Symmetries in Logic Programs

Jinzhao Wu

MPI–I–1999–2-005 April 1999

Author's Address

Jinzhao Wu
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
wu@mpi-sb.mpg.de
http://www.mpi-sb.mpg.de/~wu

Publication Notes

This paper has been submitted for publication.

Acknowledgements

I am grateful to Prof. Dr. Harald Ganzinger for his valuable suggestions, which, in particular, lead me to deeply consider various philosophies in logic programming. I am also grateful to Dr. Jürgen Stuber and Dr. Viorica Sofronie-Stokkermans, who went through the paper and provided me with useful information. Finally, thanks go to Ms. Brigitta Hansen, whose help made the paper more readable.

Abstract

We investigate the structures and above all, the applications of a class of symmetric groups induced by logic programs. After establishing the relationships between minimal models of logic programs and their simplified forms, and models of their completions, we show that in general when deriving negative information, we can apply the CWA, the GCWA, and the completion procedure directly from some simplified forms of the original logic programs. The least models and the results of SLD-resolution stay invariant for definite logic programs and their simplified forms. The results of SLDNF-resolution, the standard or perfect models stay invariant for hierarchical, stratified logic programs and some of their simplified forms, respectively. We introduce a new proposal to derive negative information termed OCWA, as well as the new concepts of quasi-definite, quasi-hierarchical and quasi-stratified logic programs. We also propose semantics for them.

1 Introduction

In logic programming, one usually makes use of some orderings to develop various notions and ideas close to the common-sense intended meaning. This is in fact taking some asymmetries into account. For instance, in $\neg p \rightarrow q$, people often bear in mind that q depends upon p, and therefore they haven't the same status. This treatment leads to many works including stratified logic programs[1, 27], the SLDNF-resolution[8], standard and perfect model semantics[1, 21].

On the other hand, like in any mathematical objects, symmetries in logic programs are also crucial and worthwhile to further explore. We shouldn't neglect their actions on the corresponding logic programs. For example, if we fail to infer $\neg p$ from $p \vee q$ under the GCWA[20], it seems natural not to infer $\neg q$ either from $p \vee q$, for p and q here have the same status. Even in $\neg p \rightarrow q$, it is not useless to consider the symmetry of p and q in $p \vee q$. By deleting $\neg p$, we obtain $\rightarrow q$. The completions of $\neg p \rightarrow q$ and $\rightarrow q$ are logically equivalent. The minimal model of $\rightarrow q$ is also a minimal model of $\neg p \rightarrow q$, and all minimal models of $\neg p \rightarrow q$ can be obtained by applying the permutations representing the symmetry to that of $\rightarrow q$. The standard models[1] of $\neg p \rightarrow q$ and $\rightarrow q$ keep invariant. These considerations sometimes not only help simplify the computation procedures, but increase the computational power and result in new concepts. In $p \vee q$ or $\neg p \rightarrow q$, we see that if we force one of p and q to be negative, it doesn't lead to any inconsistency.

This paper represents some initial work in this regard. We are concerned with a class of syntactic symmetries in logic programs, which can be represented by symmetric groups on predicate symbols. Besides making clear their structures, we try to see the behaviors of logic programs under these symmetries.

As a matter of fact, there exist a number of symmetric group structures in logic programs(See the conclusion part). We believe that they play a role in logic programming. Another encouraging phenomenon is that some permutations preserve many properties of logic programs. We have the following simple facts: If σ is a permutation on predicate symbols, and P a logic program, then I is model of P iff $\sigma(I)$ is a model of $\sigma(P)$; I is a model of $Comp(P)$ iff $\sigma(I)$ is a model of $Comp(\sigma(P))$; and P is stratified iff $\sigma(P)$ is stratified(It is straightforward to understand what $\sigma(I)$ and $\sigma(P)$ are from the point of view of renaming).

We know that two major topics in logic programming are deriving negative information and developing various semantics[2, 25]. We focus on these two issues, too.

There are generally three popular proposals to tackle negative information. The CWA was introduced by Reiter to efficiently represent completely specified world[22]. If it is in force, the negative facts can be inferred implicitly. However, CWA may lead to inconsistency even though the original logic programs are consistent. So sometimes this formalism is both inappropriate and unsafe. The GCWA was thus proposed by Minker, which is consistency-preserving[20]. The negation as failure rule based on program completions was first studied in detail by Clark [8]. Its two nice properties are that it is semi-decidable and easier to implement. We do not involve ourselves in circumscriptions[18] here.

There are also many works on semantics. In this paper, we do not discuss those using non-classical logics(See [25] and the references therein). We concentrate on the classical first-order logic framework. The least models and SLD-resolution for definite logic programs were first developed by van Emden and Kowalski[26]. Many later works in logic programming are based on these notions. Clark first introduced hierarchical logic programs, and proved the completeness of SLDNF-resolution for this class of logic programs[8]. Stratified logic programs were introduced by Apt, Blair and Walker[1], and independently by Van Gelder[27]. Standard model semantics for stratified logic programs was proposed by Apt, Blair and Walker[1]. It coincides with the perfect model semantics developed by Przymusinski[21].

As is known, the minimal models of logic programs and the models of their completions are quite important in the theory of logic programming. Motivated by them, in this paper we define a class of symmetries induced by logic programs. After making clear the structures, we first see their actions on these two kinds of models. Indeed, all the results are based on the fact that among the elementary facts with the same entry in each such model, there is no more than one whose relation symbol occurs in the same orbit of the symmetries. This is also the main reason why we prefer such symmetries. Then, we investigate the applications to CWA, GCWA, and completion procedure. We concern ourselves with the least model semantics and SLD-resolution for definite logic programs, the completeness result of SLDNF-resolution for hierarchical logic programs, as well as the standard or perfect model semantics for stratified logic programs.

we show that in general, these symmetries in logic programs are redundant and can be removed for the above three proposals and semantics. It may simplify the related computation procedures, and increase the computational powers.

Through considering these symmetries, we present a new proposal to deal with negative information that is indeed a generalization of the GCWA. We also define three classes of logic programs termed quasi-definite, quasi-hierarchical and quasi-stratified logic programs respectively, which are more general than definite, hierarchical and stratified logic programs. Finally, we propose the similar semantics for them.

2 Basic knowledge: symmetric groups and logic programs

In this section, we describe some basic notions and results on which our following discussions are directly based. For those we do not define and use in the paper, the reader may consult [23] and [16].

2.1 Symmetric groups

Let R be a non-empty set. A permutation on R is a bijection from R to R. Let S_R be the set of all permutations on R. S_R forms a group under the operation of function composition. To be convenient, in the following we call any a subgroup of S_R a symmetric group (on R) if R is finite. Every permutation on a finite set is either a cycle or a product of disjoint cycles, and it is also a product of transpositions.

Assume that G is a permutation group on R. We define a relation $\sim$ on R by the rule

$$\text{for } r_1, r_2 \in R, r_1 \sim r_2 \text{ iff there exists } \sigma \in G \text{ such that } \sigma(r_1) = r_2.$$

$\sim$ is an equivalence relation on R. Its equivalence classes are called orbits of G.

Let O be an orbit of G, if O consists of only one element r, we say that r is fixed by G. Otherwise, we say that O is proper. By $Fix(G)$ we denote the set of all elements of R fixed by G.

Suppose that G_1 and G_2 are two permutation groups on R_1 and R_2, respectively. If $R_1 \cap R_2 = \emptyset$, for any $\sigma_1 \in G_1, \sigma_2 \in G_2$, let σ be the following permutation on $R_1 \cup R_2$:

$$\sigma(r) = \sigma_1(r), \text{ if } r \in R_1; \sigma(r) = \sigma_2(r), \text{ if } r \in R_2.$$

By $G_1 \times G_2$ we denote the set of all such permutations. It is a permutation group on $R_1 \cup R_2$, and $\times$ is obviously commutative and associative.

For $i = 1, \cdots, n$, suppose that $R_i \subseteq R$, and G_i is a subgroup of S_{R_i}. Let $(1)_{R-R_i}$ represent the trivial subgroup of S_{R-R_i} consisting only of the identity. For $\sigma_i \in G_i$, we define $\sigma_1 \cdots \sigma_n$ to be $\sigma'_1 \cdots \sigma'_n$, where $\sigma'_i \in G_i \times (1)_{R-R_i}$, such that if $r \in R_i$ then $\sigma'_i(r) = \sigma_i(r)$. By $G_1 \cdots G_n$ we denote the following subset of S_R:

$$G_1 \cdots G_n = \{\sigma_1 \cdots \sigma_n | \sigma_1 \in G_1, \cdots, \sigma_n \in G_n\}.$$

Clearly, if G is a subgroup of S_R and $G_1 \cdots G_n \subseteq G$, then $G_n \cdots G_1 \subseteq G$.

2.2 Logic programs

For a given alphabet, terms, atoms, and formulae of the first-order language over this alphabet are defined as usual. A literal is an atom or the negation of an atom. A term or a formula is said to be ground if no variables occur. An interpretation consists of (1) a domain, namely a non-empty set D; (2) a constant assignment, which maps each constant to an element of D; (3) a function symbol assignment, which maps each n-ary function symbol to a function from D^n to D; (4) a predicate symbol assignment, which maps each n-ary predicate symbol to a function from D^n to $\{true, false\}$.

For the definitions of I being an Herbrand interpretation, a model, a minimal model, or an Herbrand model of a closed formula set, the reader is referred to [7, 15]. We say that a formula F follows from a formula set FS(denoted by $FS \models F$), if the models of FS are models of F.

A (normal) clause is a formula of form

$$(\wedge_{i=1}^{u} A_i) \wedge (\wedge_{j=1}^{v}(\neg B_j)) \rightarrow B_w,$$

where A_i, B_j, B_w are atoms, and all variables are supposed to be universal. We say that the predicate symbols in A_i, B_w occur positively, and the predicate symbols in B_j occur negatively, in this clause. If $v = 0$, we call this clause definite. We call $(\wedge_{i=1}^{u} A_i) \wedge (\wedge_{j=1}^{v}(\neg B_j))$ the body, and B_w the head, of this clause. $\{\neg A_1, \cdots, \neg A_u, B_1, \cdots, B_v, B_w\}$ is called the literal multiset of this clause. It actually represents the disjunction form of the clause without atom ordering. So we also say that each A_i occurs negatively, and each B_j, B_w occurs positively in this literal multiset.

For literals $L_1, \cdots, L_k$, $L_1 \wedge \cdots \wedge L_k \rightarrow$ is called a goal. It is said to be definite if $L_1, \cdots, L_k$ are all positive atoms. We also say that $L_1 \wedge \cdots \wedge L_k$ is the body of this goal.

A (normal logic) program is a finite set of clauses. It is called definite if all its clauses are definite. We note that normal programs are relatively general. In fact, Lloyd and Topor showed that more general programs can be transformed into normal ones. For details, we refer the reader to [17].

Now let P be a program. By the P-definition of a predicate symbol r, we mean the subset of P consisting of all clauses with the heads whose predicate symbols are r. P is called stratified[hierarchical], if there exists a partition

$$P = P_1 \cup \cdots \cup P_m$$

such that (1) P_1 can be empty and $P_i \cap P_j = \emptyset (i \neq j)$; (2) if a predicate symbol occurs positively in the body of a clause in P_i, then its P-definition is contained in $\cup_{j \leq i} P_j$[resp. $\cup_{j<i} P_j$]; (3) if a predicate symbol occurs negatively in a clause in P_i, then its P-definition is contained in $\cup_{j<i} P_j$.

Definite and hierarchical programs are stratified.

Clark[8] introduced the notion of completion of a program to justify the use of the negation as failure rule. Let P be a program. Suppose that r is an n-ary predicate symbol, and x an n-tuple of variables. If the P-definition of r is empty, we say that the formula

$$\forall x(\neg r(x))$$

is the completed P-definition of r. Otherwise, let $\wedge_j L_{ij} \rightarrow r(t_i)$ be all the clauses in the P-definition of r, where each t_i is an n-tuple of terms. Suppose that y_i is the tuple of all variables occurring in t_i. We then call the formula

$$\forall x((\vee_i \exists y_i((\wedge_j L_{ij}) \wedge (x = t_i))) \leftrightarrow r(x))$$

the completed P-definition of r.

We remark that $\forall x$ means $\forall x_1 \cdots \forall x_n$, and $(x = t_i)$ represents $\wedge_{j=1}^{n}(x_j = t_{ij})$ for $x = (x_1, \cdots, x_n)$ and $t_i = (t_{i1}, \cdots, t_{in})$, and $\exists y_i$ means $\exists y_{i1} \cdots \exists y_{im}$ for $y_i = (y_{i1}, \cdots, y_{im})$.

The completion of P, denoted by $Comp(P)$, is the collection of completed P-definitions of predicate symbols (appearing in P) together with the equality theory that consists of some axioms for $=$. For the description of the equality theory, please consult Section 14 of [16]. Here we just point out that it is in fact independent of P.

2.3 Notations

Suppose that A is the given alphabet, and R its predicate symbol set. For $\sigma \in S_R$, let $\sigma(A)$ be the same as A except that the predicate symbol set is $\{\sigma(r) \mid r \in R\}$, where the arity of $\sigma(r)$ equals that of r. In the following, we no longer declare the first-order language is over which alphabet, for it is not hard to recognize.

Let R be the set of all predicate symbols of the underlying first-order language, and $\sigma \in S_R$. For a formula F, We define $\sigma(F)$ to be the formula obtained from F by replacing any predicate symbol r in F by $\sigma(r)$. For a formula set FS, let $\sigma(FS) = \{\sigma(F) \mid F \in FS\}$. For a literal multiset M, we define $\sigma(M)$ to be the literal multiset $\{\sigma(L) \mid L \in M\}$.

Unless stated otherwise, in this paper P, sometimes with an index, represents a program.

Lemma 2.3.1 Suppose $\sigma \in S_R$. P is stratified[hierarchical, definite] iff $\sigma(P)$ is stratified[resp. hierarchical, definite].

Now assume that I is an interpretation, and R_I is the predicate symbol assignment of I. As usual, when causing no confusions, we often represent I by

$$I = \{R_I(r)(d) \mid R_I(r)(d) = true\},$$

where r is an n-ary predicate symbol, and d an n-tuple of elements of the domain of I. For the above n-tuple d and an n-ary predicate symbol set O, let

$$I(d) = \{R_I(r)(d) \mid R_I(r)(d) \in I\}, \text{ and } I_O(d) = \{R_I(r)(d) \in I(d) \mid r \in O\}.$$

For $\sigma \in S_R$, let $R_{\sigma(I)}$ be the following predicate symbol assignment:

$$\text{For } r \in R, R_{\sigma(I)}(r) = R_I(\sigma(r)).$$

We define $\sigma(I)$ to be the interpretation whose domain, constant assignment, function symbol assignment are the same as those of I, and predicate symbol assignment is $R_{\sigma(I)}$. Then, for a closed formula set FS, I is a model of FS iff $\sigma(I)$ is a model of $\sigma(FS)$. We also have the facts that $\sigma(Comp(P)) = Comp(\sigma(P))$, and I is a model of $Comp(P)$ iff $\sigma(I)$ is a model of $Comp(\sigma(P))$.

Throughout the paper we use $R(T)$ to denote the set of all predicate symbols appearing in T, where T is a formula, a formula set, a literal multiset, or an Herbrand interpretation. For a literal L, $r(L)$ denotes the predicate symbol in L, and L^n represents the formula $\wedge_{i=1}^{n} L (L^n = true$ if $n = 0)$. For a literal multiset M, by $M^+[M^-]$ we denote the literal multiset consisting of the positive[resp. negative] atoms in M, and $M(L)$ the number of occurrences of L in M. This also implies $A_s \neq A_t$ for $s \neq t (s, t \in \{1, \cdots, u\})$, and $B_s \neq B_t$ for $s \neq t (s, t \in \{1, \cdots, v\})$ when we say that

$$C = (\wedge_{i=1}^{u} A_i^{a_i}) \wedge (\wedge_{j=1}^{v} (\neg B_j)^{b_j}) \rightarrow B_w$$

is a clause. By (1) we denote the identity permutation or identity group.

