\{D_1\}) \cup \{B_3, B_4\}=\{B_1, B_2, B_3, B_4\}$  from (2).

$\Omega_4$ :  $E_1 \geq B_1, B_2, B_3$ . The corresponding  $r_1, r_2, r_3$  are  $<2>, <3>, <1>$ .

Because  $r_1'=r_2'=r_3'=<>$ , we have constructed a  $B_5=\{<1,1>, <2>\}$  which is  $E_1$ .

Thus  $\Omega_4=\Omega_3-\{B_1, B_2, B_3\} \cup \{B_5\}=\{B_4, B_5\}=\Pi$ .

## 8 Conclusion and future work

The problems with inverse resolutions or resolutions have to do with clauses expressed in sets of literals. These sets are more difficult to handle and thus we use clausal forms of clauses. The problems are first translated to clausal forms and we can then discuss clausal forms instead of sets. Both V- and W-operators have to begin with finding generalizations of some clausal forms. The clausal form which we have to consider in a V-operator is defined by two clauses  $C_1, C$  and a substitution from  $C_1$ . For the W-operator we need to consider the supremum of two clausal forms which represent some subclauses of given clauses  $R_1$  and  $R_2$ .

From clauses to clausal forms are interpreted by us as the translation from first level to second level. The third level has to do with the coding we choose to formulate and discuss the problems. We can use integer coding to code the term occurrence in a clausal form. Term partition is used in this coding to find generalizations. We can define a partial order relation in the set of all term partitions on a clausal form. With this order relation we can talk about minimal proper generalizations and chain of minimal proper generalizations. The fourth level over implementation is not done in this article.

The clausal forms  $P(x,y) \vee P(u,v)$  and  $P(x,y) \vee P(x,y)$  are generalization of  $C=P(a,a) \vee P(a,a)$ . The first represents  $D_1=\{P(x,y), P(u,v)\}$  and the second represents  $D_2=\{P(x,y)\}$ . In fact  $D_2$  is the reduced clause[9] of  $D_1$ . We are satisfied if we can find one of the two, especially  $D_2$ . Suppose we know how to find generalizations of  $C$  which represent reduced clauses, then we have much less work. How to find clausal forms of reduced clauses should be investigated in the future.

## Acknowledgement

I like to thank J. W. Nienhuys for his comments. His suggestion to look at problems on a higher level instead of the codings makes things clearer and more structured.

## References

- 1 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.
- 2 Stephen Muggleton. Inductive Logic Programming. First Conference on Algorithmic Learning Theory, Ohmsha, Tokyo, October 1990.
- 3 Shan-Hwei Nienhuys-Cheng. Consequent Functions and Inverse Resolutions. Report Eur-CS-90-03, Erasmus University, Rotterdam, Netherlands, May 1990.
- 4 Shan-Hwei Nienhuys-Cheng. Term Partitions and Minimal Generalizations of Clauses. Report, Eur-CS-91-01, Erasmus University, Rotterdam, Netherlands, January, 1991.

- 5 Shan-Hwei Nienhuys-Cheng. Flattening, Generalization of Clauses and Absorption Algorithms. Report, Conference Benelearn, May 1991, University of Amsterdam, Amsterdam, Netherlands.
- 6 Shan-Hwei Nienhuys-Cheng & Peter Flach. Consistent Term Mappings, Term partition and Inverse Resolutions, In: Machine Learning-EWSL-91. European Working Session on Learning, Porto, Portugal, March, 1991, Proceedings. Y. Kodratoff(Ed). Lecture notes in Artificial Intelligence 482, Springer Verlag.
- 7 Shan-Hwei Nienhuys-Cheng. Codings of Trees and Generalizations by Flattening and Term Partitions, Preprint, presented in BISFAI 91, June 1991, Israel.
- 8 Shan-Hwei Nienhuys-Cheng. A survey of Generalization Algorithms and Inverse Resolutions. (to appear in the Proceedings of IJCAI-91). Workshop 8, IJCAI-91, Sidney, Australia: Evaluation and Changing of Representation in Machine Learning, August,1991.
- 9 Gordon D. Plotkin. A Note on Inductive Generalisation. *Machine Intelligence 5*, B. Meltzer & D. Michie (eds.), Edinburgh University Press, 1970.
- 10 Céline Rouveirol & Jean-Francois Puget. A Simple Solution for Inverting Resolution. EWSL-89, Pitman, London, pp. 201-210, 1989.
- 11 Céline Rouveirol & Jean-Francois Puget. Beyond Inversion of Resolution. Proceedings of the Fifth International Conference in Machine learning. Kaufman, 1990.
- 12 John C. Reynolds. Transformational Systems and the Algebraic Structure of Atomic Formulas. *Machine Intelligence 5*, B. Meltzer & D. Michie (eds.), Edinburgh University Press, 1970.
- 13 J. A. Robinson. A Machine-Oriented Logic based on the Resolution Principle. Journal of the Association for Computing Machinery, Vol 12, No. 1, 1965
- 14 R. Wirth. Learning by Failure to Prove. Proceedings of the third European Working Session on Learning (EWSL 88). Pitman, 1988.