Vol. 45 No. 4 SCIENCE IN CHINA (Series F) August 2002

FC-normal and extended stratified logic program

XU Daoyun (许道云) & DING Decheng (丁德成)

Department of Mathematics, Nanjing University, Nanjing 210093, China
Correspondence should be addressed to Xu Daoyun (email: dgqqxdy@163.net)

Received July 25, 2001; revised January 13, 2002

Abstract This paper investigates the consistency property of FC-normal logic program and presents an equivalent deciding condition whether a logic program P is an FC-normal program. The deciding condition describes the characterizations of FC-normal program. By the Petri-net presentation of a logic program, the characterizations of stratification of FC-normal program are investigated. The stratification of FC-normal program motivates us to introduce a new kind of stratification, extended stratification, over logic program. It is shown that an extended (locally) stratified logic program is an FC-normal program. Thus, an extended (locally) stratified logic program has at least one stable model. Finally, we have presented algorithms about computation of consistency property and a few equivalent deciding methods of the finite FC-normal program.

Keywords: **forward chaining, FC-normal program, stable model, extended stratification.**

Marek's forward-chaining construction is one of important techniques for investigating the nonmonotonic reasoning. In refs. [1, 2], Marek et al. showed that a forward-chaining technique, supplemented by a properly chosen safeguard, can be used to construct stable models of a logic program. By the introduction of consistency property over a logic program, they proposed a class of logic programs, FC-normal programs, each of which has at least one stable model. In the process of construction of stable models, an FC-normal program has the property that an applied nonmomotonic clause C would not be negated by the applications of clauses later. The property ensures intuitively the existence of stable models of a logic program. And we can construct a stable model by the simplified forward-chaining construction, normal forward-chaining construction, if a logic program P is FC-normal. Please note that an FC-normal program is associated with a consistency property.

Given a logic program P, we can always construct a consistency property Con over P. But it does not hold that P is FC-normal with respect to every consistency property Con over P. We can also construct two consistency properties Con_1 and Con_2 over P such that P is FC-normal with respect to Con_1 and P is not FC-normal with respect to Con_2.

However, we can construct a consistency property $Con_{norm(P)}$, called normal consistency property, by the normal forward-chaining construction for all well-ordering over $nmon(P)$ such that

if P is FC-normal with respect to a consistency property Con, then $Con_{norm(P)} \subseteq Con$ and P is FC-normal with respect to $Con_{norm(P)}$.

It is that for a given logic program P we only need to decide whether P is FC-normal with respect to $Con_{norm(P)}$. Please note that $Con_{norm(P)}$ is unique for a given logic program P.

In this paper, we analyse the consistency property over a logic program and present an equivalent deciding condition whether a logic program P is FC-normal. The new deciding condition shows that if a nonmonotonic clause C_1 is applied at stage α in the construction, and another nonmonotonic clause C_2 is applied at stage β, where $\alpha \prec \beta$ and $\prec$ is a well-ordering over $nmon(P)$, then the application of C_2 would not negate the application of C_1. In the construction the application of any nonmonotonic clause C in P could block applications of all nonmonotonic clauses that would negate the application of C later.

For the stratification of a logic program, Apt et al.[3] and Przymusinski[4] have introduced a (locally) stratification over a logic program by defining a rank function $rank$: for every clause $a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$ $rank(a_i) \leqslant rank(a)$ for all $1 \leqslant i \leqslant m$, and $rank(b_j) < rank(a)$ for all $1 \leqslant j \leqslant k$. The conditions imposed on the rank function $rank$ over a logic program P protect from the occurrence of the nonmonotonic deduction cycles over P. By the Petri-net representation $N(P)$ of P[6], it follows that $N(P)$ contains no nonmonotonic cycles. Thus, a (locally) stratified program has the unique stable model.

In this paper, we introduce a new stratification called extended stratification by defining a new rank function $rank^*$. The conditions imposed on $rank^*$ are weaker than that on $rank$, i.e. a (locally) stratified program is extended (locally) stratified and an extended (locally) stratified program may not be (locally) stratified. In an extended (locally) stratified program P, it is allowed that $N(P)$ contains nonmonotonic cycles. We will prove that an extended (locally) stratified program is FC-normal and present an example to show that an FC-normal program may not be extended (locally) stratified.

Finally, we present algorithms to compute the normal consistency property over a finite logic program, and consider a few equivalent deciding methods of the finite FC-normal program. For our algorithms, the worst time complexity, deciding whether a finite logic program is FC-normal, is still in exponential time.

1 Preliminaries

In this section, we introduce some necessary notions and notations. It is supposed that we discuss atoms underlying language $\mathcal{L}$ and deal with the propositional case only.

Definition 1.1. (i) A definite logic program P consists of clauses of form $a \leftarrow a_1, \ldots, a_m$, where $a, a_1, \ldots, a_m$ are atoms of language $\mathcal{L}$. We call such clauses Horn program clauses or simply Horn clauses. The set of atoms occurring in clauses of P is called the Herbrand base of P, and is denoted by H_P.

(ii) A subset $M \subseteq H_P$ is called a model of a set P of program clauses if for all clauses $a \leftarrow a_1, \ldots, a_m$ of P, $a_1, \ldots, a_m \in M$ implies $a \in M$.

Definition 1.2. (i) A general logic program P consists of clauses of form

$$C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k,$$

where $a, a_1, \ldots, a_m, b_1, \ldots, b_k$ are atoms in language $\mathcal{L}$. Here $a_1, \ldots, a_m$ are called the premises of clause C, $b_1, \ldots, b_k$ are called the constraints of clause C and a is called the conclusion of clause C. We shall write $prem(C) = \{a_1, \ldots, a_m\}$, $cons(C) = \{b_1, \ldots, b_k\}$ and $c(C) = \{a\}$. Either one of $prem(C)$, $cons(C)$, or both may be empty. If $prem(C) = cons(C) = \varnothing$, then the clause C is called an axiom.

(In this paper, a logic program means a general logic program.)

(ii) Let H_P be the set of atoms occurring in clauses of P. A subset M of H_P is called a model of P if for any clause $a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$ of P, whenever the premises $a_1, \ldots, a_m$ of C are in M and the constraints $b_1, \ldots, b_k$ of C are not in M, the conclusion a of C belongs to M.

For a given general program P, let $mon(P)$ denote the set of all Horn clauses in P and $nmon(P) = P \setminus mon(P)$. The elements of $nmon(P)$ are called nonmonotonic clauses. And the monotone operator of derivation closure under the clauses in $mon(P)$ is denoted by $cl_{mon(P)}$.

For a clause $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$, the Horn clause $a \leftarrow a_1, \ldots, a_m$ is called the Horn projection of C, denoted by C_{Horn}. Let P be a logic program. Then the Horn program $mon(P) \cup \{C_{Horn} | C \in nmon(P)\}$ is called Horn projection of P, denoted by P_{Horn}.