Up to isomorphisms, $S_{R(P)}$ is a subgroup of S_R. In the following sections, we discuss some subgroups of $S_{R(P)}$.

3 Symmetric groups induced by logic programs

Now, we explore the structure of symmetric groups derived from programs, as well as the actions of a sequence of such symmetric groups on the corresponding programs.

3.1 Symmetric group associated with a logic program

For $C \in P$, let M be the literal multiset of C, and

$$G_C = \{\sigma \in S_{R(P)} \mid \sigma(M) = M\}.$$

In fact, G_C consists of the permutations keeping the disjunction form of C invariant. It is a symmetric group on $R(P)$.

Let $G = \cap_{C \in P} G_C$. Then G is again a symmetric group on $R(P)$. We call it the symmetric group associated with P.

Example 3.1.1 $P = \{Bird(tweety), Bird(x) \wedge \neg Ab(x) \rightarrow Fly(x)\}$.
$G = \{(1), (Ab\ Fly)\}$.

In what follows, if there are no other statements, G always denotes the symmetric group associated with P.

Theorem 3.1.1 For arbitrary $\sigma \in G$ and $r_1, r_2 \in R(P)$, if $\sigma(r_1) = r_2$, then $(r_1 r_2) \in G$.

Proof. If $r_1 = r_2$, then $(r_1 r_2) = (1) \in G$. Otherwise, since $G = \cap_{C \in P} G_C$, for any clause $C \in P$, $\sigma \in G_C$. Let $M = \{\neg A_1, \cdots, \neg A_u, B_1, \cdots, B_w\}$ be the literal multiset of C. Then for any $\sigma \in G$ and $r \in R(M^-)[r \in R(M^+)]$,

$$\sigma(r) \in R(M^-)[\text{resp. } \sigma(r) \in R(M^+)].$$

If $r_1 \notin R(M)$, then $r_2 \notin R(M)$ either. Otherwise, because $\sigma^{-1}(M) = M$,

$$r_1 \in R(\sigma^{-1}(M)) = R(M).$$

Contradiction. So $(r_1 r_2) \in G_C$ in this case. If $r_1 \in R(M)$, among the disjoint cycles which constitute a product of $\sigma \in G$, we assume $\alpha = (r_1 r_2 \cdots r_m)$ is the one containing r_1 and r_2. For $i = 1, \cdots, m$, let $A_i = \{\neg A_{i1}, \cdots, \neg A_{is}\}$ and $B_i = \{B_{i1}, \cdots, B_{it}\}$ denote respectively the multisets consisting of the literals in M^- and M^+ whose predicate symbols are r_i, and

$$\alpha(A_{ij}) = A_{i+1j}, \alpha(A_{mj}) = A_{1j}, j = 1, \cdots, s,$$
$$\alpha(B_{ij}) = B_{i+1j}, \alpha(B_{mj}) = B_{1j}, j = 1, \cdots, t,$$

where $i = 1, \cdots, m-1$. Let the multisets

$$M_{12} = A_1 \cup B_1 \cup A_2 \cup B_2, M' = \cup_{i=3}^{m}(A_i \cup B_i), M'' = M - (M_{12} \cup M').$$

Then M_{12}, M', M'' are a partition of M, and

$$(r_1 r_2)(M_{12}) = M_{12}, (r_1 r_2)(M') = M', (r_1 r_2)(M'') = \alpha(M'') = M''.$$

So $(r_1 r_2)(M) = M$, $(r_1 r_2) \in G_C$. We therefore have $(r_1 r_2) \in G$.

Q.E.D.

Theorem 3.1.2 $G = S_{O_1} \times \cdots \times S_{O_m}$, where $O_1, \cdots, O_m$ are all the orbits of G.

Proof. If $G = (1)$, we are done. Otherwise, we express $\sigma \in G$ as a product of transpositions: $\sigma = \prod_i \alpha_i$. Suppose $\alpha_i = (r_1 r_2)$, and $r_1 \in O_i$. Then, if α_i is seen as a transposition on O_i, $\alpha \in S_{O_i}$. So

$$\alpha_i \in S_{O_1} \times \cdots \times S_{O_m}, \sigma = \prod_i \alpha_i \in S_{O_1} \times \cdots \times S_{O_m}.$$

Let $\sigma \in S_{O_1} \times \cdots \times S_{O_m}$, $C \in P$, and M the literal multiset of C. For any $L \in M$, suppose $r(L) = r_1$, and $r_1 \in O_i$. Then $r_2 = \sigma(r_1) \in O_i$. From Theorem 3.1.1, we have

$$(r_1 r_2) \in G, \sigma(L) = (r_1 r_2)(L) \in M, \sigma(M) \subseteq M.$$

For the above L, there exists $r_0 \in R(P)$ such that $\sigma(r_0) = r_1$. Clearly $r_0 \in O_i$. By Theorem 3.1.1, $(r_1 r_0) \in G$. So $L' = (r_1 r_0)(L) \in M$, and $\sigma(L') = L$. Therefore $M \subseteq \sigma(M)$. We thus have

$$\sigma(M) = M, \sigma \in G, S_{O_1} \times \cdots \times S_{O_m} \subseteq G.$$

Q.E.D.

These two theorems show that G, the symmetric group associated with P, is of a quite simple structure. In addition, Theorem 3.1.1 indicates that G is generated by transpositions.

Before closing this section, we discuss briefly how to derive G from P. As a matter of fact, Theorem 3.1.1 also tells us, to obtain G, we needn't exhaust all permutations in $S_{R(P)}$.

Let $C \in P$. we first look how to compute the orbits of G_C.

Suppose that $O_C(r)$ is the orbit of G_C containing $r \in R(P)$. If $r \notin R(C)$, then

$$O_C(r) = \{r' \in R(P) \mid r' \notin R(C)\},$$

namely the set of all predicate symbols not occurring in C. Otherwise, for $r' \in R(P)$, $r' \in O_C(r)$ iff for any $L \in M$ in which r occurs, we have $L' = (rr')(L) \in M$ and $M(L) = M(L')$, where M is the literal multiset of C.

We claim that $\cap_{C \in P} O_C(r)$ is an orbit of G.

As a matter of fact, let O_r be the orbit of G which contains r. Since $G \subseteq G_C$, $O_r \subseteq O_C(r)$. So

$$O_r \subseteq \cap_{C \in P} O_C(r).$$

On the other hand, for any $r' \in \cap_{C \in P} O_C(r)$, by Theorem 3.1.1 we know $(rr') \in G_C$. So

$$(rr') \in \cap_{C \in P} G_C = G, r' \in O_r, \text{ and therefore } \cap_{C \in P} O_C(r) \subseteq O_r.$$

Accordingly, G, represented by its orbits, can be obtained in the following method.

First compute $O_C(r)$ for each $C \in P$; then $\cap_{C \in P} O_C(r)$ is the orbit of G containing $r \in R(P)$. When r runs out of $R(P)$, all orbits, say $O_1, \cdots, O_m$, of G are obtained, and $G = S_{O_1} \times \cdots \times S_{O_m}$.

The method terminates, for both $R(P)$ and the literal multiset of a clause are finite.

3.2 Orbits, simplified forms and symmetric group of a logic program

In this section, we define the notions of orbits and symmetric group of a program, and present a procedure to obtain a new program.

3.2.1 Orbits of a logic program

We start by describing the procedure to create a new program $E(P)$ from P. The idea is motivated by the investigation of minimal models of P and models of $Comp(P)$. Roughly speaking, for any clause $C \in P$, if some atoms with the elements of a proper orbit O of G as predicate symbols occur negatively in the literal multiset M of C, then we remove C from P. Otherwise, if they occur positively in M, we choose an element of O as its representative, delete the negative atoms with the other elements of O as predicate symbols in C, and let the representative of O act as the head predicate symbol if that of the head of C is located in O.

In the next section, we expose the relation between the minimal models of P and the new program P_n derived by successively applying this procedure, as well as that between the models of their completions. Then, we discuss the derivation of negative information and semantic issues by using P_n as a standard.

Let $O_1, \cdots, O_m$ be all the orbits of G. First, pick up a predicate symbol r_k in each O_k, and call it the representative of O_k.

Let $C(0) = C$; For $k \geq 1$, assume

$$C(k-1) = (\wedge_{i=1}^{u} A_i^{a_i}) \wedge (\wedge_{j=1}^{v} (\neg B_j)^{b_j}) \rightarrow B_w.$$

If there exists an $A_i (i \in \{1, \cdots, u\})$, such that $r(A_i) \in O_k$ and O_k is proper, then let $E(C) = \emptyset$; Otherwise, let $E(C) = \{C(m)\}$, where $C(m)$ is a clause derived in the following way:

Suppose that $B = \{B_{j1}, \cdots, B_{js}\}$ is the set of all atoms occurring in the body of $C(k-1)$ such that $r(B_{j1}) \in O_k, \cdots, r(B_{js}) \in O_k$, where $J = \{j1, \cdots, js\} \subseteq V = \{1, \cdots, v\}$.

Case 1. $r(B_w) \notin O_k$;

Choose $B_t \in B$ such that $r(B_t) = r_k$, and let

$$C(k) = (\wedge_{i=1}^{u} A_i^{a_i}) \wedge (\wedge_{j \in (V-J)} (\neg B_j)^{b_j}) \wedge (\neg B_t)^{b_t} \rightarrow B_w.$$

Case 2. $r(B_w) \in O_k$.

Case 2.1 There exists a $B_s \in B$, such that $r(B_w) = r(B_s)$;

Choose $B_t \in B \cup \{B_w\}$ such that $r(B_t) = r_k$, and let

$$C(k) = (\wedge_{i=1}^{u} A_i^{a_i}) \wedge (\wedge_{j \in (V-J)} (\neg B_j)^{b_j}) \wedge B_t^{b_s} \rightarrow B_t.$$

Case 2.2 Otherwise.
Choose $B_t \in B \cup \{B_w\}$ such that $r(B_t) = r_k$, and let

$$C(k) = (\wedge_{i=1}^{u} A_i^{a_i}) \wedge (\wedge_{j\in(V-J)}(\neg B_j)^{b_j}) \to B_t.$$

Let $E(P) = \cup_{C\in P} E(C(m))$.

This procedure is non-deterministic. If n_k is the number of elements of O_k, then we can obtain at most $\prod_{k=1}^{n} n_k$ different $E(P)$ corresponding to the different choices of representatives r_k of O_k. However, for any two of them, say $E(P)_1$ and $E(P)_2$, there exists $\sigma \in G$ such that $E(P)_1 = \sigma(E(P)_2)$. In the sequel, we sometimes identity the above $E(C)(= \{C(m)\})$ and $C(m)$.

We also note that actually we can define $E(C)$ analogously for a general clause C (a disjunction of literals). In fact, a disjunction of negative atoms can be treated as a clause with empty head. For the results on some notions independent of syntax, for example, minimal models, the CWA, and GCWA, etc, we can suppose that P is a consistent general clause set instead of only a normal program. We no longer mention this point in the later discussions.

Example 3.2.1 $P = \{p \wedge n \to z, \neg p \wedge \neg z \to n\}$.

$G = \{(1), (pn)\}$. G has two orbits: $O_0 = \{p, n\}$ and $O_0' = \{z\}$, where only O_0 is proper. If we choose p as the representative of O_0, then $E(P) = \{\neg z \to p\}$. If we choose n as the representative of O_0, then $E(P) = \{\neg z \to n\}$.

The following is the procedure successively from P to obtain a program sequence, a symmetric group sequence, and a so-called G-simplified form of P.

Let $P_0 = P$; G_0 the symmetric group associated with P;
For $k \geq 0$, while $P_k \neq \emptyset$ and $G_k \neq (1)$,
let $P_{k+1} = E(P_k)$;
G_{k+1} the symmetric group associated with $E(P_k)$
if $E(P_k) \neq \emptyset$.

This procedure terminates, since when $G_k \neq (1)$ the number of predicate symbols occurring in P_{k+1} is less than that of those in P_k. Upon the termination, we obtain a program series $P_0, \cdots, P_n$, and a symmetric group series $G_0, \cdots, G_m$ ($m = n-1$, or n). We call them a program sequence and a symmetric group sequence respectively. We say that P_n is a G-simplified form of P. Obviously the symmetric group associated with P_n is (1) if P_n is non-empty. We notice that the difference between two symmetric group sequences results from the different choices of representatives of orbits. We also remark here that the choices of representatives of orbits of P_{k+1} depends on those of P_k.

Example 3.2.2 For the P in Example 3.2.1, $P_0 = P$. $G_0 = G$.

We choose $P_1 = \{\neg z \to p\}$. Then $G_1 = \{(1), (pz)\}$.

We choose $P_2 = \{\to z\}$. Then $G_2 = \{(1)\}$.

Therefore, P_0, P_1, P_2 is a program sequence, and G_0, G_1, G_2 a symmetric group sequence. P_2 is a G-simplified form of P.

Hereafter, we assume that $P_0, \cdots, P_n$ is a program sequence, $G_0, \cdots, G_m$ a symmetric group sequence as derived above. We notice again that each G_k is a subgroup of $S_{R(P_k)}$.

Let $\mathcal{O} = \cup_{k=0}^{m}\{O \mid O \text{ is an orbit of } G_k\}$, namely the set of all orbits of G_k $(k = 0, \cdots, m)$. We define a binary relation $\sim$ on $\mathcal{O}$:

$$\text{For } O_1, O_2 \in \mathcal{O}, O_1 \sim O_2 \text{ iff } O_1 \cap O_2 \neq \emptyset.$$

$\sim$ is reflexive and symmetric. Let $\simeq$ be its transitive closure. $\simeq$ is thus an equivalence on $\mathcal{O}$. Let $\mathcal{O}_i (i = 1, \cdots, l)$ be all the equivalence classes of $\simeq$. We call each

$$O_i(P) = \cup_{O \in \mathcal{O}_i} O$$

an orbit of P. And each $O \in \mathcal{O}_i$ is called a component of this orbit. Such a component is said to be proper, if it is not a singleton. If an orbit of P consists of more than one element, we say that it is proper. Otherwise, we call its element fixed by P. By $Fix(P)$ we denote the set of all elements of $R(P)$ fixed by P.

Example 3.2.3 We consider the P in Example 3.2.2.

G_0 has two orbits: $O_0 = \{p, n\}$ and $O'_0 = \{z\}$.
G_1 has one orbit: $O_1 = \{p, z\}$.
G_2 has one orbit: $O_2 = \{z\}$.
So P has one orbit: $O(P) = \{p, n, z\}$.

We may also use graphs to define the notion of orbits of P. Let W be the graph with $\mathcal{O}$ as the vertex set, such that there is an edge between vertices O_1 and O_2 iff $O_1 \cap O_2 \neq \emptyset$. If $W_1, \cdots, W_l$ are all the connected components of W, then each union of vertex sets of W_i is an orbit of P. To be convenient, in what follows we may suppose that in W, there are no rings, and O_1 and O_2 appear as two different vertices if $O_1 = O_2$, however, they are orbits of two different G_s and G_t. So there is exactly one edge between two vertices.

All orbits of P constitute a partition of $R(P)$.

In fact, for any $r \in R(P)$, there exists an $O_i(P)$, $r \in O_i(P)$. And for any $i = 1, \cdots, l$, $O_i(P) \subseteq R(P)$. Now let

$$O_i(P) = \cup_{O \in \mathcal{O}_i} O (i = 1, 2)$$

be two different orbits of P. If $O_1(P) \cap O_2(P) \neq \emptyset$, we choose an $r \in O_1(P) \cap O_2(P)$. Suppose $r \in O_1 \in \mathcal{O}_1$ and $r \in O_2 \in \mathcal{O}_2$. Then

$$O_1 \cap O_2 \neq \emptyset, O_2 \in \mathcal{O}_1, \mathcal{O}_2 \subseteq \mathcal{O}_1.$$

Similarly $\mathcal{O}_1 \subseteq \mathcal{O}_2$. Thus $\mathcal{O}_1 = \mathcal{O}_2, O_1(P) = O_2(P)$. This is a contradiction.