Given a logic program P, sets $M \subseteq H_P$ and $I \subseteq H_P$, a M-deduction of c from I in P is a finite sequence $< c_1, c_2, \ldots, c_l >$ such that $c_l = c$ and for all $1 \leqslant i \leqslant l$, each c_i either

(i) $c_i \in I$, or

(ii) $c_i = c(C)$ for some axiom $C \in P$, or

(iii) $c_i = c(C)$ for some clause $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$ in P such that $\{a_1, \ldots, a_m\} \subseteq \{c_1, \ldots, c_{i-1}\}$ and $\{b_1, \ldots, b_k\} \cap M = \varnothing$.

For the case (iii), we say that the clause $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$ is applied in M-deduction of c from I in P. Especially, we say that $< c_1, c_2, \ldots, c_l >$ is a proof sequence of c from I in P if all applied clauses in the deduction are in $mon(P)$, i.e. all applied clauses are monotonic clauses in P.

For $c \in H_P$, we say that c is M-deducible from I over P if there is a M-deduction of c from I over P. We denote $C_M(I) = \{c |\ c$ is M-deducible from I over $P\}$. M is called a stable model of P from I if $C_M(I) = M$. Especially, M is called a stable model of P if $C_M(\varnothing) = M$.

Moreover, we present a kind of Petri-net representation for a logic program to describe the characterization of FC-normal program.

Given a logic program P, every non-axiom clause

$$C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$$

in P is associated to a transition t_C as given in fig. 1, where an arrow with a small circle corresponds to a constraint b_i in the clause C. It means that the clause C is not applicable whenever b_i is deducible.

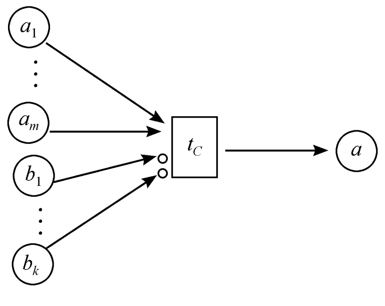

Fig. 1

Define notations:

${}^\bullet t_C = \{a_1, \ldots, a_m\}$, ${}^\circ t_C = \{b_1, \ldots, b_k\}$ and $t_C^\bullet = \{a\}$.

Generally, we assume that ${}^\bullet t_C \cap {}^\circ t_C = \varnothing$.

Definition 1.3. Let P be a (propositional) logic program and let N be a Petri-net with a set S of states and a set T of transitions. N is called the Petri-net representation of P if the following conditions hold:

(i) $S = H_P$ and $T = \{t_C \mid C$ is a non-axiom clause in $P\}$.

(ii) A state $s \in prem(C)$ if and only if an arrow from s to t_C occurring in N.

(iii) A state $s \in cons(C)$ if and only if an arrow with a small circle from s to t_C occurring in N.

(iv) A state $s \in c(C)$ if and only if an arrow from t_C to s occurring in N.

We denote the Petri-net representation of P by $N(P)$. The set $\{a \mid a \leftarrow \in P\}$ is the set of initial states (initial active conditions) of $N(P)$, denoted by I_N.

Let $N(P)$ be the Petri-net representation of a logic program P. In $N(P)$, a cycle is a sequence: $C' = a'_1t'_1a'_2t'_2 \ldots a'_pt'_pa'_1$, where $t'_1, \ldots, t'_p$ are transitions, $a'_i \in^{\bullet} t'_i \cup^{\circ} t'_i (1 \leqslant i \leqslant p)$, $a'_{i+1} \in t'^{\bullet}_i (1 \leqslant i < p)$ and $a'_1 \in t'^{\bullet}_p$. If $a'_i \in^{\bullet} t'_i$ for each $1 \leqslant i \leqslant p$, then C' is called a monotonic cycle. A cycle is called a nonmonotonic cycle if the cycle is not monotonic. In this paper, we always assume a cycle to be a simple cycle.

Example 1.1. The logic program P consists of clauses $\{a \leftarrow,\ b \leftarrow c,\ c \leftarrow b,\ c \leftarrow a, \neg d,\ e \leftarrow c, \neg f,\ d \leftarrow e, \neg f\}$.

In fig. 2, a is the initial state of Petri-net $N(P)$, t_3 is first applied (fired) transition and t_5 has to be applied. On the other hand, the application of t_5 must negate the application of t_3. This is a contradiction. Thus, P has no stable model.

Moreover, the sequence ct_1bt_2c is a monotonic cycle and $ct_4et_5dt_3c$ is a nonmonotonic cycle in fig. 2.

Example 1.2. The logic program P consists of clauses $\{a \leftarrow,\ b \leftarrow c,\ c \leftarrow b,\ c \leftarrow a, \neg d,\ e \leftarrow c, \neg f,\ f \leftarrow c, \neg e\}$.

In fig. 3, a is the initial state of Petri-net $N(P)$. t_3 is first applied (fired) transition. It follows that if t_5 is applied first, then the application of t_5 will block the application of t_4. Symmetrically, if t_4 is applied first, then the application of t_4 will block the application of t_5. Thus, P has two stable models $\{a, b, c, e\}$ and $\{a, b, c, f\}$.

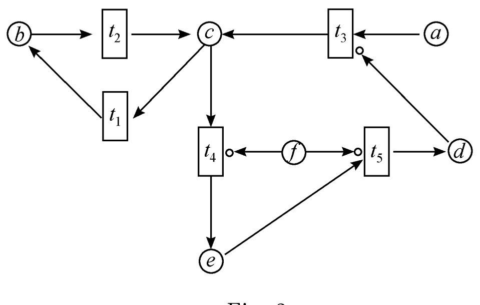

Fig. 2

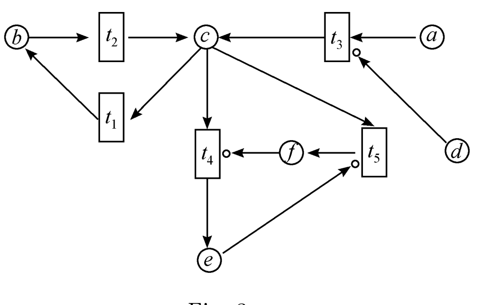

Fig. 3

2 Equivalent deciding condition of *FC*-normal program

In this section, we will present an equivalent condition deciding whether a logic program is an FC-normal program.

Definition 2.1[1]**.** Let P be a logic program. We say that a subset Con of $\mathcal{P}(H_P)$, where $\mathcal{P}(H_P)$ is the power set of H_P, is a consistency property over P if

(i) $\varnothing \in Con$,

(ii) $\forall_{A,B\subseteq H_P}[(A \subseteq B)\&(B \in Con) \Rightarrow (A \in Con)]$,

(iii) $\forall_{A\subseteq H_P}[(A \in Con) \Rightarrow (cl_{mon(P)}(A) \in Con)]$, and

(iv) whenever $\mathcal{A} \subseteq Con$ has the property that $(A, B \in \mathcal{A}) \to \exists_{C\in\mathcal{A}}[(A \subseteq C)\&(B \subseteq C)]$, then $\bigcup \mathcal{A} \in Con$.

Definition 2.2[1]**.** Let P be a logic program and Con a consistency property over P.

(i) A clause $C = a \leftarrow a_1,\dots,a_m,\neg b_1,\dots,\neg b_k \in nmon(P)$ is FC-normal with respect to (w.r.t.) Con if $V \cup \{a\} \in Con$ and not $V \cup \{a, b_i\} \in Con$ for all $1 \leqslant i \leqslant k$ whenever a subset V of H_P is such that $V \in Con$, $cl_{mon(P)}(V) = V$, $a_1,\dots,a_m \in V$, and $a, b_1,\dots,b_k \notin V$.

In other words, if for any subset V of H_P such that $V \in Con$, $cl_{mon}(V) = V$, $a_1,\dots,a_m \in V$ and $a, b_1,\dots,b_k \notin V$, we have $V \cup \{a\} \in Con$ and $V \cup \{a, b_i\} \notin Con$ for all $1 \leqslant i \leqslant k$, then the clause $C = a \leftarrow a_1,\dots,a_m,\neg b_1,\dots,\neg b_k \in nmon(P)$ is FC-normal w.r.t. Con.

(ii) P is an FC-normal program w.r.t. Con if all clauses C in P are FC-normal w.r.t. Con.

(iii) P is an FC-normal program if for some consistency property $Con \subseteq \mathcal{P}(H_P)$, P is FC-normal w.r.t. Con.

In ref. [1], it is proved that an FC-normal program has at least one stable model.

Definition 2.3. Let P be a logic program and Con a consistency property over P, and let V be in Con and $cl_{mon(P)}(V) = V$.

(i) A clause $C = a \leftarrow a_1,\dots,a_m,\neg b_1,\dots,\neg b_k \in nomn(P)$ is V-applicable if $prem(P) \subseteq V$ and $cons(C) \cap V = \varnothing$, i.e. $a_1,\dots,a_m \in V$ and $b_1,\dots,b_k \notin V$.

(ii) A clause $C = a \leftarrow a_1,\dots,a_m,\neg b_1,\dots,\neg b_k \in nomn(P)$ is V-auto-inconsistent if C is V-applicable and for some i $(1 \leqslant i \leqslant k)$ b_i is in $cl_{mon(P)}(V \cup \{a\})$. A clause $C = a \leftarrow a_1,\dots,a_m,\neg b_1,\dots,\neg b_k \in nomn(P)$ is V-auto-consistent if C is not V-auto-inconsistent. Clearly, if C is V-auto-consistent, then $cl_{mon(P)}(V \cup \{a\}) \cap \{b_1,\dots,b_k\} = \varnothing$.

(iii)[7] Two clauses $C_1 = a \leftarrow a_1,\dots,a_m,\neg b_1,\dots,\neg b_k$, $C_2 = a' \leftarrow a'_1,\dots,a'_n,\neg b'_1,\dots,\neg b'_l$ in $nmon(P)$ are V-independent. If both C_1 and C_2 are V-applicable, then the following relation holds:

$$cl_{mon(P)}(V \cup \{a\}) \cap \{b'_1,\dots,b'_l\} = \varnothing \Longleftrightarrow cl_{mon(P)}(V \cup \{a'\}) \cap \{b_1,\dots,b_k\} = \varnothing,$$

$$(\text{or } cl_{mon(P)}(V \cup \{a\}) \cap \{b'_1,\dots,b'_l\} \neq \varnothing \Longleftrightarrow cl_{mon(P)}(V \cup \{a'\}) \cap \{b_1,\dots,b_k\} \neq \varnothing.)$$

Theorem 2.1. Let P be a logic program and Con a consistency property over P. Then, the program P is FC-normal w.r.t. Con if and only if for any $V \in Con$ satisfying $cl_{mon(P)}(V) = V$ the following conditions hold:

(P_1) For any clause $C \in nmon(P)$, if C is V-applicable, then C is V-auto-consistent and $V \cup \{a\} \in Con$.

(P_2) For any clauses C_1, $C_2 \in nmon(P)$, if both C_1 and C_2 are V-applicable, then C_1, C_2 are V-independent.

Corollary 2.1. Let P be a logic program and Con a consistency property over P. For any $V \in Con$ satisfying $cl_{mon(P)}(V) = V$, if the conditions (P_1) and (P_2) in Theorem 2.1 hold, then P has stable models.

Proof of Theorem 2.1. (Only if part) Assume that P is FC-normal w.r.t. the consistency property Con. Then for all clauses $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots \neg b_k \in nmon(P)$, we have $V \cup \{a\} \in Con$ and $V \cup \{a, b_i\} \notin Con$ for all $1 \leqslant i \leqslant k$ whenever a subset V of H_P is such that $V \in Con, cl_{mon(P)}(V) = V,\ a_1, \ldots, a_m \in V$ and $a, b_1, \ldots, b_k \notin V$.

For the condition (P_1), the result is clear for $a \in V$. We assume that a clause $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots \neg b_k \in nmon(P)$ and $a_1, \ldots a_k \in V$ and $a, b_1, \ldots, b_k \notin V$. We will prove that $cl_{mon(P)}(V \cup \{a\}) \cap \{b_1, \ldots, b_k\} = \varnothing$. Otherwise, there exists some b_{i_0} ($1 \leqslant i_0 \leqslant k$) such that $b_{i_0} \in cl_{mon(P)}(V \cup \{a\})$. Hence $V \cup \{a, b_{i_0}\} \subseteq cl_{mon(P)}(V \cup \{a\})$. Since $V \cup \{a\} \in Con$, $cl_{mon(P)}(V \cup \{a\})$ is in Con too. Thus $V \cup \{a, b_{i_0}\} \in Con$, a contradiction.

For the condition (P_2), let $C_1 = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$, $C_2 = a' \leftarrow a_1', \ldots, a_n', \neg b_1', \ldots, \neg b_l'$ be in $nmon(P)$ and both V-applicable, i.e. $\{a_1, \ldots, a_m\} \subseteq V$, $b_1, \ldots, b_k \notin V$ and $\{a_1, \ldots, a_n\} \subseteq V$, $b_1', \ldots, b_l' \notin V$. Let $cl_{mon(P)}(V \cup \{a\}) \cap \{b_1', \ldots, b_l'\} = \varnothing$. We will prove that $cl_{mon(P)}(V \cup \{a'\}) \cap \{b_1, \ldots, b_k\} = \varnothing$. We discuss it by the following cases.