Lemma 3.2.1 Suppose that O_s and O_t are orbits of G_s and G_t respectively ($s < t$), and $O_s \cap O_t \neq \emptyset$. (1) $O_s \cap O_t$ consists of only one predicate symbol, which is just the representative of O_s; (2) For any $s \leq k \leq t$, there exists an orbit O_k of G_k, such that $O_s \cap O_k \neq \emptyset$. Furthermore, if $k < t$, then the representatives of O_k and O_s are the same; (3) If O'_t is an orbit of G_t and $O_s \cap O'_t \neq \emptyset$, then $O_t = O'_t$.

Proof. Let r_s be the representative of O_s. If $O_s = \{r_s\}$, we are done; Otherwise, $(O_s - \{r_s\}) \cap R(P_t) = \emptyset$. So $O_s \cap O_t = \{r_s\}$ if it is not empty. (1) holds.

On the other hand, $r_s \in R(P_t) \subseteq R(P_k)$. So the first part of (2) holds. In addition, if $k < t$, r_s has to be the representative of O_k to ensure $r_s \in P_t$. So the second part of (2) holds.

From (1) we know $r_s \in O'_t$. So $O_t \cap O'_t \neq \emptyset$, $O_t = O'_t$. (3) holds.

Q.E.D.

Lemma 3.2.2 Suppose that O_s and O'_s are two orbits of G_s, and $O_s \simeq O'_s$. If $s = m$, then $O_s = O'_s$. Otherwise, if for any orbit O_t of $G_t (t > s)$, $O_s \cap O_t = \emptyset$, $O'_s \cap O_t = \emptyset$, then $O_s = O'_s$.

Proof. Since the graph of the orbit of P containing O_s and O'_s is connected, there exist two paths l and l' containing O_s and O'_s respectively, such that l and l' share a common vertex, say O. By Lemma 3.2.1(2) and the fact that there is no longer the edge with O_s or O'_s and any O_t as two vertices, we may suppose

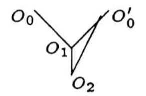
$$l = O_0 - O_1 - \cdots - O_s, \; l' = O'_0 - O'_1 - \cdots - O'_s,$$

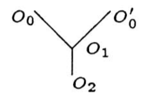
where each pair O_k, O'_k are orbits of G_k. Assume that $O = O_k = O'_k$. From Lemma 3.2.1(3), $O_h = O'_h$ for any $h \geq k$. Especially, $O_s = O'_s$.

Q.E.D.

Now let $O_i(P)$ be an orbit of P, W_i its graph. In W_i we delete the edge whose two vertices are respectively orbits of G_s and G_t, and $t > s + 1$. We then obtain a graph, denoted by $T(O_i(P))$. By Lemma 3.2.1(2), (3) and Lemma 3.2.2, $T(O_i(P))$ is a tree supporting W_i, in which all vertices of W_i appear, the root is the O_s described in Lemma 3.2.2, all leaf nodes are orbits of G_0, and if a father node is an orbit of some G_{k+1}, then all its sons are orbits of G_k. According to Lemma 3.2.1(1), the representatives of son nodes are in its father node.

Example 3.2.4 For the P in Example 3.2.3, the graph and tree of $O(P)$ are as follows.





We claim that the orbits of P are independent of the choice of symmetric group sequences.

Let $O(P)$ and $O'(P)$ be two orbits of P containing $r \in R(P)$ and derived respectively from two symmetric group sequences $G_0, \cdots, G_m$ and $G'_0, \cdots, G'_m$.

To show $O(P) \subseteq O'(P)$, we have to prove

$$\text{for any node } O_k \text{of } T(O(P)), O_k \subseteq O'(P).$$

Suppose that in the tree $T(O(P))$, O_k and $O_{k+1} (k \geq 0)$ are two nodes, and O_k is a son of O_{k+1}. We first prove $O_k \subseteq O'(P)$ iff $O_{k+1} \subseteq O'(P)$.

Assume that O_k and O_{k+1} are orbits of G_k and G_{k+1} respectively, and r_k is the representative of O_k, then $r_k \in O_{k+1}$. From the construction of orbits of P, we know that there exist finite $\sigma_1, \cdots, \sigma_l \in G'_0 \cdots G'_k$ such that $\sigma_1 \cdots \sigma_l(O_{k+1})$ and $\sigma_1 \cdots \sigma_l(O_k)$ are orbits of G'_{k+1} and G'_k respectively. Let $r'_k = \sigma_1 \cdots \sigma_l(r_k)$. Then

$$r'_k \in \sigma_1 \cdots \sigma_l(O_k), r'_k \in \sigma_1 \cdots \sigma_l(O_{k+1}).$$

If $O_k \subseteq O'(P)$[or $O_{k+1} \subseteq O'(P)$], then $r_k \in O'(P)$. By the definition of orbits of P,

$$\sigma_1 \cdots \sigma_l(O_k) \subseteq O'(P) \text{ [resp. } \sigma_1 \cdots \sigma_l(O_{k+1}) \subseteq O'(P)].$$

However, $\sigma_1 \cdots \sigma_l(O_k)$ and $\sigma_1 \cdots \sigma_l(O_{k+1})$ share r'_k. We have

$$\sigma_1 \cdots \sigma_l(O_{k+1}) \subseteq O'(P)\text{[resp. } \sigma_1 \cdots \sigma_l(O_k) \subseteq O'(P)].$$

Also from the definition of orbits of P, we have

$$O_{k+1} \subseteq O'(P)\text{[resp. } O_k \subseteq O'(P)].$$

According to this result, now we only need to show, for the root O of $T(O(P))$, $O \subseteq O'(P)$. In fact, if we let O_r be the orbit of G_0 that contains r, then $O_r \subseteq O(P)$. Since $G'_0 = G_0$, $O_r \subseteq O'(P)$. Again, from the above result, we know $O \subseteq O'(P)$.

Analogously, we can prove $O'(P) \subseteq O(P)$. Therefore $O'(P) = O(P)$.

3.2.2 Simplified forms and symmetric group of a logic program

In the procedure to derive $E(P)$ in Section 3.2.1, if we let $O_1, \cdots, O_m$ be all the orbits of P, then we call the new derived program $E(P)$ a simplified form of P.

If $O_1(P), \cdots, O_l(P)$ are all the orbits of P,

$$G(P) = S_{O_1(P)} \times \cdots \times S_{O_l(P)}$$

is called the symmetric group of P. The orbits of $G(P)$ are just those of P, and by Theorem 3.1.2, $G_0 \cdots G_m \subseteq G(P)$ for a symmetric group sequence $G_0, \cdots, G_m$.

Example 3.2.5 The P in Example 3.2.2 has one orbit $O(P) = \{p, n, z\}$. If we choose z as its representative, $\{\rightarrow z\}$ is a simplified form of P, and $G(P) = S_{\{p,n,z\}}$.

If each $O_i(P)$ contains n_i elements, then there are $\prod_{i=1}^{l} n_i$ simplified forms of P. However, for any two simplified forms P_n and P_n' of P, there exists $\sigma \in G(P)$ such that $P_n' = \sigma(P_n)$. Therefore, by Theorem 2.3.1, a simplified form of P is stratified[hierarchical, definite], iff all simplified forms of P are stratified[resp. hierarchical, definite].

Lemma 3.2.3 Let M be the literal multiset of $C \in P$. (1) Suppose that $O_i(P)$ are all the proper orbits of P such that $O_i(P) \cap R(M^-) \neq \emptyset$. There exist an $O_i(P)$ and a proper component O_i of $O_i(P)$ such that $O_i \subseteq R(M^-)$; (2) If $R(M^-) \subseteq Fix(P)$, then for any $r \in M^+$ and the orbit $O_r(P)$ of P containing r, $O_r(P) \subseteq R(M^+)$.

Proof. (1) For a proper $O_j(P)$, if $O_j(P) \cap R(M^-) \neq \emptyset$, let $r_j \in O_j(P) \cap R(M^-)$, and suppose that O_j is a component of $O_j(P)$, such that $r_j \in O_j$. Since at least one component of $O_j(P)$ is proper, by Lemma 3.2.1(1), we may assume that O_j is proper.

If O_j is an orbit of G_0, then $O_j \subseteq R(M^-)$. We are done. Otherwise, let O_j be an orbit of $G_t (t \geq 1)$. If $O_j \subseteq R(M^-)$, we are done. Otherwise, there exists $s \leq t-1$ such that

$$\underbrace{E(\cdots E(E}_{s}\ (C))) = \emptyset \text{ (See the procedure to derive } E(P)).$$

This means that there exist $A_1, A_2 \in M^-$, $r(A_1) \neq r(A_2)$, however they both belong to the same orbit, say O_s, of G_s, and $O_s \subseteq R(M^-)$.

(2) Let O_k and O_{k+1} be two nodes of the tree $T(O_r(P))$, and O_k a son of O_{k+1}. We first prove $O_k \subseteq R(M^+)$ iff $O_{k+1} \subseteq R(M^+)$.

As a matter of fact, since $R(M^-) \subseteq Fix(P)$, we know

$$\underbrace{E(\cdots E(E}_{k+1}(C))) \neq \emptyset, \text{ of course } \underbrace{E(\cdots E(E}_{k}(C))) \neq \emptyset.$$

Let C_{k+1} and C_k be the respective clauses of them, and M_{k+1} and M_k the literal multisets of C_{k+1} and C_k, respectively. Let r_k be the representative of O_k.

If $O_k \subseteq R(M^+)$, then $r_k \in R(M_k^+)$. However, $r_k \in O_{k+1}$. So

$$r_k \in R(M_{k+1}^+),$$

for $O_{k+1} \subseteq R(M_{k+1}^+) \subseteq R(M^+)$.

Conversely, suppose $O_{k+1} \subseteq R(M^+)$. Then $r_k \in R(M_{k+1}^+)$. So

$$r_k \in R(M_k^+) \supseteq R(M_{k+1}^+).$$

Thus $O_k \subseteq R(M_k^+) \subseteq R(M^+)$.

If O_0 is the orbit of G_0 containing r, then $O_0 \subseteq R(M^+)$. According to the above claim, the root of $T(O_r(P))$ is contained in $R(M^+)$. Again from the above claim, all nodes of $T(O_r(P))$ are contained in $R(M^+)$. We therefore get the result as required.

Q.E.D.

Let $C \in P$, and M the literal multiset of C. If $R(M^-) \not\subseteq Fix(P)$, by Lemma 3.2.3, there exists a proper orbit O_i of some G_i, such that $O_i \subseteq R(M^-)$. We call such an O_i the nearest in C, if $i = 0$, or for any $j < i$ and proper orbit O_j of G_j, $O_j \not\subseteq R(M^-)$. Obviously, if the above O_i is the nearest in C and $i > 0$, then for any $j < i$ and proper orbit O_j of G_j, $O_j \cap R(M^-) = \emptyset$. Moreover, we have

Lemma 3.2.4 Let the orbit O_i of $G_i (i > 0)$ be the nearest in $C \in P$. Then for any $j < i$, $O_i \subseteq Fix(G_j)$.

Proof. If O_i is the nearest in C, then for $j < i$,

$$\underbrace{E(\cdots E(E}_{j}(C))) \neq \emptyset. \text{ So } O_i \subseteq Fix(G_j).$$

Q.E.D.

Theorem 3.2.5 A G-simplified form of P is a simplified form of P. Conversely, a simplified form of P is a G-simplified form of P.

Proof. From the symmetric group sequence corresponding to a given G-simplified form of P, we can obtain all orbits $O_i(P)$ of P and their trees $T(O_i(P))$. We choose the representative of root of $T(O_i(P))$ as the representative of $O_i(P)$, we then obtain a simplified form of P. By Lemma 3.2.3, it is the same as the given G-simplified form of P.

On the contrary, Let G_0 be the symmetric group associated with P, $P_0 = P$. For an orbit O_0 of G_0, if it contains the representative of some orbit of P, we then choose it as the representative of O_0. Otherwise, choose an arbitrary element of O_0 as its representative. Suppose now we have got G_k and $P_k (k \geq 1)$. We go on the same procedure for the orbits of G_k, and eventually obtain a G-simplified form of P. By Lemma 3.2.3 and 3.2.4, it is the same as the given simplified form of P.

Q.E.D.

This theorem indicates that the G-simplified forms of P and the simplified forms of P are exactly the same. In what follows, we discuss the applications of these notions.

4 Minimal models of logic programs and models of their completions

A clause set has minimal models under the model ordering defined by Bossu and Siegel[5]. However, people are usually interested in minimal Herbrand models. A clause set has models iff it has Herbrand models. In this section, we consider the relations between the minimal models of P and its simplified forms, as well as the models of their completions. To be convenient, in the following when we mention minimal models, we mean that they are minimal Herbrand models.

From now on we suppose that P_n is a simplified form of P, I is an interpretation, R_I is the predicate symbol assignment of I, and d a tuple of the domain elements of I.

According to Theorem 3.2.5, P_n is a G-simplified form of P. We assume that $P_0 = P, \cdots, P_n$ and $G_0 = G, \cdots, G_m$ are the corresponding program and symmetric group sequences respectively.

Lemma 4.1 Suppose that O is an orbit and I a minimal model of P. For two different $r_1, r_2 \in O$, if $r_1(d) \in I_O(d)$ then $r_2(d) \notin I_O(d)$.

Proof. Assume $r_2(d) \in I_O(d)$. Let $I' = I - \{r_1(d)\}$. We want to show that I' is still a model of P.

Suppose that C is an instance of a clause in P, and M the literal multiset of C.

For the case where $R(M^-) \subseteq Fix(P)$, if I makes a literal in M^- true, then I' makes this literal true since $r_1 \notin R(M^-)$. So I' makes C true. Otherwise, I makes a literal B in M^+ true. If $B \neq r_1(d)$, I' makes C true. If $B = r_1(d)$, by Lemma 3.2.3(2), $r_2(d) \in M^+$. However, $r_2(d) \in I'$. So I' still makes C true.

Now suppose $R(M^-) \not\subseteq Fix(P)$. By Lemma 3.2.3(1), there is a proper orbit O_k of some G_k such that $O_k \subseteq R(M^-)$. We assume that this O_k is the nearest in C, and

$$O_k(d') = \{r(d') \mid r \in O_k\} \subseteq M^-.$$

We claim that at most one element of $O_k(d')$ is in $I(d')$. If this claim is true, then at most one element of $O_k(d')$ is in I'. So I' makes C true. Now, we want to prove this claim. Actually, we have the following facts.

(1) If I^i is a minimal model of P_i, and O_i an orbit of G_i, then $I^i_{O_i}(d)$ contains at most one element$(i = 0, \cdots, n-1)$.

Indeed, if $r_3(d), r_4(d) \in I^i_{O_i}(d)$, where $r_3 \neq r_4$ and $r_3, r_4 \in O_i$, then $I'_i = I^i - \{r_3(d)\}$ is a model of P_i. In fact, for an arbitrary $C \in P_i$, let M be the literal multiset of C. If $r_3, r_4 \notin R(M)$, We are done. Otherwise, if $\{r_3, r_4\} \cap R(M^+) \neq \emptyset$, then

$$r_3, r_4 \in R(M^+).$$

So I'_i is a model of C. If $\{r_3, r_4\} \cap R(M^-) \neq \emptyset$, we have

$$r_3, r_4 \in R(M^-).$$

So I'_i is still a model of C. However, $I'_i \subset I^i$. This is in contradiction with the fact that I^i is minimal.

(2) If I^i is a minimal model of P_i, and O_i an orbit of G_i, then there exist $\sigma_{id} \in G_i$, such that $I^{i+1} = \cup_d \sigma_{id}(I^i(d))$ is a minimal model of P_{i+1} $(i = 0, \cdots, n-1)$.