Case 1. $a \in V$ and $a' \in V$. It is clear.

Case 2. $a \in V$ and $a' \notin V$. In this case, we note that $V \cup \{b_i\} \notin Con$ for all $1 \leqslant i \leqslant k$, $V = cl_{mon(P)}(V) = cl_{mon(P)}(V \cup \{a\})$ and $V \cup \{a'\} \in Con$. Thus, $cl_{mon(P)}(V \cup \{a'\}) \cap \{b_1, \ldots, b_k\} = \varnothing$. Otherwise, there exists some b_{i_0} ($1 \leqslant i_0 \leqslant k$) such that $V \cup \{b_{i_0}\} \subseteq cl_{mon(P)}(V \cup \{a'\}) \in Con$. Hence $V \cup \{b_{i_0}\} \in Con$. This is a contradiction.

Case 3. $a \notin V$ and $a' \in cl_{mon(P)}(V \cup \{a\})$. In this case, $cl_{mon(P)}(V \cup \{a'\}) \subseteq cl_{mon(P)}(V \cup \{a\})$. Thus $cl_{mon(P)}(V \cup \{a'\}) \cap \{b_1, \ldots, b_k\} = \varnothing$.

Case 4. $a \notin V$ and $a' \notin cl_{mon(P)}(V \cup \{a\})$. Let $V_a = cl_{mon(P)}(V \cup \{a\})$. Then $cl_{mon(P)}(V_a) = V_a$ and $V_a \in Con$. By the assumption, we have $\{b_1', \ldots, b_l'\} \cap V_a = \varnothing$. Thus, $cl_{mon(P)}(V \cup \{a, a'\}) = cl_{mon(P)}(V_a \cup \{a'\}) \in Con$ because C_2 is a FC-normal clause w.r.t. Con. Therefore, $cl_{mon(P)}(V \cup \{a'\}) \cap \{b_1, \ldots, b_k\} = \varnothing$ since $V \cup \{a, b_i\} \notin Con$ for all $1 \leqslant i \leqslant k$. (Otherwise, there exists some b_{i_0} ($1 \leqslant i_0 \leqslant k$) such that $V \cup \{a, b_{i_0}\} \subseteq cl_{mon(P)}(V \cup \{a, a'\}) \in Con$. Hence $V \cup \{a, b_{i_0}\} \in Con$. This is a contradiction.)

Finally, we have $cl_{mon(P)}(V \cup \{a'\}) \cap \{b_1, \ldots, b_k\} = \varnothing$. Symmetrically, if $cl_{mon(P)}(V \cup \{a'\}) \cap \{b_1, \ldots, b_k\} = \varnothing$, then $cl_{mon(P)}(V \cup \{a\}) \cap \{b_1', \ldots, b_l'\} = \varnothing$.

(If part) Assume that P satisfies the conditions (P_1) and (P_2) w.r.t. Con. We show that P is FC-normal w.r.t. Con. Given any clause $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k \in nmon(P)$ and any $V \subseteq H_P$ such that $V \in Con$, $cl_{mon(P)}(V) = V, \{a_1, \ldots, a_m\} \subseteq V$ and $a, b_1, \ldots, b_k \notin V$. By the condition ($\mathrm{P}_1$), $V \cup \{a\} \in Con$. Thus, we only show $V \cup \{a, b_i\} \notin Con$ for all $1 \leqslant i \leqslant k$. If there exists some b_{i_0} ($1 \leqslant i_0 \leqslant k$) such that $V \cup \{a, b_{i_0}\} \in Con$, then there exists a set $V_0 \in Con$ satisfying $cl_{mon(P)}(V_0) = V_0$ and $V \cup \{a, b_{i_0}\} \subseteq V_0$. By Kuratowski-Zorn Lemma, there exists a minimal subset V^* of H_P such that $V^* \in Con$, $V \cup \{a, b_{i_0}\} \subseteq V^*$ and $cl_{mon(P)}(V^*) = V^*$. By the condition (P_1), we note that $cl_{mon(P)}(V \cup \{a\}) \cap \{b_1, \ldots, b_k\} = \varnothing$. Thus $cl_{mon(P)}(V \cup \{a\}) \subset V^*$ (proper inclusion). Thus, we have a clause $C' = a' \leftarrow a_1', \ldots, a_n', \neg b_1', \ldots, \neg b_l' \in nmon(P)$ and a set $V' \subseteq H_P$ such that $a'_1, \ldots, a'_n \in V'$, $a', b'_1, \ldots, b'_l \notin V'$, $cl_{mon(P)}(V \cup \{a\}) \subseteq V' \subseteq V^*$ and $cl_{mon(P)}(V' \cup \{a'\}) \subseteq V^*$. Therefore, we have $cl_{mon(P)}(V \cup \{a\}) \subset cl_{mon(P)}(V \cup \{a, a'\}) \subseteq cl_{mon(P)}(V' \cup \{a'\}) \subseteq V^*$. (Please note that $V' \in Con$ since $V' \subseteq V^*$ and $V^* \in Con$.)

We can repeat the above procedure until we get a set $V'' \in Con$ and $V'' \subseteq V^*$ and a clause $C'' = a'' \leftarrow a''_1, \ldots, a''_s, \neg b''_1, \ldots, \neg b''_t \in nmon(P)$ satisfying the following conditions:

$$a''_1, \ldots, a''_s \in V'', a'', b''_1, \ldots, b''_t \notin V'' \text{ and } cl_{mon(P)}(V'' \cup \{a''\}) \supseteq V^*.$$

Thus, $V \cup \{a, b_{i_0}\} \subseteq cl_{mon(P)}(V'' \cup \{a''\})$, i.e. $cl_{mon(P)}(V'' \cup \{a''\}) \cap \{b_1, \ldots, b_k\} \neq \varnothing$.

By the condition (P$_1$), $cl_{mon(P)}(V'' \cup \{a''\}) \cap \{b''_1, \ldots, b''_t\} = \varnothing$. Please note that $a \in V' \subseteq V''$. Thus $cl_{mon(P)}(V'' \cup \{a\}) \cap \{b''_1, \ldots, b''_t\} = \varnothing$. By the condition (P$_2$), $cl_{mon(P)}(V'' \cup \{a\}) \cap \{b''_1, \ldots, b''_t\} = \varnothing$ if and only if $cl_{mon(P)}(V'' \cup \{a''\}) \cap \{b_1, \ldots, b_k\} = \varnothing$. This is a contradiction.

Q.E.D.

Intuitively, the condition (P$_1$) in Theorem 2.1 shows that if $C \in nmon(P)$ is applicable at some stage of the construction of stable model of P, the application of C would not negate itself. And the the condition (P$_2$) in Theorem 2.1 shows that for any two clauses $C_1, C_2 \in nmon(P)$ and any well-order $\prec$, if C_1 is applied before C_2 is applied, then the application of C_2 would not negate the application of C_1. The condition of FC-normal program shows that for any clause $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k \in nmon(P)$, the application of C in some consistency property Con would not cause any inconsistentency.

Example 2.1. Let $P = \{a \leftarrow, \ b \leftarrow c, \ c \leftarrow a, \neg b\}$. Then P is not FC-normal because the clause $C = c \leftarrow a, \neg b$ is V-auto-inconsistent, where $V = \{a\}$.

Example 2.2. Let $P = \{a \leftarrow, \ b \leftarrow c, \ c \leftarrow b, \ c \leftarrow a, \neg d, \ e \leftarrow c, \neg f, \ f \leftarrow c, \neg e\}$. Then P is FC-normal w.r.t. the consistency property $Con = \mathcal{P}(\{a, b, c, e\}) \cup \mathcal{P}(\{a, b, c, f\})$. Clearly, P satisfies the conditions (P$_1$) and (P$_2$) w.r.t. Con in Theorem 2.1.

On the other hand, we can construct a family of subsets of H_P, denoted by $\mathcal{C}(P)$, over a logic program P as follows: that is the least subset of $\mathcal{P}(H_P)$ satisfying the following conditions:

(i) $\varnothing \in \mathcal{C}(P)$,

(ii) for any $A \subseteq H_P$, $cl_{mon(P)}(A) \in \mathcal{C}(P)$,

(iii) for any $A \subseteq H_P$, if $A \in \mathcal{C}(P)$, then any subset B of A belongs to $\mathcal{C}(P)$,

(iv) for any $A \subseteq H_P$, if $A \in \mathcal{C}(P)$, then $cl_{mon(P)}(A \cup \{a\}) \in \mathcal{C}(P)$ whenever any clause $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$ in P is such that $\{a_1, \ldots, a_m\} \subseteq A$ and $a, b_1, \ldots, b_k \notin A$.

(v) $\mathcal{C}(P)$ contains no others.

It is easy to check that the above $\mathcal{C}(P)$ is a consistency property over P. We say that $\mathcal{C}(P)$ is normal consistency property of P, denoted by $Con_{norm(P)}$. Please note that P need not be FC-normal here. Clearly, for any consistency property Con such that P is FC-normal w.r.t. Con, we have $Con_{norm(P)} \subseteq Con$. In other words, if P is FC-normal w.r.t. $Con_{norm(P)}$, then $Con_{norm(P)}$ is the least consistency property, where P is FC-normal w.r.t., under the inclusion of sets. In Example 2.2, P is FC-normal w.r.t. $Con_{norm(P)} = \mathcal{P}(\{a, b, c, e\}) \cup \mathcal{P}(\{a, b, c, f\})$, and P is also FC-normal w.r.t. $Con' = \mathcal{P}(\{a, b, c, e\}) \cup \mathcal{P}(\{a, b, c, f\}) \cup \mathcal{P}(\{a, b, d, e\}) \cup \mathcal{P}(\{a, b, d, f\})$.

In fact, we can introduce a well-order $\prec$ of $nmon(P)$, that is the well-order $\prec$ determining some listing of clauses in $nmon(P)$, $\{\gamma_\alpha : \alpha \in \gamma\}$, where γ is some ordinal. Let Θ_γ be the least cardinal such that $\gamma \leqslant \Theta_\gamma$. By the normal forward chaining construction in ref. [1], we can construct a set $M^{\prec} = \bigcup_{\alpha \in \Theta_\gamma} M_\alpha^{\prec}$.

The Normal Forward Chaining Construction of $M^{\prec}$

Step 0. Let $M_0^{\prec} = cl_{mon(P)}(\varnothing)$.

Step 1. $\alpha = \eta + 1$ is a successor ordinal. Given $M_\eta^{\prec}$, let $l(\alpha)$ be the least $\lambda \in \gamma$ such that

$$\gamma_\lambda = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$$

where $a_1, \ldots, a_m \in M_\eta^{\prec}$ and $b_1, \ldots, b_k, a \notin M_\eta^{\prec}$. If there is no such $l(\alpha)$, then let $M_{\eta+1}^{\prec} = M_\alpha^{\prec} = M_\eta^{\prec}$. Otherwise, let

$$M_{\eta+1}^{\prec} = M_\alpha^{\prec} = cl_{mon(P)}(M_\eta^{\prec} \cup \{c(\gamma_{l(\alpha)})\}).$$

Step 2. α is a limit ordinal. Then $M_\alpha^{\prec} = \bigcup_{\beta \in \alpha} M_\beta^{\prec}$.

Finally, $M^{\prec} = \bigcup_{\alpha \in \Theta_\gamma} M_\alpha^{\prec}$.

Clearly, $Con_{norm(P)} = \bigcup_{\prec \in WO(P)} \mathcal{P}(M^{\prec})$, where $WO(P)$ is the set of all well-orders over $nmon(P)$.

3 Extended stratification over logic program

In this section, we introduce a new stratification, extended stratification, over logic program by defining a new rank function. We restrict our attention to propositional logic program. Following refs. [3, 4], a logic program P called (locally) stratified if there exists an ordinal ν and a function $rank : H_P \rightarrow \nu$ such that for every clause $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$, $rank(a_i) \leqslant rank(a)$ for all $1 \leqslant i \leqslant m$ and $rank(b_j) < rank(a)$, for all $1 \leqslant j \leqslant k$. Using a generally well-known argument (see ref. [5]) one can show that a stratified program has the unique stable model.

Please note that in the definition of stratification function $rank$ and in nonmonotonic derivation, the equal symbol $=$ is applied to dealing with monotonic derivation cycle and the exact nonequal symbol $<$ is applied to preventing nonmonotonic derivation cycle. It means intuitively that the earlier applied nonmonotonic clauses would not be negated by the later applied (monotonic and nonmonotonic) clauses.

We, however, note that FC-normal program has the characterization:

the earlier applied nonmonotonic clause C_1 would not be negated by the later applied nonmonotonic clauses C_2 because the application of C_1 could block the application of C_2.

In Petri-net representation $N(P)$ of an FC-normal program P, both monotonic cycles and nonmonotonic cycle are allowed to occur. For the convenience of description, we view a transition t_C and the clause C corresponding to t_C to be the same. It is interesting that in the derivation of FC-normal program, for any nonmonotonic cycle $\mathcal{C}$ in $N(P)$, if a nonmonotonic clause C in $\mathcal{C}$ is applied, then all other nonmonotonic clauses in $\mathcal{C}$, which may be applied, are immediately blocked by the application of C.

Our motivation is how to deal with the two kinds of cycles in $N(P)$.