As a matter of fact, if exists, for an arbitrary non-empty

$$I^i(d) = \{r'_{i1}(d), \cdots, r'_{il}(d)\},$$

where $r'_{ij} \in R(P_i)$, let O_{ij} be the orbit of G_i such that $r'_{ij} \in O_{ij}$. By fact (1), $O_{iu} \neq O_{iv}$ if $u \neq v$.

In the procedure to derive P_{i+1} from P_i, suppose that r_{ij} is the representative of O_{ij}. Let $\sigma_{id} = (r'_{i1} r_{i1}) \cdots (r'_{il} r_{il})$. Then $\sigma_{id} \in G_i$. In what follows, we show that $I^{i+1} = \cup_d \sigma_{id}(I^i(d))$ is a minimal model of P_{i+1}.

Let $C_1 \in P_{i+1}$, $C \in P_i$, and $C_1 = E(C)$. Suppose that C'_1 and C' are ground instances of C_1 and C respectively, and $C'_1 = E(C')$. By M and M_1 we denote the literal multisets of C' and C'_1 respectively. If I^i makes a literal $\neg r'_{ij}(d)$ in M^- true, then $\sigma_{id}(I^i(d))$ makes $\neg r_{ij}(d)$ true. Since $R(M^-) \subseteq Fix(G_i)$, $r_{ij} = r'_{ij}$. However,

$$\neg r'_{ij}(d) \in M^- = M_1^-.$$

So $\sigma_{id}(I^i(d))$ makes a literal in M_1^- true, it makes C'_1 true. If I^i makes a literal $r'_{ij}(d)$ in M^+ true, then $\sigma_{id}(I^i(d))$ makes $r_{ij}(d)$ true. However,

$$r_{ij}(d) \in M_1^+.$$

So it makes C_1' true. We have thus proved that I^{i+1} is a model of P_{i+1}.

Additionally, for $i = 0, \cdots, n-1$, we have: (2.1) The minimal models of P_{i+1} are minimal models of P_i; (2.2) If I^i is a minimal model of P_i, then for any $\sigma_{id} \in G_i$, $\cup_d \sigma_{id}(I^i(d))$ is a minimal model of P_i.

The proofs of (2.1) and (2.2) are similar to those of the following Lemma 4.2 (2)and (3). We just need to replace "Lemma 4.1" and "Lemma 3.2.3" in those proofs by the above fact (1) and the fact that if a predicate symbol $r \in R(M^-)$[or $r \in R(M^+)$], the orbit of G_i in which r is located is contained in $R(M^-)$[resp. $R(M^+)$].

Assume that $\cup_d \sigma_{id}(I^i(d) - W(d))$ is a minimal model of P_{i+1}, $W(d) \subseteq I^i(d)$, and at least one $W(d) \neq \emptyset$, then by (2.1), it is a minimal model of P_i. By (2.2),

$$\cup_d \sigma_{id}^{-1} \sigma_{id}(I^i(d) - W(d)) = \cup_d(I^i(d) - W(d)) \subset \cup_d(I^i(d)) = I^i$$

is a model of P_i. Contradiction.

Now we continue to prove the above claim. If $k = 0$, by fact (1), we get the result as required. Otherwise, from fact (2), there are $\sigma_{0d} \in G_0$, $\cdots$, $\sigma_{k-1d} \in G_{k-1}$ such that $\cup_d \sigma_{k-1d} \cdots \sigma_{0d}(I(d))$ is a minimal model of P_k. Let

$$\sigma = \sigma_{k-1d} \cdots \sigma_{0d}.$$

According to fact (1), $\sigma(I(d')) \cap O_k(d')$ contains at most one element. So $I(d') \cap \sigma^{-1}(O_k(d'))$ contains at most one element. From Lemma 3.2.4, we know $\sigma^{-1}(O_k(d')) = O_k(d')$. We are done.

We thus proved that $I' \subset I$ is a model of P. However, since I is minimal, this is impossible. Therefore $r_2(d) \notin I_O(d)$.

Q.E.D.

Lemma 4.2 (1) If I is a model of P and $R(I) \subseteq R(P_n)$, then I is a model of P_n. Furthermore, if this I is a minimal model of P, then it is a minimal model of P_n; (2) If I_n is a minimal model of P_n, then I_n is a minimal model of P; (3) If I is a minimal model of P, then for any $\sigma_d \in G(P)$, $\cup_d \sigma_d(I(d))$ is a minimal model of P.

Proof. In the procedure to get P_n, let $C_n = E(C) \in P_n$, $C \in P$. Suppose that C_n' and C' are ground instances of C_n and C respectively, and $C_n' = E(C')$. Let M and M_n be the literal multisets of C' and C_n' respectively. Then $M_n^- = M^-$.

(1) If I makes a literal in M^- true, then I makes C_n' true. If I makes a literal, say B, in M^+ true, then since $r(B) \in R(I) \subseteq R(P_n)$, we have $B \in M_n^+$. Thus I makes C_n' true. So the first part of (1) holds.

If $I' \subseteq I$ is a minimal model of P_n, then by (2), I' is a minimal model of P. So if I is a minimal model of P, $I' = I$. Therefore I is a minimal model of P_n.

(2) If $E(C) = \emptyset$, there exist at least two different literals $\neg r_1(d)$ and $\neg r_2(d)$ in M^- such that the predicate symbols r_1 and r_2 are in the same orbit of P. Clearly, at most one of $r_1(d)$ and $r_2(d)$ is in I_n. So $I_n(d)$ makes $r_1(d) \wedge r_2(d)$ false, I_n satisfies C'. Now assume $E(C) \neq \emptyset$. If I_n makes a literal in M_n^- true, since $M_n^- = M^-$, I_n makes a literal in M^- true, I_n satisfies C'. Otherwise I_n makes a literal in M_n^+ true. Because $M_n^+ \subseteq M^+$, I_n makes a literal in M^+ true. So I_n still satisfies C'. I_n is therefore a model of P.

Suppose that $I' \subseteq I_n$ is a model of P. Since $R(I') \subseteq R(P_n)$, from the first part of (1), I' is a model of P_n. However, I_n is a minimal model of P_n. $I' = I_n$. So I_n is a minimal model of P.

(3) Let $\sigma_d \in G(P)$. For the case where I makes a literal in M^- true, if there exist two different literals $\neg r_1(d)$ and $\neg r_2(d)$ in M^- such that the predicate symbols r_1 and r_2 are in the same orbit of P, then from Lemma 4.1, at most one of $r_1(d)$ and $r_2(d)$ is in $I(d)$. So at most one of $r_1(d)$ and $r_2(d)$ is in $\sigma_d(I(d))$, $\sigma_d(I(d))$ makes C' true. Otherwise, by Lemma 3.2.3(1), we have

$$R(M^-) \subseteq Fix(P).$$

Thus $\sigma_d(M^-) = M^-$. However, $\sigma_d(I(d))$ makes a literal in $\sigma_d(M^-)$ true. Hence it makes a literal in M^-, and thus C', true. Now suppose that I makes all the literal in M^- false. Then I makes a literal, say $r(d)$, in M^+ true, and by Lemma 4.1 and Lemma 3.2.3(1),

$$R(M^-) \subseteq Fix(P).$$

According to Lemma 3.2.3(2), $\sigma_d(r(d)) \in R(M^+)$. Since $\sigma_d(I(d))$ makes $\sigma_d(r(d))$ true, it makes C' true. We therefore proved that $\cup_d \sigma_d(I(d))$ is a model of P.

If $\cup_d \sigma_d(I(d) - W(d))$ is a minimal model of P, $W(d) \subseteq I(d)$, and at least one $W(d) \neq \emptyset$, then $\cup_d \sigma_d^{-1} \sigma_d(I(d) - W(d)) = \cup_d(I(d) - W(d)) \subset \cup_d(I(d)) = I$ is a model of P. This is in contradiction with the fact that I is minimal. So $\cup_d \sigma_d(I(d))$ is minimal.

Q.E.D.

Theorem 4.3 If I is a minimal model of P, then there exist a minimal model I_n of P_n and $\sigma_d \in G(P)$ such that $I = \cup_d \sigma_d(I_n(d))$. If I_n is a minimal model of P_n, then for any $\sigma_d \in G(P)$, $I = \cup_d \sigma_d(I_n(d))$ is a minimal model of P.

Proof. If I is a minimal model of P, by fact (2) in the proof of Lemma 4.1, there exist $\sigma_{0d} \in G_0, \cdots, \sigma_{n-1d} \in G_{n-1}$ such that $\cup_d \sigma_{n-1d} \cdots \sigma_{0d}(I(d))$ is a minimal model of P_n. Let

$$I_n(d) = \sigma_{n-1d} \cdots \sigma_{0d}(I(d)), \text{ and } \sigma_d = \sigma_{0d}^{-1} \cdots \sigma_{n-1d}^{-1}.$$

Then $\sigma_d \in G_0 \cdots G_m \subseteq G(P)$, and $I = \cup_d \sigma_d(I_n(d))$.

If I_n is a minimal model of P_n, by Lemma 4.2(2), I_n is a minimal model of P. From Lemma 4.2(3), for any $\sigma_d \in G(P)$, $I = \cup_d \sigma_d(I_n(d))$ is a minimal model of P.

Q.E.D.

Theorem 4.3 tells us that all minimal models of P can be obtained from those of P_n. The actions of $G(P)$ on all minimal models of P_n lead to all minimal models of P. In this regard, we say that P_n keeps the minimal models of P.

Actually, Lemma 4.1 can be proved in a simpler way. However, the proof procedure we presented successively reflects the relations between minimal models of the program sequence. It simplifies the proof of Theorem 4.3, and shows that we may replace $G(P)$ by $G_0 \cdots G_m$ in this theorem. the reasons why we use $G(P)$ instead of $G_0 \cdots G_m$ are that $G(P)$ is of a quite simple group structure, it doesn't rely upon the choices of symmetric group sequences, and it is easier to get the minimal models from P_n. We only need to substitute predicate symbols for those in the same orbits of P.

Now, we turn to the completion procedure of programs. We notice that the models of $Comp(P)$ may not be models of $Comp(P_n)$. On the contrary, the models of $Comp(P_n)$ may not be models of $Comp(P)$ either. However, we have the following theorem, which shows that they are the same up to permutations. We note again that the equality $=$ is supposed to be in $Fix(P)$.

Lemma 4.4 Suppose that I is a model of $Comp(P)$, and O is an orbit of G. For two different $r_1, r_2 \in O$, if $R_I(r_1)(d) \in I_O(d)$ then $R_I(r_2)(d) \notin I_O(d)$.

Proof. Assume that r_1 is n-ary. Since $R_I(r_1)(d) \in I_O(d) \subseteq I$, there exists at least one clause C_i in P whose head is $r_1(t_i)$, where t_i is an n-tuple of terms. Clearly, $\neg r_2(t_i)$ appears in the body of C_i, and if we let M_i be the literal multiset of C_i, and $b_i = M_i(\neg r_2(t_i))$, then we can write C_i as

$$C_i = C_i' \wedge (\neg r_2(t_i))^{b_i} \wedge (\neg r_1(t_i))^{b_i - 1} \rightarrow r_1(t_i).$$

So, for the n-tuple x of variables,

$$C = \forall x (\vee_i \exists y_i ((x = t_i) \wedge C_i' \wedge (\neg r_2(t_i))^{b_i} \wedge (\neg r_1(t_i))^{b_i - 1}) \leftrightarrow r_1(x)) \in Comp(P),$$

where y_i is the tuple of all variables in t_i. Hence, if $R_I(r_1)(d) \in I$, there is at least one b_i such that $b_i - 1 = 0$, and therefore $R_I(r_2)(d) \notin I$. Otherwise I makes C false.

Q.E.D.

Let O be a proper orbit of G, and

$$P_\wedge = \{C \in P \mid O \subseteq R(M^-)\},$$

where M stands for the literal multiset of C. By Lemma 4.4, I is a model of $Comp(P)$ iff I is a model of $Comp(P - P_\wedge)$. On the other hand, in the procedure to derive $E(P)$, we can ensure that removing all clauses in $P_\wedge$ from P doesn't have any impact on the final result. So, in the following, without loss of generality, we assume $P_\wedge = \emptyset$. Let r_1, r_2 be two different n-ary predicate symbols in O, and r_2 the representative of O in the procedure to derive P_1.

Let the following P_{H1} and P_{H2} be the P-definitions of r_1 and r_2, respectively.

$$P_{H1} = \{C_{1i} \wedge (\neg r_1(t_{1i}))^{b_{1i}-1} \wedge (\neg r_2(t_{1i}))^{b_{1i}} \rightarrow r_1(t_{1i}) : i = 1, \cdots, m_1\},$$

$$P_{H2} = \{C_{2i} \wedge (\neg r_1(t_{2i}))^{b_{2i}} \wedge (\neg r_2(t_{2i}))^{b_{2i}-1} \rightarrow r_2(t_{2i}) : i = 1, \cdots, m_2\}.$$

Let $P_\neg$ be the set of all clauses in $P - (P_{H1} \cup P_{H2})$ in which r_1 and r_2 occur negatively. Suppose

$$P_\neg = \{D_{ij} \wedge (\neg(r_1(t_{ij}))^{b_{ij}} \wedge (\neg(r_2(t_{ij}))^{b_{ij}} \rightarrow p_i(u_{ij}) : i = 1, \cdots, l, j = 1, \cdots, k_i\}.$$

For $i = 1, \cdots, l$, assume that the following $P_{\neg i}$ is the $(P - P_\neg)$-definition of p_i.

$$P_{\neg i} = \{F_{ij} \rightarrow p_i(v_{ij}) : j = 1, \cdots, k_i\}.$$

In the above expressions, C_{ij}, D_{ij} and F_{ij} are conjunctions of literals where r_1, r_2 do not occur. t_{ij} are n-tuples of terms. If the predicate symbol p_i is n_i-ary, then u_{ij} and v_{ij} are n_i-tuples of terms.

Now let $P' = P - (P_{H1} \cup P_{H2} \cup P_\neg \cup \cup_i P_{\neg i})$, and

$$P_{H1}(O) = \{C_{1i} \wedge (\neg r_2(t_{1i}))^{b_{1i}-1} \rightarrow r_2(t_{1i}) : i = 1, \cdots, m_1\},$$

$$P_{H2}(O) = \{C_{2i} \wedge (\neg r_2(t_{2i}))^{b_{2i}-1} \rightarrow r_2(t_{2i}) : i = 1, \cdots, m_2\},$$

$$P_\neg(O) = \{D_{ij} \wedge (\neg(r_2(t_{ij}))^{b_{ij}} \rightarrow p_i(u_{ij}) : i = 1, \cdots, l, j = 1, \cdots, k_i\}.$$

Let $P(O) = P' \cup P_{H1}(O) \cup P_{H2}(O) \cup P_\neg(O) \cup \cup_i P_{\neg i}$. Assume that $CompStep1(P)$ and $CompStep1(P(O))$ are the sets consisting respectively of the completed P-definitions and $P(O)$-definitions of r_1, r_2 and $p_i (i = 1, \cdots, l)$. By Lemma 4.4 and the definition of applying a permutation to interpretations, we have the following fact:

Suppose that I is an interpretation satisfying the equality theory. If I is a model of $CompStep1(P)[CompStep1(P(O))]$, then there exist $\sigma_d \in S_{\{r_1, r_2\}}$ such that $\cup_d \sigma_d(I(d))$ is a model of $CompStep1(P(O))$[resp. $CompStep1(P)$].

Lemma 4.5 If I is a model of $Comp(P)[Comp(P(O))]$, then there exist $\sigma_d \in S_{\{r_1, r_2\}}$ such that $\cup_d \sigma_d(I(d))$ is a model of $Comp(P(O))$[resp. $Comp(P)$].