Definition 3.1. Let P be a (propositional) logic program. P is called extended (locally) stratified if there exist an ordinal ν and a function $rank^* : H_P \rightarrow \nu \times \nu$ such that for every clause $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$ in P, the following conditions hold:

(i) for all i, $1 \leqslant i \leqslant m$, $\max(rank^*(a_i)) \leqslant \min(rank^*(a))$ and

(ii) for all j, $1 \leqslant j \leqslant k$, $\min(rank^*(b_j)) < \max(rank^*(a))$,

where $rank^*(a) = (\nu_l, \nu_r)$, $\max(rank^*(a)) = \max(\nu_l, \nu_r)$ and $\min(rank^*(a)) = \min(\nu_l, \nu_r)$. Obviously, we have

Corollary 3.1. Let P be a (propositional) logic program. If P is (locally) stratified, then P is extended (locally) stratified.

However, an extended (locally) stratified program may not be (locally) stratified.

Example 3.1. The logic program P consists of clauses $\{a \leftarrow,\ b \leftarrow c,\ c \leftarrow b,\ c \leftarrow a, \neg d,\ e \leftarrow c, \neg f, \neg g,\ f \leftarrow c, \neg e,\ g \leftarrow f, \neg e,\ h \leftarrow g,\ g \leftarrow h\}$. Define a rank function $rank^*$ over P as follows: $rank^*(a) = (0,0)$, $rank^*(b) = (1,1)$, $rank^*(c) = (1,1)$, $rank^*(d) = (0,0), rank^*(e) = (2,4)$, $rank^*(f) = (2,3)$, $rank^*(g) = (3,3)$, $rank^*(h) = (3,3)$ (fig. 4).

Clearly, P is extended (locally) stratified, and P is not (locally) stratified since P has two stable models, $\{a, b, c, e\}$ and $\{a, b, c, f, g, h\}$.

Example 3.2. The logic program P consists of clauses $\{a \leftarrow,\ c \leftarrow b,\ b \leftarrow a, \neg c\}$.

Clearly, P is neither (locally) stratified nor FC-normal.

Moreover, we have an example that an FC-normal program may not be extended (locally) stratified.

Example 3.3. The logic program P consists of clauses $\{a \leftarrow,\ c \leftarrow a, \neg b,\ b \leftarrow a, \neg c,\ b \leftarrow d,\ d \leftarrow b,\ c \leftarrow e,\ e \leftarrow c\}$ (fig. 5).

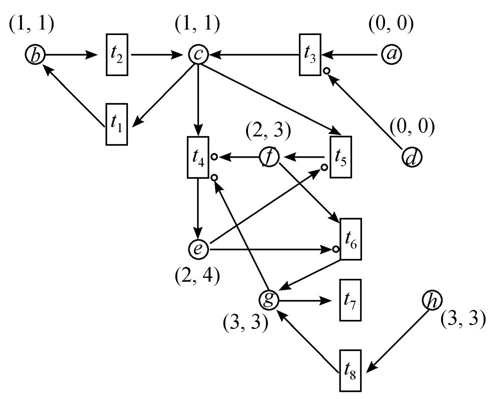

Fig. 4

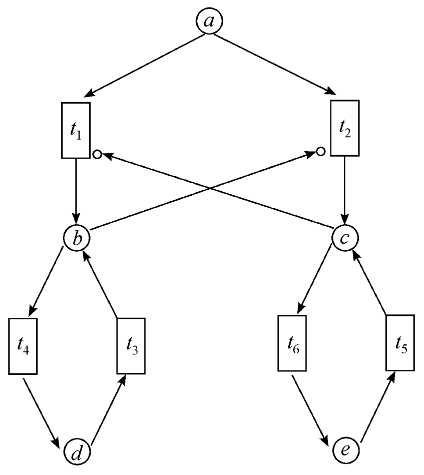

Fig. 5

Clearly, P is FC-normal. And it is not extended (locally) stratified, if otherwise, there exists a rank function $rank^*$ satisfying the conditions in Definition 3.1. Thus, we have the following contradictory relations:

$$\max(rank^*(b)) = \min(rank^*(b)),\ \max(rank^*(c) = \min(rank^*(c)),$$

$$\min(rank^*(b)) < \max(rank^*(c)) \text{ and } \min(rank^*(c) < max(rank^*(b)).$$

We, however, will prove that an extended (locally) stratified program is FC-normal.

The following lemmas are helpful for the proof later.

Lemma 3.1. Let P be a logic program and I a subset of H_P. If $c \in cl_{mon(P)}(I)$ and $c \notin I \cup \{a'|\ a' \leftarrow \in P\}$, then

(i) there exists a proof sequence $< c_1, c_2, \ldots, c_l >$ of c from I in P, and

(ii) there exists a subsequence $< c_{i_1}, c_{i_2}, \ldots, c_{i_p}, c_l >$ of $< c_1, c_2, \ldots, c_l >$ and a sequence $C'_1, \ldots, C'_p$ of clauses in $mon(P)$ such that

(a) $c_{i_1} \in I \cup \{a'|\ a' \leftarrow \in P\}$,

(b) for all $1 < j \leqslant p$, $c_{i_j} = c(C'_{j-1})$,

(c) $c_l = c(C_{i_p})$.

Proof. (i) By the compactness theorem in propositional calculus.

(ii) By the definition of proof sequence. Q. E. D.

Lemma 3.2. Let P be a logic program and V a subset of H_P satisfying $cl_{mon(P)}(V) = V$. If neither a nor b is in $V \cup \{a'|\ a' \leftarrow \in P\}$, and $b \in cl_{mon(P)}(V \cup \{a\})$, then a occurs in the sequence $< c_1, c_2, \ldots, c_l >$, where $< c_1, c_2, \ldots, c_l, b >$ is a proof sequence of b from $V \cup \{a\}$ over P.

Furthermore, we have a subsequence $< a, c_{i_1}, c_{i_2}, \ldots, c_{i_p}, b >$ of $< c_1, c_2, \ldots, c_l, b >$ and a sequence $C'_1, \ldots, C'_p, C^*$ of clauses in $mon(P)$ such that for all j $(1 \leqslant j \leqslant p)$, $c_{i_j} = c(C'_j)$ and $b = c(C^*)$, and $a \in prem(C'_1)$.

Proof. By Lemma 3.1 and the definition of proof sequence. Q.E.D.

Theorem 3.1. Let P be a logic program. If P is extended (locally) stratified, then P is FC-normal.

Proof. Assume that the logic program P is extended (locally) stratified w.r.t. the rank function $rank^*$: $H_P \to \nu \times \nu$, where ν is an ordinal.

We now prove that P is FC-normal w.r.t. the consistency property $Con_{norm(P)}$ defined in section 2.