Proof. Let W be the set of completed P'-definitions of r_1, r_2 and $p_i (i = 1, \cdots, l)$. Then

$$Comp(P) = CompStep1(P) \cup (Comp(P') - W),$$

$$Comp(P(O)) = CompStep1(P(O)) \cup (Comp(P') - W).$$

Since r_1 and r_2 do not appear in $Comp(P') - W$, for any $\sigma_d \in S_{\{r_1,r_2\}}$, if I is a model of $Comp(P)[Comp(P(O))]$, then $\cup_d \sigma_d(I(d))$ is a model of $Comp(P') - W$. According to the above fact, we get the result as required.

Q.E.D.

Theorem 4.6 If I is a model of $Comp(P)$, then there exist $\sigma_d \in G(P)$ such that $\cup_d \sigma_d(I(d))$ is a model of $Comp(P_n)$. Conversely, if I_n is a model of $Comp(P_n)$, then there exist $\sigma_d \in G(P)$ such that $\cup_d \sigma_d(I_n(d))$ is a model of $Comp(P)$.

Proof. By Lemma 4.5 and Theorem 3.1.1, we know that for $i = 0, \cdots, n-1$, if I^i is a model of $Comp(P_i)[Comp(P_{i+1})]$ then there exist $\sigma_{id} \in G_i$, such that $\cup_d \sigma_d(I(d))$ is a model of $Comp(P_{i+1})$[resp. $Comp(P_i)$]. Let

$$\sigma_d = \sigma_{n-1d} \cdots \sigma_{0d}, \text{[resp. } \sigma_d = \sigma_{0d} \cdots \sigma_{n-1d}\text{]}.$$

Then $\sigma_d \in G_m \cdots G_0 \subseteq G(P)$ [resp. $\sigma_d \in G_0 \cdots G_m \subseteq G(P)$], and $\cup_d \sigma_d(I(d))$ is a model of $Comp(P_n)$[resp. $Comp(P_0) = Comp(P)$].

Q.E.D.

So, up to permutations, $Comp(P_n)$ keeps models of $Comp(P)$. As a corollary, $Comp(P)$ is consistent iff the completions of its simplified forms are consistent.

Example 4.1 In Example 3.2.5, $P_2 = \{z\}$ is a simplified form of P, and the unique orbit of P is $O(P) = \{p, n, z\}$. P_2 has one minimal model $\{z\}$. So all the minimal model of P are $\{\sigma(\{z\}) | \sigma \in S_{\{p,n,z\}}\} = \{\{p\}, \{n\}, \{z\}\}$. On the other hand, $\{n\}$ and $\{z\}$ are models of $Comp(P)$ and $Comp(P_2)$ respectively. For $\sigma = (nz) \in S_{\{p,n,z\}}$, we see that $\{n\} = \sigma\{z\}$, and $\{z\} = \sigma\{n\}$.

5 On the derivation of negative information

In this section, we discuss the applications of the symmetric groups we defined to the CWA, GCWA, and the completion procedure. We are mainly interested in what negative information can be assumed. We also give another proposal to cope with negative information based on the notion of orbits of programs.

5.1 CWA and GCWA

We begin with the CWA and GCWA. For more details and the other sophisticated generalizations, the reader is referred to [22], [20] and [10].

5.1.1 Deriving negative information

For a ground atom A, we say that $\neg A$ is derivable from P under the CWA (denoted by $CWA(P) \models \neg A$), if $P \not\models A$. We say that $\neg A$ is derivable from P under the GCWA (denoted by $GCWA(P) \models \neg A$), if for any disjunction C of ground atoms, $P \models A \vee C$ implies $P \models C$.

When the CWA is consistent, the GCWA coincides with it. For this, please see any textbooks, such as [16] or [18], on logic programming or on non-monotonic reasoning. Here we just note that $GCWA(P) \models \neg A$ iff A is not in the union of all minimal models of P. In the following, we suppose that A is a ground atom, and P_n a simplified form of P.

Lemma 5.1.1 $P \models A$ iff $P_n \models A$ and $r(A) \in Fix(P)$.

Proof. Let $A = r(d')$, where $r = r(A)$, and d' is a tuple of the Herbrand constants.

If $P \not\models A$, there exists a minimal model I of P, such that

$$A \notin I(d')(P \cup \{\neg A\} \text{ has no models iff it has no Herbrand models}).$$

By Theorem 4.3, there exist $\sigma_d \in G(P)$ such that $I_n = \cup_d \sigma_d(I(d))$ is a model of P_n. If $r \in Fix(P)$, then $A \notin \sigma_{d'}(I(d'))$. Therefore,

$$A \notin I_n, P_n \not\models A.$$

If $P_n \not\models A$, there exists a minimal model I_n of P_n such that $A \notin I_n$. However, by Theorem 4.3, I_n is a model of P, so

$$P \not\models A.$$

Otherwise, if $P_n \models A$ and $r \notin Fix(P)$, we let O_r be the orbit of P where r locates, $r' \in O_r$ and $r' \neq r$. Let $\sigma_d = (rr')$, and I_n a minimal model of P_n. Then $\sigma_d \in G(P)$, and by Theorem 4.3, $I = \cup_d \sigma_d(I_n(d))$ is a model of P. However, $r \notin R(I)$. Thus

$$A \notin I, P \not\models A.$$

Q.E.D.

According to this lemma, we have

Theorem 5.1.2 $CWA(P) \models \neg A$ iff $CWA(P_n) \models \neg A$ or $r(A) \notin Fix(P)$.

This theorem indicates that if an n-ary predicate symbol r lies in a proper orbit of P, then for any n-tuple d of the Herbrand constants, we can infer $\neg r(d)$ from P under the CWA. Otherwise, we need to test whether $\neg r(d)$ is derivable from P_n under the CWA. In fact, we even have a stronger result: Let $P_F = \{C \in P | R(C) \subseteq Fix(P)\}$. Then $CWA(P) \models \neg A$ iff $CWA(P_F) \models \neg A$.

In the sequel, when we say $GCWA(P_n) \models \neg A$, we always assume without loss of generality that in the procedure to derive P_n from P, $r(A)$ is the representative of the orbit of P where it locates.

Theorem 5.1.3 If $GCWA(P) \models \neg A$, then there is $\sigma \in G(P)$, such that $GCWA(P_n) \models \neg\sigma(A)$; If $GCWA(P_n) \models \neg A$, then for any $\sigma \in G(P)$, $GCWA(P) \models \neg\sigma(A)$.

Proof. As in the proof of Lemma 5.1.1, we let $A = r(d')$, and assume that r' is the representative of the orbit of P where r locates. Then $\sigma = (rr') \in G(P)$, $\sigma(A)$ meets the above assumption. For an arbitrary minimal model I_n of P_n, if $\sigma(A) \in I_n$ then by Theorem 4.3, $I'_n = \sigma(I_n)$ is still a minimal model of P. And $A \in I'_n$. Hence

$$GCWA(P) \not\models \neg A.$$

This is in contradiction with the hypothesis.

If there is $\sigma \in G(P)$ such that $GCWA(P) \not\models \neg\sigma(A)$, then there exists a minimal model I of P such that $\sigma(r)(d') \in I(d')$. If $I(d) = \{r_1(d), \cdots, r_k(d)\}$, we let $I'(d) = \{r'_1(d), \cdots, r'_k(d)\}$, where r'_i is the representative of the orbit of P where r_i locates in the procedure to obtain the simplified form $\sigma(P_n)$ of P. Similar to the fact (2) in the proof of Lemma 4.1, $\cup_d I'(d)$ is a minimal model of $\sigma(P_n)$. Because in the procedure to derive P_n from P, r is the representative of the orbit of P where it locates, in the procedure to derive $\sigma(P_n)$ from P, $\sigma(r)$ is the representative of the orbit of P where it locates. So,

$$\sigma(r)(d') \in I'(d'), GCWA(\sigma(P_n)) \not\models \neg\sigma(A).$$

Therefore, there is a minimal model I' of $\sigma(P_n)$, such that $\sigma(A) \in I'$. Obviously $A \in \sigma^{-1}(I')$, and $\sigma^{-1}(I')$ is a minimal model of P_n. Hence

$$GCWA(P_n) \not\models \neg A,$$

which is in contradiction with the hypothesis.

Q.E.D.

Theorem 5.1.3 demonstrates that, to obtain the negative facts from P under the GCWA, we may apply the permutations in $G(P)$ to those from P_n under the GCWA.

About the positive facts, from Lemma 5.1.1, we have

Theorem 5.1.4 (1) $CWA(P) \models A$ iff $CWA(P_n) \models A$ and $r(A) \in Fix(P)$; (2) $GCWA(P) \models A$ iff $GCWA(P_n) \models A$ and $r(A) \in Fix(P)$.

Example 5.1.1 $P = \{p \wedge n \rightarrow z, \neg z \rightarrow r\}$.

P has two orbits: $O_1 = \{p, n\}$ and $O_2 = \{z, r\}$. They are both proper. So $\{\neg p, \neg n, \neg z, \neg r\}$ is the set of all negative facts derivable from P under the CWA. CWA is inconsistent in this example.

If we choose p and z as the representatives of O_1 and O_2, we then obtain a simplified form of P: $P_2 = \{z\}$. Only the negation of representative p is derivable from P_2 under the GCWA. So $\{\neg\sigma(p) \mid \sigma \in G(P) = O_1 \times O_2\} = \{\neg p, \neg n\}$ is the set of all negative facts derivable from P under the GCWA.

5.1.2 On the computation of GCWA

The negation of a ground atom is derivable from P under the GCWA iff this ground atom belongs to no minimal model of P. It is very difficult to evaluate this condition directly[12]. One method is to test if some disjunctions of ground atoms can be deduced from P[11]. In this section. We discuss how to simplify this procedure based on the symmetric group we defined. To be able to conduct computing, we suppose that there are no function symbols. Without loss of generality, we discuss in the propositional logic.

In the sequel, a disjunction means the empty clause or a disjunction of finite ground atoms. For a set O of ground atoms, by $\vee O$ we denote the disjunction of all atoms in O. For a disjunction C, we say $r \in C$, if atom r appears in C. Similarly, for an atom set O, we say $O \subseteq C$, if for any $r \in O$, $r \in C$. We call C irredundant, if for any proper subset O of the set consisting of the atoms in C, $P \models C$, however, $P \not\models \vee O$.

Theorem 5.1.5 Let C be an irredundant disjunction, r be a ground atom, and O_r the orbit of P containing r. If $r \in C$, then $O_r \subseteq C$.

Proof. Suppose $C = (\vee_i r_i') \vee (\vee_j r_j)$, where r_i', r_j are ground atoms, and r_j are all the ground atoms in C belonging to O_r. Obviously, $r \in \{r_j \mid j\} \neq \emptyset$. Let MM be the set of all minimal models of P, and

$$MM_1 = \{I \in MM \mid \text{There exists } i, \text{ such that } r_i' \in I\}, MM_2 = MM - MM_1.$$

$MM_2 \neq \emptyset$. Otherwise, $P \models \vee_i r_i'$, which is in contradiction with the fact that C is irredundant. For an arbitrary $I \in MM_2$, there exists an $r_s \in O_r$ such that $r_s \in I$. Choose an arbitrary $r_t \in O_r$. By Lemma 4.1 and 4.2(3),

$$I' = (I - \{r_s\}) \cup \{r_t\}$$

is a minimal model of P. And $I' \in MM_2$. Thus I' satisfies $\vee_j r_j$. From Lemma 4.1, we have $r_t \in \{r_j \mid j\}$. Hence, $O_r = \{r_j \mid j\}$.

Q.E.D.

Let n and m be respectively the numbers of elements of Herbrand universe and orbits of P. To test whether or not $GCWA(P) \models \neg r$, we basically have to test 2^{n-1} disjunctions C to see whether $P \models C \vee r$ implies $P \models C$. However, according to Theorem 5.1.5, we have

$GCWA(P) \models \neg r$, iff for orbits $O_1, \cdots, O_k$ of P, $P \models (\vee O_1) \vee \cdots \vee (\vee O_k) \vee (\vee O_r)$ implies $P \models (\vee O_1) \vee \cdots \vee (\vee O_k)$.

So, now we only need try 2^{m-1} disjunctions C to see if $P \models C \vee (\vee O_r)$ implies $P \models C$. This may simplify the computation procedure in the case when the symmetric group of P is not (1). In addition, if $GCWA(P) \models \neg r$, then for any $r' \in O_r$, $GCWA(P) \models \neg r'$. This means that we only need to test one element in each orbit of P.

Example 5.1.2 We consider the P in Example 5.1.1. To see if $GCWA(P) \models \neg p$, in general one has to test the following 8 disjunctions:

$$n \vee z \vee r, n \vee z, n \vee r, z \vee r, z, r, n \text{ and the empty clause.}$$

According to our result, we only need to test the following 2 disjunctions:

$$z \vee r \text{ and the empty clause.}$$

Also, when we have known $GCWA(P) \models \neg p$, we can conclude $GCWA(P) \models \neg n$ without any testing. This is because p and n are in the same orbit of P.

5.2 OCWA: An extension

We present a new proposal to derive negative information termed orbit CWA (OCWA), which is actually an extension of the GCWA.

Assume that $O_i (i = 1, \cdots, m)$ are all the orbits of P. For each O_i, let O_i^+ be a set consisting of an arbitrarily chosen predicate symbol in O_i, and $O_i^- = O_i - O_i^+$. In the sequel, A still represents a ground atom. Let

$$NF_1 = \{\neg A \mid \text{ There exists } O_i : r(A) \in O_i^-\},$$
$$NF_2 = \{\neg A \mid \text{ There exists } O_i : r(A) \in O_i^+, \text{ and for any disjunction } C, P \models C \vee A \text{ implies } P \models C\}.$$

Let $OCWA(P) = P \cup NF_1 \cup NF_2$. $\neg A$ is said to be derivable from P under the OCWA if $OCWA(P) \models \neg A$. The OCWA preserves consistency:

Theorem 5.2.1 *$OCWA(P)$ is consistent.*

Proof. In the procedure to derive a simplified form of P, we choose the predicate symbol in O_i^+ as the representative. Let P_n be the obtained simplified form of P, and I_n a minimal model of it. By Theorem 4.3, I_n is a minimal model of P. Since the predicate symbols in O_i^- do not appear in P_n anymore, I_n is a model of NF_1. On the other hand, for any $\neg A \in NF_2$, A doesn't belong to any minimal model of P. So $A \notin I_n$, I_n is a model of NF_2. We therefore know that I_n is a model of $OCWA(P)$, it is consistent.

Q.E.D.

Actually, $OCWA(P)$ is equivalent to $GCWA(P_n)$ to derive negative information. The next theorem shows that although new positive facts might be deduced by applying the OCWA, we are able to recognize the situation very easily.

Theorem 5.2.2 $P \models A$ iff $OCWA(P) \models A$ and $r(A) \in Fix(P)$.

Proof. If $P \models A$, then $OCWA(P) \models A$, and by Lemma 5.1.1, $r(A) \in Fix(P)$.

Assume $OCWA(P) \models A$. Let P_n be the simplified form of P as in the proof of Theorem 5.2.1. It is not hard to see that $OCWA(P)$ is logically equivalent to $P_n \cup NF_1 \cup NF_2$(in fact, P_n may be obtained from P and NF_1 by subsumption and resolution[7]). So $P_n \cup NF_1 \cup NF_2 \models A$. If $r(A) \in Fix(P)$, then $r(A) \notin R(NF_1)$. Clearly,

$$R(NF_1) \cap R(P_n) = \emptyset, R(NF_1) \cap R(NF_2) = \emptyset.$$

Thus $P_n \cup NF_2 \models A$. Similar to the proof of first part of Theorem 5.1.3, for any $\neg A' \in NF_2$, $GCWA(P_n) \models \neg A'$. So $GCWA(P_n) \models A$, and thus $P_n \models A$. Again from Lemma 5.1.1, we have $P \models A$.

Q.E.D.