For any $V \in Con_{norm(P)}$ and clause $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k \in P$, we assume that $cl_{mon(P)}(V) = V$, $\{a_1, \ldots, a_m\} \subseteq V$ and $V \cap \{a, b_1, \ldots, b_k\} = \varnothing$. We will show that

(i) $V \cup \{a\} \in Con_{norm(P)}$ and

(ii) for all $1 \leqslant i \leqslant k$, $V \cup \{a, b_i\} \notin Con_{norm(P)}$.

By the assumptions and definition of $Con_{norm(P)}$, we have $V \cup \{a\} \in Con_{norm(P)}$. Thus, we only prove that $V \cup \{a, b_i\} \notin Con_{norm(P)}$ for all i $(1 \leqslant i \leqslant k)$. By the construction of $Con_{norm(P)}$, we have to prove the following conclusion:

for any $V' \in Con_{norm(P)}$, if $V \cup \{a\} \subseteq V'$ and $cl_{mon(P)}(V') = V'$, then $b_1, \ldots, b_k \notin V'$.

We prove the above conclusion by the following cases.

Case 1. $V' = cl_{mon(P)}(V \cup \{a\})$.

(Please note that $cl_{mon(P)}(V) = V$, $\{a_1, \ldots, a_m\} \subseteq V$ and $V \cap \{a, b_1, \ldots, b_k\} = \varnothing$.)

Otherwise, there exists some $b_{j_0} \in cl_{mon(P)}(V \cup \{a\})$. By Lemma 3.2, we have a sequence $< a, c_{i_1}, c_{i_2}, \ldots, c_{i_p}, b_{j_0} >$ over H_P and a sequence $C'_1, \ldots, C'_p, C^*$ of clauses in $mon(P)$ such that for all j $(1 \leqslant j \leqslant p)$ $c_{i_j} = c(C'_j)$ and $b_{j_0} = c(C^*)$. By the definition of $rank^*$, we have: $\max(rank^*(a)) \leqslant \min(rank^*(c_{i_1})) \leqslant \max(rank^*(c_{i_1})) \leqslant \ldots \leqslant \min(rank^*(c_{i_p})) \leqslant \max(rank^*(c_{i_p})) \leqslant \min(rank^*(b_{j_0}))$.

It contradicts $\min(rank^*(b_{j_0})) < \max(rank^*(a))$.

Case 2. $cl_{mon(P)}(V \cup \{a\}) \subset V' \in Con_{norm(P)}$.

Please note that the nonmonotonic clause $C = a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_k$ in P is applied to V to get $V \cup \{a\}$, and only some monotonic clauses in P are applied in the procedure of forming $cl_{mon(P)}(V \cup \{a\})$ from $V \cup \{a\}$. Thus, at least one nonmonotonic clause in P has been applied from $cl_{mon(P)}(V \cup \{a\})$ to V'. We denote the set of these nonmonotonic clauses in P by $P' = \{C'_1, \ldots, C'_\zeta, \ldots\}$, and assume that the order of applied clauses is $C'_1 \prec \ldots \prec C'_\zeta \prec \ldots$ in some well-order $\prec$ over $nmon(P)$.

From the set P' of clauses and $C'_1 \prec \ldots \prec C'_\zeta \prec \ldots$, we set $V_0 = cl_{mon(P)}(V \cup \{a\})$, and if $\gamma = \beta+1$, then set $V_\gamma = cl_{mon(P)}(V_\beta \cup \{c(C'_\gamma)\})$, and if γ is a limit ordinal, then set $V_\gamma = \bigcup_{\zeta<\gamma} V_\zeta$. Finally, we have $V' = \bigcup_{\zeta<|P'|^+} V_\zeta$.

Additionally, $V_0 \subseteq V_1 \ldots, V_\zeta \subseteq \ldots \subseteq V'$, $V_0 \subset V'$ and $V' \in Con_{norm(P)}$.

From Case 1, we know that $V_0 \cap \{b_1, \ldots, b_k\} = V_0 \cap cons(C) = \varnothing$. By transfinite induction on γ, we can prove that $V_\gamma \cap cons(C_\gamma) = \varnothing$ for any C'_γ in P'.

Suppose that there exists some $b_{j_0} \in V' = cl_{mon(P)}(V')$. Consider the Horn projection of P', P'_{Horn}. We know that b_{j_0} is in $cl_{mon(P^*)}(V \cup \{a\})$, where $P^* = P \cup P'_{Horn}$. By a similar reason to that in Case 1, we have a contradiction with $\min(rank^*(b_{j_0})) < \max(rank^*(a))$.

Q. E. D.

Corollary 3.2. An extended (locally) stratified program has at least one stable model.

4 Finite FC-normal logic program

In this section, we focus our attention on finite logic program. By the Petri-net representation of a logic program P, please note that in the construction of stable models the number of all applicable well-ordering over $nmon(P)$ is much less than $2^{|nmon(P)|}$ in many cases. In an application of nonmonotonic clauses, the orders among some nonmonotonic clauses are fixed. For example, in fig. 4, the order $\{t_3\}$, $\{t_4, t_5\}$, $\{t_6\}$ is fixed. That is, in the application, t_3 must be applied first, and if t_6 is applied, then it must be the last one. In fact, we only use two well-orders in Example 3.1.

In ref. [1], Marek et al. presented two versions of normal forward chaining constructions for general and countable logic program, respectively. If we restrict ourselves to finite logic program P, then define an index function $Ind : nmon(P) \to N$, where N denotes the set of natural numbers, to replace a well-order $\prec$ over $nmon(P)$. For some fixed Ind, we can introduce the following algorithm to compute M^{Ind}.

Algorithm 4.1. Finite normal forward chaining construction.

```
Input: A finite logic program $P$ and an index function $Ind$ of $nmon(P)$.
Output: $M^{Ind}$.
procedure $FFC(P, Ind)$;
begin
```

```
    $M^{Ind} := cl_{mon(P)}(\varnothing)$;
    $mark := 1$;
    **while** $(mark == 1)$ **do**
        $AC := \{C \in nmon(P) |\ prem(C) \subseteq M^{Ind}$, and $(\{c(C)\} \cup cons(C)) \cap M^{Ind} = \varnothing\}$;
        **if** $(AC == \varnothing)$ **then** $mark := 0$
            **else**
                $\{_1$ $\min\_ind := \min\{Ind(C) |\ C \in AC\}$;
                    $M^{Ind} := cl_{mon(P)}(M^{Ind} \cup \{c(C_{\min\_ind})\})$; $\}_1$  $*Ind(C_{\min\_ind}) = \min\_ind*$
    **end(while)**
    **return** $M^{Ind}$;
**end**;
```

M^{Ind} is called the base set w.r.t. the index function Ind here.

Clearly, the complexity of Algorithm 4.1 is $O(|nmon(P)|^2)$, where a computation of $cl_{mon(P)}(.)$ is viewed as a unit time.