Example 5.2.1 For the P in Example 5.1.2, We choose $O_1^+ = \{n\}$, $O_2^+ = \{r\}$. $\{\neg z, \neg p, \neg n\}$ are all negative facts derivable from P under the OCWA. However, $\neg z$ isn't derivable from P under the GCWA.

In the procedure to formulate the OCWA, if we choose each O_i^+ as an arbitrary non-empty subset of O_i, all the results above still hold. Especially, when $O_i^+ = O_i$, the OCWA coincides with the GCWA. So, the OCWA is in fact a generalization of the GCWA. So far we don't know yet the relations between the OCWA and the CCWA developed by Gelfond and Przymusinska[10]. Nevertheless, the OCWA has an advantage that one can modify the negative facts very easily when changing his mind on the world. However, in the CCWA, it is difficult to change directly from the derived negative facts. One inadvantage of the OCWA is that when the symmetric group of P is (1), the OCWA is exactly the GCWA.

5.3 Completion procedure

Hereafter, we consider another inference rule to derive negative information, which is usually termed negation as failure. The basic idea is that we admit a negative ground atom if it isn't a logical consequence of $Comp(P)$. However, in the case when $Comp(P)$ is inconsistent, it is invalid. Fortunately, the completions of stratified programs are consistent[1].

Let P be stratified. Then all simplified forms of P are stratified.

Indeed, we can first obtain a program sequence $P_0 = P, \cdots, P_n$ in the following way:

Let $P_i (i \geq 0)$ be stratified. For any proper orbit O_i of the symmetric group G_i associated with P_i, if there exist an $r_i \in O_i$ and $C_i \in P_i$ such that r_i is the predicate symbol of the head of C_i, then all predicate symbols in $O_i - \{r_i\}$ cannot appear in the head of any clause in P_i. In the procedure to get P_{i+1} from P_i, this r_i is chosen as the representative of O_i. Otherwise, let an arbitrary element of O_i be its representative.

We thus obtained P_{i+1}, and it is obviously stratified. Hence, P_n is stratified. However, P_n is a simplified form of P by Theorem 3.2.5, and for any a simplified form P_n' of P, there is a $\sigma \in G(P)$ such that $P_n' = \sigma(P_n)$. By Lemma 2.3.1, P_n' is stratified.

Actually, $Comp(P)$ is logically equivalent to $Comp(P_n)$. In what follows, we show that such a P_n can be chosen directly based on the orbits of P.

Let $P_0 = P, \cdots, P_n$ and $G_0, \cdots, G_m$ be a program sequence and the corresponding symmetric group sequence, and

$$P_\wedge = \{C \in P \mid R(M^-) \not\subseteq Fix(P)\},$$

where M denotes the literal multiset of C. For an arbitrary $C \in P_\wedge$, by Lemma 3.2.3, there exists a proper orbit O_i of G_i such that $O_i \subseteq R(M^-)$. Suppose that this O_i is the nearest in C. For two different n-ary predicate symbols in O_i, assume $r_1(t), r_2(t) \in M^-$, where t is an n-tuple of terms.

Let I be a model of $Comp(P)$. I satisfied the equality theory. If $i = 0$, by Lemma 4.4, for any n-tuple d of the elements of domain of I, at most one of $R_I(r_1)(d)$ and $R_I(r_2)(d)$ is in $I(d)$. So I makes the body of C false. If $i > 0$, from the proof of Theorem 4.6, we know that there are $\sigma_{0d}, \cdots, \sigma_{i-1d}$ such that $\cup_d \sigma_{i-1d} \cdots \sigma_{0d}(I(d))$ is a model of $Comp(P_i)$. According to Lemma 4.4, $\sigma_{i-1d} \cdots \sigma_{0d}(I(d))$ contains at most one of $R_I(r_1)(d)$ and $R_I(r_2)(d)$. Then by Lemma 3.2.4, $I(d)$ contains at most one of $R_I(r_1)(d)$ and $R_I(r_2)(d)$. I still makes the body of C false. So I is a model of $Comp(P - P_\wedge)$.

On the contrary, let I be a model of $Comp(P - P_\wedge)$. I satisfied the equality theory. If there is not a clause in $P - P_\wedge$, in whose head r_1 or r_2 is the predicate symbol, then I obviously makes the body of C false. Otherwise, for example, we assume there are clauses C_j in $P - P_\wedge$, whose heads are $r_1(t_j)$. From Lemma 3.2.3(2), $\neg r_2(t_j)$ appears in the body of C_j. Similar to the proof of Lemma 4.4, at most one of $R_I(r_1)(d)$ and $R_I(r_2)(d)$ is in $I(d)$, I still makes the body of C false. So I is a model of $Comp(P)$.

We therefore proved that I is a model of $Comp(P)$ iff I is a model of $Comp(P - P_\wedge)$. Namely, $Comp(P)$and $Comp(P - P_\wedge)$ are logically equivalent.

For any proper orbit O of P, if there exist $r \in O$ and $C \in P - P_\wedge$ such that r is the predicate symbol of head of C, in the procedure to get a simplified form of P, we choose this r as the representative of O. Otherwise, choose an arbitrary element of O as its representative. We call such a derived simplified form P_n of P keep heads. If P is stratified, by Lemma 3.2.3(2), all predicate symbols in $O - \{r\}$ cannot appear in the heads of clauses in $P - P_\wedge$.

Theorem 5.3.1 If P is stratified and P_n keeps heads, then $Comp(P)$ and $Comp(P_n)$ are logically equivalent.

Proof. For an n-ary predicate symbol r, if its completed $(P - P_\wedge)$-definition is $\forall x(\neg r(x))$, then it is still the completed P_n-definition of r. The contrary also holds since P is stratified and P_n keeps heads.

Otherwise, suppose $\{C_i \mid i\}$ is the $(P - P_\wedge)$-definition of r. Without loss of generality, we assume that besides the orbit $O_r = \{r_1, \cdots, r_s, r\}$ containing r, there is one proper orbit $O_i = \{p_{i1}, \cdots, p_{ik_i}\}$ of P, whose elements appear in C_i:

$$C_i = C_i' \wedge (\wedge_{j=1}^{k_i}(\neg p_{ij}(t_i))^{b_i}) \wedge (\wedge_{j=1}^{s}(\neg r_j(t))) \rightarrow r(t),$$

where C_i' is a conjunction of literals whose predicate symbols are in $Fix(P)$, and t, t_i are tuples of terms, b_i is the number of occurrences of $\neg p_{ij}(t_i)$ in the literal multiset of C_i(Since P is stratified, the number of occurrences of $r_j(t)$ is 1). Then, the completed $(P - P_\wedge)$-definition of r is

$$F(r) = \forall x(\vee_i \exists y_i(C_i' \wedge (\wedge_{j=1}^{k_i}(\neg p_{ij}(t_i))^{b_i}) \wedge (\wedge_{j=1}^{s}(\neg r_j(t))) \wedge (x = t)) \leftrightarrow r(x)),$$

where y_i are the tuples of variables in C_i. Suppose that p_{i1} is the representative of O_i. Since P is stratified, by Lemma 3.2.3(2), $p_{ij} \in O_i - \{p_{i1}\}$ and r_j can no longer appear in the heads of any clauses in P. So their completed $(P - P_\wedge)$-definitions are

$$F(p_{ij}) = \forall x'(\neg p_{ij}(x')) \text{ and } F(r_j) = \forall x(\neg r_j(x))$$

respectively. Because $P_{ij} (j \neq 1)$ and r_j do not appear in P_n, $F(p_{ij})$ and $F(r_j)$ are in $Comp(P_n)$, too. And since r is the representative of O_r, the completed P_n-definition of r is

$$F_n(r) = \forall x(\vee_i \exists y_i(C_i' \wedge (\neg p_{i1}(t_i))^{b_i} \wedge (x = t)) \leftrightarrow r(x)).$$

Let $F = \{F(p_{ij}) \mid i; j = 2, \cdots, k_i\} \cup \{F(r_j) \mid j = 1, \cdots, s\}$. Then, together with the equality theory, $\{F(r)\} \cup F$ and $\{F_n(r)\} \cup F$ are logically equivalent. Therefore $Comp(P - P_\wedge)$ and $Comp(P_n)$ are logically equivalent. We are done.

Q.E.D.

Corollary 5.3.2 Suppose that P is stratified and P_n keeps heads. For a ground literal L, $Comp(P) \models L$ iff $Comp(P_n) \models L$.

Example 5.3.1 $P = \{p \wedge n \to z, p \wedge n \to r, \neg z \to r\}$.

P is stratified. It has two simplified forms $P_2 = \{r\}$ and $P_2' = \{z\}$. Only P_2 keeps heads.

$Comp(P) = \{\neg p, \neg n, (p \wedge n) \vee \neg z \leftrightarrow r, p \wedge n \leftrightarrow z\}$,

$Comp(P_2) = \{\neg p, \neg n, \neg z, r\}, Comp(P_2') = \{\neg p, \neg n, z, \neg r\}$.

$Comp(P_2)$ is logically equivalent to $Comp(P)$. But $Comp(P_2')$ isn't.

6 On semantics

We first discuss the semantics for definite, hierarchical and stratified programs. Then we extend these notions to more general ones, and develop the similar semantics.

6.1 Definite, hierarchical and stratified logic programs

We mainly focus on the model and procedural semantics for definite programs, the completeness of a procedural semantics for hierarchical programs, and the standard or perfect model semantics for stratified programs.

6.1.1 Definite logic programs

If P is definite, then P has the least model, the simplified form P_n of P is unique, and it is a subset of P. P_n is obviously definite, it also has the least model. We refer the reader to [13, 15, 26] or Chapter 2 and 3 of [16] for the definitions of program function T_P associated with P, (fair) SLD-refutation, as well as correct and computed answers for $P \cup \{Q\}$(Q is a definite goal). We might as well require that $Q\theta$ is ground if θ is a computed answer for $P \cup \{Q\}$. Here we only note that the negation as failure rule can be implemented by fair SLD-resolution. In the following, $gfp(T_P)$ denotes the greatest fix-point of T_P, and ω is the first infinite ordinal.

Theorem 6.1.1 Suppose that P is definite. (1) The least model of P_n is exactly that of P; (2) $T_P \uparrow \omega = T_{P_n} \uparrow \omega$; (3) $gfp(T_P) = gfp(T_{P_n})$; (4) $T_P \downarrow \omega = T_{P_n} \downarrow \omega$.

Proof. (1) Immediate from Theorem 4.3.

(2) Let I be the least model of P. Then $T_P \uparrow \omega = I = T_{P_n} \uparrow \omega$.

(3) $A \in gfp(T_P)$, iff $Comp(P) \cup \{A\}$ has an Herbrand model (See Chapter 3 of [16]), iff $Comp(P_n) \cup \{A\}$ has an Herbrand model(Theorem 5.3.1), iff $A \in gfp(T_{P_n})$.

(4) $A \notin T_P \downarrow \omega$, iff $Comp(P) \models \neg A$ (Chapter 3 of [16]), iff $Comp(P_n) \models \neg A$(Corollary 5.3.2), iff $A \notin T_{P_n} \downarrow \omega$.

Q.E.D.

Actually, we can prove that for any natural number n, $T_P \uparrow n = T_{P_n} \uparrow n$. However, this doesn't hold for $T_P \downarrow n$.

We know that for a ground atom A, (1) $\neg A$ can be inferred from P under the negation as failure rule iff every fair SLD-tree for $P \cup \{\neg A\}$ is finitely failed iff $A \notin T_P \downarrow \omega$; (2) $\neg A$ can be inferred from P under the Herbrand rule iff $A \notin gfp(T_P)$; (3) $\neg A$ can be inferred from P under the CWA iff $A \notin T_P \uparrow \omega$. So, by Theorem 6.1.1, we have

(1) $\neg A$ can be inferred from P under the negation as failure rule iff every fair SLD-tree for $P_n \cup \{\neg A\}$ is finitely failed; (2) $\neg A$ can be inferred from P under the Herbrand rule iff $\neg A$ can be inferred from P_n under the Herbrand rule; (3) $\neg A$ can be inferred from P under the CWA iff $\neg A$ can be inferred from P_n under the CWA.

Theorem 6.1.2 Let Q be a definite goal. θ is a correct answer for $P \cup \{Q\}$ iff θ is a computed answer for $P_n \cup \{Q\}$.

Proof. θ is a correct answer for $P \cup \{Q\}$, iff $P \models \neg Q\theta$, iff $P_n \models \neg Q\theta$(See Theorem 6.1.1(1)), iff θ is a computed answer for $P_n \cup \{Q\}$ (The soundness and completeness of SLD-resolution, see Chapter 2 of [16]).

Q.E.D.

The facts we showed in this section demonstrate that if P is definite, basically various semantics keep invariant. We can utilize its simplified form rather than P itself, to proceed these semantics.

Example 6.1.1 $P = \{p \wedge n \rightarrow z, z \rightarrow r\}$.

The simplified form of P is $P_1 = \{z \rightarrow r\}$. There are no definite goals that succeed through the SLD-resolution from P_1. So no definite goals can succeed through the SLD-resolution from P.

Example 6.1.1 also indicates that in general P and P_n are neither (weakly) subsumption-equivalent nor Herbrand-equivalent[19]. However, they both are completion-equivalent (Theorem 5.3.1), and equivalent with respect to some observables, for instance, successful derivations and computed answers[9].

6.1.2 Hierarchical logic programs

Now, we concentrate on the completeness result of SLDNF-resolution for hierarchical programs. For the definitions of SLDNF-resolution, a safe computation rule *scr*, a correct answer for $Comp(P) \cup \{Q\}$, as well as an *scr*-computed answer for $P \cup \{Q\}(Q$ is a goal), the reader is referred to [8] or Chapter 3 of [16].

To avoid floundering, Clark defined allowed programs and goals[8]. A clause or a goal is said to be allowed if every variable occurring in it occurs in a positive literal of its body. We say that $P \cup \{Q\}$ is allowed if its members are all allowed. This definition is stronger than that defined by Lloyd, et al[16]. We call the latter weakly allowed in the following.

Now assume that P_n is a simplified form of P. Similar to the discussion in Section 5.3.1, we have the following fact:

If P is hierarchical, then P_n is hierarchical.

Theorem 6.1.3 Suppose that P is hierarchical, P_n keeps heads, Q is a goal, and $P_n \cup \{Q\}$ is (weakly) allowed. θ is a correct answer for $Comp(P) \cup \{Q\}$ and θ is a ground substitution for all variables in Q iff θ is an *scr*-computed answer for $P_n \cup \{Q\}$.

Proof. By the above fact, Theorem 5.3.1, as well as the soundness and completeness of SLDNF-resolution for hierarchical programs(Chapter 3 of [16]), we get the result as required.

Q.E.D.

This theorem indicates that if P is hierarchical, generally the results of SLDNF-resolutions from P and its simplified forms keeping heads are invariant. we may utilize simplified forms of P keeping heads instead of P to conduct the SLDNF-resolution without destroying the completeness. On the other hand, this theorem is also a generalization of the completeness result for hierarchical programs. As a matter of fact, it is obvious that if $P \cup \{Q\}$ is allowed, then $P_n \cup \{Q\}$ is allowed. However, if $P \cup \{Q\}$ is weakly allowed, $P_n \cup \{Q\}$ may not be weakly allowed even when P is hierarchical. We consider the following example

$$P = \{p(a) \wedge \neg q_1(x) \rightarrow q_2(x)\}.$$

$P_1 = \{p(a) \rightarrow q_1(x)\}$ is a simplified form of P. Let $Q = \leftarrow q_1(x)$. Then $P \cup \{Q\}$ is weakly allowed. $P_1 \cup \{Q\}$ isn't, since in the P_1-definition of q_1, $p(a) \rightarrow q_1(x)$ is not allowed. Fortunately, in the case when P is hierarchical(or more general, stratified) and P_n keeps heads, $P \cup \{Q\}$ is weakly allowed implies that $P_n \cup \{Q\}$ is weakly allowed. The converse isn't true.