Modifying Algorithm 4.1 and using the code function $< x, y >= \frac{1}{2}(x^2 + 2xy + y^2 + 3x + y)$ and defining $< x, y, z >=< x, < y, z >>$, we can compute the normal consistency property $Con_{norm(P)}$ of a finite logic program P by the following algorithm.

Algorithm 4.2. Base set of normal consistency property

Input: A finite logic program P.

Output: the base set of the normal consistency property over P, $Base \subseteq \mathcal{P}(H_P)$.

```
**procedure** $Base\_set(P)$;
**begin**
    $nmon_0(P) = nmon(P)$;
    $I_0 := \{< 0, 0 >\}$;
    $M_0^{<0,0>} := cl_{mon(P)}(\varnothing)$;
    $n := 0$;
    $Base := \varnothing$;
    **while** $(n < |nmon(P)|)$ **do**
        $I_{n+1} := \varnothing$;
        **for** every $< i, j >\in I_n$ **do**
        $\{_1$ $A_n^{<i,j>} := \{C \in nmon_n(P) |\ prem(C) \subseteq M_n^{<i,j>}$ and
                        $(\{c(C)\} \cup cons(C)) \cap M_n^{<i,j>} = \varnothing\}$;
            $k_{<n,i,j>} := |A_n^{<i,j>}|$;
            **if** $(k_{<n,i,j>} == 0)$ **then**
                $\{_2$ $Base := Base \cup \{M_n^{<i,j>}\}$; $I_n := I_n \setminus \{< i, j >\}$; $\}_2$
                **else**
                    $\{_3$ listing $A_n^{<i,j>}$ : $\{C_{<<n,i,j>,0>}, \ldots, C_{<<n,i,j>,k_{<n,i,j>}-1>}\}$;
                        $I_{<n,i,j>} := \{<< n, i, j >, 0 >, \ldots, << n, i, j >, k_{<n,i,j>} - 1 >\}$;
                        $I_{n+1} := I_{n+1} \cup I_{<n,i,j>}$; $\}_3$
        $\}_1$
        $nmon_{n+1}(P) := nmon_n(P)$;
        for every $<< n,i,j >, k >\in I_{n+1}$ do
        $\{_4\ M_{n+1}^{<<n,i,j>,k>} := cl_{mon(P)}(M_n^{<i,j>} \cup \{c(C_{<<n,i,j>,k>})\})$;
            $nmon_{n+1}(P) := nmon_{n+1}(P) \setminus A_n^{<i,j>}$; $\}_4$
        $n := n + 1$;
    end(while);
    return $Base$;
end.
```

For the logic program P in example 3.1, algorithm 4.2 returns to $Base = \{\{a,b,c,e\}, \{a,b,c,f,g,h\}\}$.

The worst time complexity of Algorithm 4.2 is $O(2^{|nmon(P)|}|nmon(P)|^2)$. For example, $P = \{a \leftarrow,\ c_1 \leftarrow a, \neg b_1,\ \ldots,\ c_n \leftarrow a, \neg b_n\}$ although $Base = \{\{a, c_1, \ldots, c_n\}\}$ and $|Base| = 1$.

By Algorithms 4.1 and 4.2, it follows that

Theorem 4.1. Let P be a finite logic program. Then, the following holds:

(i) For any index function $Ind:\ nmon(P) \to N$, $M^{Ind} \in Base$.

(ii) For any $M \in Base$, there exists an index function $Ind:\ nmon(P) \to N$ such that $M = M^{Ind}$,

where M^{Ind} is the output set of Algorithm 4.1 and $Base$ is the output set of Algorithm 4.2.

Theorem 4.2. Let P be a finite logic program and $Base$ the output set of Algorithm 4.2. Then $Con_{norm(P)} = \bigcup_{M\in Base} \mathcal{P}(M)$ is a consistency property, normal consistency property.

Theorem 4.3. Let P be a finite logic and Con a consistency property over P. If P is FC-normal with respect to Con, then $Con_{norm(P)} \subseteq Con$ and P is FC-normal with respect to $Con_{norm}(P)$.

Theorem 4.4. Let P be a finite logic program and $Base$ the output set of Algorithm 4.2. For $M \in Base$, M is a stable model of P if and only if $(\forall C \in nmon(P))(prem(P) \subseteq M \Rightarrow cons(C) \cap M = \varnothing)$.

Theorem 4.5. Let P be a finite logic and $Base$ the output set of Algorithm 4.2. The following are equivalent.

(i) P is FC-normal.

(ii) P is FC-normal with respect to $Con_{norm(P)}$.

(iii) For any $M \in Base$ and any $C \in nmon(P)$, if $prem(C) \subseteq M$, then $cons(C) \cap M = \varnothing$.

(iv) For any $M \in Base$, M is a stable model of P.

Theorem 4.5 shows that for the deciding of a finite FC-normal program P, we only check whether P is FC-normal with respect to $Con_{norm(P)}$.

5 Conclusions

In this paper, we have presented an equivalent condition of deciding whether a logic program is FC-normal. The deciding condition describes clearly the characterization of an FC-normal program. By the Petri-net representation of a logic program, we have introduced extended (locally) stratification over a logic program, and proved that an extended (locally) stratified program is FC-normal, thus an extended (locally) stratified program has at least one stable model. Moreover, we have given examples that an FC-normal program may not be extended (locally) stratified and an extended (locally) stratified program may not be (locally) stratified. Finally, we have presented algorithms about computation of consistency property and some deciding methods of FC-normal program for finite logic program. The Petri-net representation of a logic program is useful for investigating the stable models of the logic program and its complexity.

References

1. Marek, V. M., Nerode, A., Remmel, J. B., Logic programs, well-ordering, and forward chaining, Annals of Pure and Applied Logic, 1999, 96: 231—276.
2. Marek, V. M., Nerode, A., Remmel, J. B., A context for belief revision: forward chaining-normal nonmonotonic rules systems, Annals of Pure and Applied Logic, 1994, 67: 269—323.
3. Apt, K., Blair, H. A., Walker, A., Towards a theory of declaritive knowledge, in Foundation of Deductive Databases and Logic Programming (ed. Minker, J.), Los Altos, CA: Morgan Kaufmann, 1987, 89—142.
4. Przymusinski, T., On the declarative semanitics of stratified deductive databases and logic programs, in Foundation of Deductive Databases and Logic Programming (ed. Minker, J.), Los Altos, CA: Morgan Kaufmann, 1987, 99: 193—216.
5. Marek, V. M., Truszczyński, M., Nonmonotonic Logic, Berlin: Springer-Verlag, 1993.
6. Lin, C., Murata, T., A Petri net model for nonmonotonic reasoning based on annotated logic programs, IEICE Trans. Fundamentals, 1994, E77-A (10): 1579—1587.
7. Zhao Xi-shun, Disjunction-free default logic and its complexity, Ph. D. Thesis., Nanjing University in China, 1999.