We point out that in Theorem 6.1.3, the results of SLDNF-resolution from P_n doesn't depend on the choice of P_n. in fact, if P'_n is another simplified form of P keeping heads, then $P_n \cup \{Q\}$ is weakly allowed iff $P'_n \cup \{Q\}$ is weakly allowed. And by Theorem 5.3.1, $Comp(P_n)$ and $Comp(P'_n)$ are logically equivalent.

Example 6.1.2 $P = \{p \wedge n \to z, \neg z \to r\}$.

$P_2 = \{r\}$ is a simplified form of P keeping heads. $Q = \neg p \wedge \neg n \wedge \neg z \wedge r \to$ succeeds through the SLDNF-resolution from P_2. So it can succeed through the SLDNF-resolution from P.

Finally, we remark that Clark's completeness result was pushed up to strict stratified programs by Cavedon and Lloyd[6], and to call-consistent programs by Kunen[14]. We can similarly discuss these two classes of programs. Basically, if P is strict and stratified, then so is P_n. If P is call-consistent, then so is P_n. We no longer pay more attention to the details.

6.1.3 Stratified logic programs

We show that if P is stratified, its simplified forms keeping heads keep the standard model of P. For the definition of standard model and levels of predicate symbols for a stratified program, please see [1] or [16]. Hereafter, we assume that P_n is a simplified form of P. P_n is then stratified.

Theorem 6.1.4 Suppose that P is stratified and P_n keeps heads. The standard model of P_n is exactly that of P.

Proof. Since P_n keeps heads, we may assume that a predicate symbol has the same level in P_n and P, and the levels are $0, \cdots, t$. Let A represent a ground atom, and for $k \leq t$,

$$P^k = \{C \in P \mid \text{ The maximum level of predicate symbols in } C \text{ is } k\},$$

$$P_n^k = \{C \in P_n \mid \text{ The maximum level of predicate symbols in } C \text{ is } k\}.$$

Then, in the procedure to obtain P_n from P, $P_n^k = E(P^k)$. Let

$$D_0 = \{S \mid S \subseteq \{A : \text{ the level of } r(A) \text{ is } 0\}\}.$$

P^0 and P_n^0 are definite, and the least fix-point I_n^0 of $T_{P_n^0}$(restricted to D_0) is exactly the least fix-point I^0 of T_{P^0}(restricted to D_0). For $k \geq 0$, suppose that I^k and I_n^k are the least fix-points of T_{P^k} and $T_{P_n^k}$, restricted to D_k, respectively, and $I^k = I_n^k$. Let

$$D_{k+1} = \{I^k \cup S \mid S \subseteq \{A : \text{ the level of } r(A) \text{ is } k+1\}\}.$$

Then, D_{k+1} is a complete lattice under set inclusion. Furthermore, D_{k+1} is a sublattice of the lattice of Herbrand interpretations, and $T_{P^{k+1}}$ and $T_{P_n^{k+1}}$, restricted to D_{k+1}, are both well-defined and monotonic. So, $T_{P^{k+1}}$ and $T_{P_n^{k+1}}$, restricted to D_{k+1}, have the least fix-points I^{k+1} and I_n^{k+1} respectively. Similar to the proof of Lemma 4.2(1) and (2), we have $I^{k+1} = I_n^{k+1}$. We therefore have $I^t = I_n^t$, i.e., the standard models of P and P_n coincide.

Q.E.D.

Example 6.1.3 $P = \{\rightarrow p, p \wedge \neg n \rightarrow r\}$.

$P_1 = \{\rightarrow p, p \rightarrow r\}$ is the simplified form of P keeping heads. The standard model of P_1 is $\{p, r\}$. It's the standard model of P.

According to Theorem 6.1.4, the standard models of P and its simplified forms keeping heads are invariant. We can thus apply Apt, Blair and Walker's interpreter[1] to P_n without any impact on the final results. Finally, we note that if P is locally stratified[21], then P_n is locally stratified, and the perfect models of P and P_n coincide. In fact, for the set $P|H$ of all ground instances of P(May be infinite), when it is viewed as a propositional clause set, we may analogously define the corresponding permutation group $G(P|H)$. And $G(P)$ is a subgroup of $G(P|H)$ up to isomorphism. Similar to the above proof, we can get the result.

6.2 Quasi-definite(hierarchical, stratified) logic programs

We generalize the notions of definite, hierarchical and stratified programs, and propose semantics for them.

6.2.1 Definitions

A program is said to be quasi-definite[quasi-hierarchical, quasi-stratified], if its simplified forms are definite[resp. hierarchical, stratified].

Obviously, definite[hierarchical, stratified] programs are quasi-definite[resp. quasi-hierarchical, quasi-stratified]. Quasi-definite and quasi-hierarchical programs are quasi-stratified.

Example 6.2.1 $P' = \{\neg r \rightarrow z, \neg z \rightarrow r\}$ is quasi-definite, for one of its simplified forms $P_1' = \{\rightarrow r\}$ is definite.

$P = \{\neg r \rightarrow z, p \wedge n \wedge r \rightarrow r\}$ is quasi-definite, since $P_2 = \{\rightarrow z\}$ is a simplified form of P, and it is definite.

Example 6.2.2 $P = \{z \rightarrow r, \neg z \rightarrow r, p \wedge n \wedge z \rightarrow z\}$ is quasi-hierarchical, for $P_1 = \{z \rightarrow r, \neg z \rightarrow r\}$ is a simplified form of P, which is hierarchical.

Example 6.2.3 $P = \{\neg p \rightarrow n, \neg n \rightarrow p, r \wedge \neg p \wedge \neg n \rightarrow z, z \rightarrow r\}$ is quasi-stratified, because $P_1 = \{\rightarrow p, r \wedge \neg p \rightarrow z, z \rightarrow r\}$ is a simplified form of P, which is stratified.

6.2.2 Quasi-definite logic programs

We follow the terminologies for definite programs in Section 6.1.1. By A we represent a ground atom. Let P_n be a simplified form of P. If P is quasi-definite, then P_n is definite. Without confusions, we also use $Fix(P)$ to denote the set $\{A \mid r(A) \in Fix(P)\}$. Then by Lemma 5.1.1, we have

Theorem 6.2.1 Let P be quasi-definite. The followings are the same: (1)$\{A \mid P \models A\}$; (2)$T_{P_n} \uparrow \omega \cap Fix(P)$; (3) $I_n \cap Fix(P)$, where I_n is the least model of P_n; (4) $\{A|$ there is an SLD-refutation from $P_n \cup \{\rightarrow A\}\} \cap Fix(P)$.

Let Q be a definite goal. We call a substitution θ a correct answer for $P \cup \{Q\}$, if $P \models \neg Q\theta$, and we require that θ makes Q ground. From Theorem 6.2.1, we have

Theorem 6.2.2 Let P be quasi-definite, and Q a definite goal. θ is a correct answer for $P \cup \{Q\}$ iff θ is a computed answer for $P_n \cup \{Q\}$, and $R(Q) \subseteq Fix(P)$.

So, the SLD-resolution is sound and complete for quasi-definite programs and definite goals. we may employ the SLD-resolution to compute definite goals in this case. And it is independent of the choice of P_n.

Now we turn to deriving negative facts. Let P be quasi-definite. First we choose some simplified forms, say $P_n^0 = P_n, \cdots, P_n^k$, of P according to some given requirements(for example, keeping heads). We say that $\neg A$ is derivable (under the requirements) if every fair SLD-tree for $P_n^i \cup \{A \rightarrow\} (i = 0, \cdots, k)$ is finitely failed.

When P is definite, this rule coincides with the negation as failure rule.

Let $\sigma_i \in G(P)$ such that $\sigma_i(P_n^i) = P_n (i = 0, \cdots, k)$. Then $\neg A$ is derivable iff every fair SLD-tree for $P_n \cup \{\sigma_i(A) \rightarrow\} (i = 0, \cdots, k)$ is finitely failed.

Therefore this rule can be implemented through fair SLD-resolution from P_n. It is sound and complete with respect to $Comp(P_n)$. As a matter of fact, $\neg A$ is derivable iff $Comp(P_n) \models \wedge_{i=0}^k \sigma_i(A)$.

$P \cup \{\neg A \mid \neg A$ is derivable$\}$ is consistent. Indeed, $Comp(P_n)$ is consistent. So $P_n \cup \{\neg A \mid \neg A$ is derivable$\}$ is consistent. Let I be an Herbrand model of it. Then I is a model of P_n, and $A \notin I$ if $\neg A$ is derivable. Suppose that $I_n \subseteq I$ is a minimal model of P_n. Then $A \notin I_n$ if $\neg A$ is derivable, and by Theorem 4.3, I_n is a model of P. Thus it is a model of $P \cup \{\neg A \mid \neg A$ is derivable$\}$.

Example 6.2.4 For the P in Example 6.2.1, no positive atoms can succeed through the SLD-resolution from P since $Fix(P) = \emptyset$. Now we choose a simplified form of P: $P_1 = \{z\}$ (it keeps heads) . Then $\neg p, \neg n$ and $\neg r$ are derivable. However, although $\neg r$ is a logic consequence of $Comp(P)$, generally we can by no means obtain $\neg r$ through the SLDNF-resolution from P.

6.2.3 Quasi-hierarchical and quasi-stratified logic programs

Let H be the set of all predicate symbols appearing in the heads of some clauses in P, $O_1, \cdots, O_k$ be all the orbits of P whose intersections with H are non-empty, and

$$G_H = S_{O_1 \cap H} \times \cdots \times S_{O_k \cap H}.$$

G_H is then a subgroup of $G(P)$. If $Q = \wedge_i L_i \rightarrow$ is a goal, we use $G_H(Q)$ to denote the goal $\wedge_i \wedge_{\sigma \in G_H} \sigma(L_i) \rightarrow$. Now, suppose that P_n is a simplified form of P keeping heads. We say that a substitution θ is a correct answer for $Comp(P) \cup \{Q\}$ in symmetric sense, if it is a correct answer for $Comp(P_n) \cup \{G_H(Q)\}$. A substitution θ is called an *scr*-computed answer for $P \cup \{Q\}$ in symmetric sense, if it is an *scr*-computed answer for $P_n \cup \{G_H(Q)\}$.

Actually, θ is a correct answer for $Comp(P) \cup \{Q\}$ in symmetric sense[an *scr*-computed answer for $P \cup \{Q\}$ in symmetric sense] iff for every P_n keeping heads, it is a correct answer for $Comp(P_n) \cup \{Q\}$[resp. *scr*-computed answer for $P_n \cup \{Q\}$]. It seems reasonable to define correct answers by utilizing $Comp(P_n)$. According to Theorems 4.3 and 4.6, we may say that P_n keeps the minimal models of P, and $Comp(P_n)$ and $Comp(P)$ are logically equivalent up to permutations. When P is hierarchical or stratified, $G_H = (1)$, and by Theorem 5.3.1, these two definitions coincide with the usual ones.

If $P \cup \{Q\}$ is allowed, then so is $P_n \cup \{G_H(Q)\}$. We then have

Theorem 6.2.3 Suppose that P is quasi-hierarchical, Q is a goal, and $P \cup \{Q\}$ is allowed. θ is a correct answer for $Comp(P) \cup \{Q\}$ in symmetric sense and θ is a ground substitution for all variables in Q iff θ is an *scr*-computed answer for $P \cup \{Q\}$ in symmetric sense.

Quasi-hierarchical programs may allow some recursion. Theorem 6.2.3 is a generalization of Clark's completeness result of the SLDNF-resolution for hierarchical programs. It isn't covered by the other generalizations such as in [6] and [14]. Unfortunately, P being weakly allowed does not ensure that P_n is. So the completeness part of this theorem may not hold if we just require P to be weakly allowed.

Example 6.2.5, For the P in Example 6.2.2, it has one simplified form $P_1 = \{z \rightarrow r, \neg z \rightarrow r\}$. Of course P_1 keeps heads. r is a logic consequence of $Comp(P)$. However, $r \rightarrow$ cannot succeed through the SLDNF-resolution from P. Namely, the identity substitution is not computed for $P \cup \{r \rightarrow\}$. But $r \rightarrow$ succeeds through the SLDNF-resolution from P_1. So now the identity substitution is computed.

Now, we take a look at quasi-stratified programs. By Theorem 4.6, we have the following

Theorem 6.2.4 If P is quasi-stratified, then $Comp(P)$ is consistent.

This theorem extends the result that $Comp(P)$ is consistent for stratified programs. The latter result was ever generalized to call-consistent programs[24]. We remark that it doesn't cover Theorem 6.2.4, or vise visa. Let's consider

$$P = \{\neg p \wedge \neg z \rightarrow n, \neg n \wedge \neg z \rightarrow p, \neg p \wedge \neg n \rightarrow z\}.$$

P is quasi-stratified. But it isn't call-consistent. On the other hand,

$$P = \{\neg p \wedge n \rightarrow n, \neg n \wedge p \rightarrow p\}$$

is call-consistent. However, it isn't quasi-stratified.

Although a quasi-stratified program may not have the unique standard model, Apt, Blair and Walker's interpreter[1] does apply.

Let P be quasi-stratified, and P_n a simplified form of P keeping heads. We call the standard models of $\sigma(P_n)(\sigma \in G_H)$ the standard models of P.

It is reasonable to utilize these models as the intended meaning. For a ground atom A, we say $P \models_{SM} A$ if A is in the intersection of standard models of P, $P \models_{SM} \neg A$ if A isn't in the union of standard models of P.

When P is stratified, it coincides with the usual standard model semantics. Also, we needn't test for all simplified forms of P keeping head. We only need to fix one. As a matter of fact, we have

Theorem 6.2.5 Suppose that P is quasi-stratified, P_n keeps heads, and L is a ground literal. $P \models_{SM} L$ iff for any $\sigma \in G_H$, $P_n \models_{SM} \sigma(L)$.

Example 6.2.6 For the P in Example 6.2.3, $P_1 = \{\rightarrow p, r \wedge \neg p \rightarrow z, z \rightarrow r\}$ is a simplified form of P keeping heads. $G_H = \{(pn), (1)\}$. $G_H(z) = \{z\}$, and $P_1 \models_{SM} \neg z$. So $P \models_{SM} \neg z$. Similarly, $P \models_{SM} \neg r$.

Bachmair and Ganzinger showed that the perfect model semantics can be defined for stratified programs up to redundancy[3]. We point out that the notion of being quasi-stratified isn't covered by that of being stratified up to redundancy, or vise visa. Let

$$P = \{p \wedge n \rightarrow z, \neg p \wedge \neg z \rightarrow n\}.$$

Then P is quasi-stratified. But it isn't stratified up to redundancy. However,

$$P = \{\neg p \rightarrow p\}$$

is stratified up to redundancy. it isn't quasi-stratified.

7 Conclusions

In this paper, we first discussed structures of the symmetric group corresponding to a symmetry in a program. Then we presented the notions of symmetric group and simplified forms of a program based on a sequence of such symmetries.

Actually, these notions were due to the investigation of minimal models of a program and models of its completion. As is known, they play a central role in the theory of logic programming. We showed the relationships between the minimal models of a program and its simplified forms, and the relationships between the models of the completions of a program and its simplified forms.

We then focus on the applications to derivations of negative information and semantic issues. Roughly speaking, the CWA, the GCWA, and the completion procedure can be applied directly to the simplified forms instead of the original program(For the completion procedure, we have to require that the simplified forms keep heads). A definite program and its simplified form have the same least model and procedural semantics(The latter means SLD-resolution). A hierarchical program and its simplified forms keeping heads have the invariant procedural semantics(SLDNF-resolution). And a stratified program and its simplified forms keeping heads have the invariant standard or perfect model semantics.

We also introduced some new concepts based on these symmetries. We presented a new rule to assume negative information termed OCWA, which is in fact a generalization of the GCWA. We defined three classes of programs called respectively quasi-definite, quasi-hierarchical and quasi-stratified programs, which are more general than definite, hierarchical and stratified programs. Finally, we briefly described the similar model and procedural semantics for quasi-definite programs, procedural semantics for quasi-hierarchical programs, and model semantics for quasi-stratified programs.

To sum up, the considerations of these symmetries may simplify the related computation procedures, increase the computational power, and lead to new concepts. Of course, the price is to compute some symmetric groups. One future work is to study the complexity issues.

Before closing the paper, we pose the following problems. The first one is indeed to see to what extent the positions of atoms in a program affect the results. The second is to investigate other symmetries in a program as well as their applications.

We want to figure out the behaviors of goals with respect to the completion set

$$\{Comp(\cup_{C\in P}\{\sigma_C(C)\}) \mid \sigma_C \in G_C\}.$$

For example, for $P = \{\rightarrow p, p \wedge \neg q_1 \rightarrow q_2\}$, and goal $p \rightarrow$, the SLDNF-resolution procedure doesn't depend on the positions of q_1 and q_2. So, in this situation we can say: "Don't worry, it doesn't matter to choose whom as the negative hypothesis".

One shortcoming of the symmetric group we defined in the paper is that in many cases it is trivial, which means that sometimes we probably restrict too much. This requires us to investigate more general symmetries. As a matter of fact, a program induces a number of symmetric group structures, which can be roughly classified into the syntactic and semantic ones. Besides the symmetric group G associated with P we defined in the paper, for instance,

$$G_1 = \{\sigma \in S_R \mid \forall C \in P, \exists C' \in P : \sigma(M_C) = M_{C'}\}$$

is another syntactic symmetric group induced by P, where M_C and $M_{C'}$ are the literal multisets of C and C' respectively. And $G \leq G_1$(i.e., G is a subgroup of G_1). The followings are some examples belonging to the category of semantic symmetric groups induced by P.

$$\begin{aligned}
G_2 &= \{\sigma \in S_R \mid \forall C \in P, \exists C' \in P : \sigma(C) \text{ is logically equivalent to } C'\},\\
G_3 &= \{\sigma \in S_R \mid I \text{ is a model of } P \Rightarrow \sigma(I) \text{ is a model of } P\},\\
G_4 &= \{\sigma \in S_R \mid I \text{ is a minimal model of } P \Rightarrow \text{ so is } \sigma(I)\},\\
G_5 &= \{\sigma \in S_R \mid Comp(P) \text{ is logically equivalent to } Comp(\sigma(P))\}.
\end{aligned}$$

It is not hard to see that $G \leq G_1 \leq G_2 \leq G_3 \leq G_4$. As an example, let's think of

$$P = \{q_1 \to p, q_2 \to p, \neg q_1 \to q_2\}[4].$$

For this P, $G = (1)$. However, $G_1 = G_2 = G_3 = G_4 = \{(1), (q_1 q_2)\}$, and $G_5 = \{(1), (p q_2)\}$. The structures of $G_i (i \geq 1)$ are more complicated than G. These symmetric groups keep some syntactic or semantic properties invariant. It is interesting to explore the behaviors of a program under them or their combinations. We believe this is meaningful.

On the other hand, we needn't limit ourselves to the symmetries on predicate symbols. We can also consider, for instance, those on function symbols and constants, and look the actions of them or their combinations. Let's see the example

$$P = \{\neg p(a) \wedge \neg p(b) \wedge \neg q(a) \to q(b)\}.$$

the symmetries on predicate and function symbols together result in a simplified form $P_1 = \{\to q(b)\}$. P_1 is definite, all minimal models of P can be obtained from those of P_1, and $Comp(P)$ is logically equivalent to $Comp(P_1)$ (up to permutations).

In fact, if we similarly define the symmetric group on function symbols and simplified forms of a program, generally Theorem 4.3 and 4.6 do not hold anymore. One obvious counter-example is

$$P = \{p(a) \wedge p(b) \to q, \to p(x)\}.$$

However, if P is a ground clause set, all the similar conclusions hold.

It is also interesting to investigate the applications of symmetries to mechanical theorem proving[7]. For example, by Theorem 4.3, a clause set is unsatisfiable iff a simplified form of it is unsatisfiable. It may help to decide the SAT problem in practice.

References

1. Apt, K. R., Blair, H. A. & Walker, A., Towards a Theory of Declarative Knowledge, in: *Foundations of Deductive Databases and Logic Programming, Minker, J. (ed.)*, Morgan Kaufmann, Los Altos, 1988, 89-148.

2. Apt, K. R., Bol, R. N., Logic Programming and Negation: A Survey, *J. Logic Programming*, 19/20, 1994, 9-71.

3. Bachnair, L., Ganzinger, H., Perfect Model Semantics for Logic Programs with Equality, Proc. Int. Conf. on Logic Programming, 1991, 645-659.

4. Baral, C., Lobo, J., & Minker, J., Generalized Well-Founded Semantics for Logic Programs, Proc. 10th Int. Conf. on Automated Deduction, LNAI 449, Springer-Verlag, 1990, 102-116.

5. Bossu, G., Siegel, P., Saturation,Nonmonotonic Reasoning and the Closed World Assumption, *Artificial Intelligence*, 25, 1985, 13-63.

6. Cavedon, L., Lloyd, J. W., A Completeness Theorem for SLDNF resolution, *J. Logic Programming*, 7(3), 177-191, 1989.

7. Chang, C.-L., Lee, R. C., *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.

8. Clark, K. L., Negation as failure, in: *Readings in Nonmonotonic Reasoning, M. L. Ginsberg (ed.)*, Morgan Kaufmann, Los Altos, 1987, 311-325.

9. Gabbrielli, M., Levi, G. & Meo, M. C., Observational Equivalences for Logic Programs, *Proceedings of the Joint International Conference and Symposium on Logic Programming(JICSLP-92)*, MIT press, 1992, 131-145.

10. Gelfond, M., Przymusinska, H., Negation as Failure: Careful Closure of Procedure, *Artificial Intelligence* 30, 1986, 273-287.

11. Grant, J., Minker, J., Answering Queries in Indefinite Databases and the Null Value Problem, *Advances in Computing Theory(Kanellakis, Guest Ed)*, JAI Press, 1986, 247-267.

12. Henschen, L., Park, H., Compiling the GCWA in Indefinite Deductive Databases, in: *Foundations of Deductive Databases and Logic Programming, Minker, J. (ed.)*, Morgan Kaufmann, Los Altos, 1988, 395-438.

13. Kowalski, R. A., Predicate Logic as a Programming Language, *Information Processing*, 74, 1974, 569-574.

14. Kunen, K., *Negation in Logic programming*, J. Logic Programming, 4, 1987, 289-308.

15. Lassez, J.-L., Maher, M. J., Closures and Fairness in the Semantics of Programming Logics, *Theoretical Computer Science*, 29, 1984, 167-184.

16. Lloyd, J. W., *Foundations of Logic Programming*, Springer-Verlag, Berlin Heidelberg New York London Paris Tokyo, 1987

17. Lloyd, J. W., Toper, R. W., A Basis for Deductive Databases Systems II, *J. Logic Programming*, 3, 1986, 55-67.

18. Lukaszewicz, W., *Non-Monotonic Reasoning - Formalization of Commonsense Reasoning*, Ellis Horwood, 1990.

19. Maher, M. J., Equivalences of Logic Programs, in:*Foundations of Deductive Databases and Logic Programming, Minker, J. (ed.)*, Morgan Kaufmann, Los Altos, 1988, 627-658.

20. Minker, J., On Indefinite Databases and the Closed World Assumption, in: *Readings in Nonmonotonic Reasoning, M. L. Ginsberg (ed.)*, Morgan Kaufmann, Los Altos, 1987, 326-333.

21. Przymusinski, T. C., On the Declarative Semantics of Deductive Databases and Logic Programs, in: *Foundations of Deductive Databases and Logic Programming, Minker, J. (ed.)*, Morgan Kaufmann, Los Altos, 1988, 193-216.

22. Reiter, R., On Closed World Data Bases, in: *Readings in Nonmonotonic Reasoning, M. L. Ginsberg (ed.)*, Morgan Kaufmann, Los Altos, 1987, 300-310.

23. Rotman, J. J., *An Introduction to the Theory of Groups*, Springer-Verlag, New York, 1994.

24. Sato, T., *On the Consistency of First Order Logic Programs*, Technical Report 87-12, Electrotechnical Laboratory, Ibarki, Japan, 1987.

25. Shepherdson, J. C., Negation as Failure, Completion and Stratification, in: *Handbook of Logic in Artificial Intelligence and Logic Programming, Dov M. Gabbay et al (eds.)*, Vol.5, 1998, 356-419.

26. Van Emden, M. H., Kowalski, R. A., The Semantics of Predicate Logic as a Programming Language, *J. ACM*, 23, 1976, 733-742.

27. Van Gelder, A., Negation as Failure Using Tight Derivations for General Logic Programs, in: *Foundations of Deductive Databases and Logic Programming, Minker, J. (ed.)*, Morgan Kaufmann, Los Altos, 1988, 149-176.

I N F O R M A T I K

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Birgit Hofmann
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-1999-2-004	V. Cortier, H. Ganzinger, F. Jacquemard, M. Veanes	Decidable fragments of simultaneous rigid reachability
MPI-I-1999-2-003	U. Waldmann	Cancellative Superposition Decides the Theory of Divisible Torsion-Free Abelian Groups
MPI-I-1999-2-001	W. Charatonik	Automata on DAG Representations of Finite Trees
MPI-I-1999-1-002	N.P. Boghossian, O. Kohlbacher, H.-. Lenhof	BALL: Biochemical Algorithms Library
MPI-I-1999-1-001	A. Crauser, P. Ferragina	A Theoretical and Experimental Study on the Construction of Suffix Arrays in External Memory
MPI-I-98-2-018	F. Eisenbrand	A Note on the Membership Problem for the First Elementary Closure of a Polyhedron
MPI-I-98-2-017	M. Tzakova, P. Blackburn	Hybridizing Concept Languages
MPI-I-98-2-014	Y. Gurevich, M. Veanes	Partisan Corroboration, and Shifted Pairing
MPI-I-98-2-013	H. Ganzinger, F. Jacquemard, M. Veanes	Rigid Reachability
MPI-I-98-2-012	G. Delzanno, A. Podelski	Model Checking Infinite-state Systems in CLP
MPI-I-98-2-011	A. Degtyarev, A. Voronkov	Equality Reasoning in Sequent-Based Calculi
MPI-I-98-2-010	S. Ramangalahy	Strategies for Conformance Testing
MPI-I-98-2-009	S. Vorobyov	The Undecidability of the First-Order Theories of One Step Rewriting in Linear Canonical Systems
MPI-I-98-2-008	S. Vorobyov	AE-Equational theory of context unification is Co-RE-Hard
MPI-I-98-2-007	S. Vorobyov	The Most Nonelementary Theory (A Direct Lower Bound Proof)
MPI-I-98-2-006	P. Blackburn, M. Tzakova	Hybrid Languages and Temporal Logic
MPI-I-98-2-005	M. Veanes	The Relation Between Second-Order Unification and Simultaneous Rigid E-Unification
MPI-I-98-2-004	S. Vorobyov	Satisfiability of Functional+Record Subtype Constraints is NP-Hard
MPI-I-98-2-003	R.A. Schmidt	E-Unification for Subsystems of S4
MPI-I-98-2-002	F. Jacquemard, C. Meyer, C. Weidenbach	Unification in Extensions of Shallow Equational Theories
MPI-I-98-1-031	G.W. Klau, P. Mutzel	Optimal Compaction of Orthogonal Grid Drawings
MPI-I-98-1-030	H. Brönniman, L. Kettner, S. Schirra, R. Veltkamp	Applications of the Generic Programming Paradigm in the Design of CGAL
MPI-I-98-1-029	P. Mutzel, R. Weiskircher	Optimizing Over All Combinatorial Embeddings of a Planar Graph

MPI-I-98-1-028	A. Crauser, K. Mehlhorn, E. Althaus, K. Brengel, T. Buchheit, J. Keller, H. Krone, O. Lambert, R. Schulte, S. Thiel, M. Westphal, R. Wirth	On the performance of LEDA-SM
MPI-I-98-1-027	C. Burnikel	Delaunay Graphs by Divide and Conquer
MPI-I-98-1-026	K. Jansen, L. Porkolab	Improved Approximation Schemes for Scheduling Unrelated Parallel Machines
MPI-I-98-1-025	K. Jansen, L. Porkolab	Linear-time Approximation Schemes for Scheduling Malleable Parallel Tasks
MPI-I-98-1-024	S. Burkhardt, A. Crauser, P. Ferragina, H. Lenhof, E. Rivals, M. Vingron	q-gram Based Database Searching Using a Suffix Array (QUASAR)
MPI-I-98-1-023	C. Burnikel	Rational Points on Circles
MPI-I-98-1-022	C. Burnikel, J. Ziegler	Fast Recursive Division
MPI-I-98-1-021	S. Albers, G. Schmidt	Scheduling with Unexpected Machine Breakdowns
MPI-I-98-1-020	C. Rüb	On Wallace's Method for the Generation of Normal Variates
MPI-I-98-1-019		2nd Workshop on Algorithm Engineering WAE '98 - Proceedings
MPI-I-98-1-018	D. Dubhashi, D. Ranjan	On Positive Influence and Negative Dependence
MPI-I-98-1-017	A. Crauser, P. Ferragina, K. Mehlhorn, U. Meyer, E. Ramos	Randomized External-Memory Algorithms for Some Geometric Problems
MPI-I-98-1-016	P. Krysta, K. Loryś	New Approximation Algorithms for the Achromatic Number
MPI-I-98-1-015	M.R. Henzinger, S. Leonardi	Scheduling Multicasts on Unit-Capacity Trees and Meshes
MPI-I-98-1-014	U. Meyer, J.F. Sibeyn	Time-Independent Gossiping on Full-Port Tori
MPI-I-98-1-013	G.W. Klau, P. Mutzel	Quasi-Orthogonal Drawing of Planar Graphs
MPI-I-98-1-012	S. Mahajan, E.A. Ramos, K.V. Subrahmanyam	Solving some discrepancy problems in NC*
MPI-I-98-1-011	G.N. Frederickson, R. Solis-Oba	Robustness analysis in combinatorial optimization
MPI-I-98-1-010	R. Solis-Oba	2-Approximation algorithm for finding a spanning tree with maximum number of leaves
MPI-I-98-1-009	D. Frigioni, A. Marchetti-Spaccamela, U. Nanni	Fully dynamic shortest paths and negative cycle detection on diagraphs with Arbitrary Arc Weights
MPI-I-98-1-008	M. Jünger, S. Leipert, P. Mutzel	A Note on Computing a Maximal Planar Subgraph using PQ-Trees
MPI-I-98-1-007	A. Fabri, G. Giezeman, L. Kettner, S. Schirra, S. Schönherr	On the Design of CGAL, the Computational Geometry Algorithms Library
MPI-I-98-1-006	K. Jansen	A new characterization for parity graphs and a coloring problem with costs
MPI-I-98-1-005	K. Jansen	The mutual exclusion scheduling problem for permutation and comparability graphs
MPI-I-98-1-004	S. Schirra	Robustness and Precision Issues in Geometric Computation
MPI-I-98-1-003	S. Schirra	Parameterized Implementations of Classical Planar Convex Hull Algorithms and Extreme Point Compuations
MPI-I-98-1-002	G.S. Brodal, M.C. Pinotti	Comparator Networks for Binary Heap Construction
MPI-I-98-1-001	T. Hagerup	Simpler and Faster Static AC^0 Dictionaries
MPI-I-97-2-012	L. Bachmair, H. Ganzinger, A. Voronkov	Elimination of Equality via Transformation with Ordering Constraints
MPI-I-97-2-011	L. Bachmair, H. Ganzinger	Strict Basic Superposition and Chaining
MPI-I-97-2-010	S. Vorobyov, A. Voronkov	Complexity of Nonrecursive Logic Programs with Complex Values
MPI-I-97-2-009	A. Bockmayr, F. Eisenbrand	On the Chvátal Rank of Polytopes in the 0/1 Cube