Unfold/fold transformation of stratified programs

Hirohisa Seki*

Central Research Laboratory, Mitsubishi Electric Corporation, 8-1-1 Tsukaguchi-Honmachi, Amagasaki, Hyogo 661, Japan

Abstract


This paper describes some extensions of Tamaki-Sato's (1984) unfold/fold transformation of definite programs. We first propose unfold/fold rules also preserving the finite failure set (by SLD-resolution) of a definite program, which the original rules proposed by Tamaki and Sato do not. Then, we show that our unfold/fold rules can be extended to rules for stratified programs and prove that both the success set and the finite failure set (by SLDNF-resolution) of a stratified program are preserved. Preservation of equivalence of the perfect model semantics (Przymusinski (1988)) is also discussed.

1. Introduction

Program transformation provides a powerful methodology for program development, especially for derivation of an efficient program preserving the same meaning as that of an original and possibly inefficient program. Thus, one of the most important properties of program transformation is preservation of equivalence (Maher [10] investigated various formulations of equivalence for logic programs).

Tamaki and Sato proposed an elegant framework for unfold/fold transformation of logic programs [16]. Their transformation rules preserve the equivalence of a definite program in the sense of the least Herbrand model. Kawamura and Kanamori [7] recently proved that Tamaki-Sato's transformation also preserves the success set of a program, that is, a transformed program has the same computed answer substitution as that of the original program for any goal. Thus, the transformation rules by Tamaki and Sato seem to be sufficient, at least as far as positive information inferred from a program is concerned.

In general, however, their transformation does not always preserve the finite failure set (by SLD-resolution) of a definite program. The evaluation of a goal in

* This work was done while the author was at ICOT (Institute for New Generation Computer Technology).
a transformed program might not be terminating, even if the evaluation of that goal is finitely failed in the original program. Thus, when we are interested in negative information inferred from a program and Clark's negation as failure rule [3] is used, their transformation is not sufficient. Furthermore, when we consider an extension of their rules to a general logic program where the body of a clause may contain negative literals, the failure to preserve the finite failure of a program would lead to failure to preserve positive information inferred from the program.

In this paper, we propose unfold/fold rules which also preserve the finite failure set of a definite program. Then, we extend them to a stratified program and show that our transformation preserves both the success set and the finite failure set (by SLDNF-resolution) of a given stratified program. Preservation of equivalence of transformation in the perfect model semantics [12] is also discussed.

The organization of this paper is as follows. After summarizing preliminaries, Section 2 gives transformation rules which preserve the finite failure set of a definite program. Section 3 extends them to stratified programs. Section 4 discusses transformation rules which preserve the perfect model semantics. Finally, a summary of this work and a discussion of related work are given in Section 5.

Throughout this paper, we assume that the reader is familiar with the basic concept of logic programming, and the terminology follows that in [9]. As notation, variables are denoted by $X, Y, \ldots$, and atoms by $A, B, \ldots$. Multisets of atoms are denoted by $L, K, M, \ldots$, and $\theta, \sigma, \ldots$ are used for substitutions.

2. Unfold/fold transformation

2.1. Preliminaries: Rules of transformation

This section describes Tamaki-Sato's unfold/fold transformation for definite programs [16]. The following descriptions of transformation rules are borrowed mainly from [15] and [7].

**Definition 2.1.1 (initial (definite) program).** An initial (definite) program $P_0$ is a definite program satisfying the following conditions:

(1) $P_0$ is divided into two disjoint sets of clauses, $P_{new}$ and $P_{old}$. The predicates defined in $P_{new}$ are called *new predicates*, while those defined in $P_{old}$ are called *old predicates*.

(2) The new predicates appear neither in $P_{old}$ nor in the bodies of the clauses in $P_{new}$.

**Example 2.1.1.** Let $P_0 = \{C_1, C_2, C_3\} \cup DB$, where

\[
\begin{align*}
C_1: & \quad \text{reach}(X, Y) \leftarrow \text{arc}(X, Y); \\
C_2: & \quad \text{reach}(X, Y) \leftarrow \text{arc}(X, Z), \text{reach}(Z, Y); \\
C_3: & \quad \text{br}(X, Y, N) \leftarrow \text{reach}(X, N), \text{reach}(Y, N);
\end{align*}
\]
and DB is a set of the following unit clauses defining predicate arc:

\[
\begin{align*}
\text{arc}(a, c) &. \quad \text{arc}(c, a). \quad \text{arc}(c, e). \\
\text{arc}(e, b) &. \quad \text{arc}(e, d). \quad \text{arc}(b, d).
\end{align*}
\]

Predicate reach(X, Y) is supposed to hold if there exists a path starting from node X and ending at node Y in a directed graph (shown above) whose relationship of arcs is given in DB. Predicate br(X, Y, N) is supposed to hold if node N is reachable from both node X and node Y.

Let Fold = \{C, C, C\} ∪ DB, I* = \{C, \}. Thus, “br” is a new predicate, while the other predicates are old predicates.

We call an atom A a new atom (an old atom) when the predicate of A is a new predicate (an old predicate), respectively.

**Definition 2.1.2 (unfolding).** Let P_i be a program and C a clause in P_i of the form:

\[ H \leftarrow A, L. \]

Suppose that C_1, ..., C_k are all the clauses in P_i such that C_j is of the form:

\[ A_j \leftarrow K_j \]

and A_j is unifiable with A, by an mgu, say \( \theta_j \), for each j (1 ≤ j ≤ k).

Let C_j (1 ≤ j ≤ k) be the result of applying \( \theta_j \) after replacing A in C with the body of C_j, namely, C_j = H\( \theta_j \) \leftarrow K_j\( \theta_j \), I\( \theta_j \). Then, \( P_{i+1} = (P_i - \{C\}) \cup \{C_1, ..., C_k\} \). C is called the unfolded clause and C_1, ..., C_k are called the unfolding clauses.

**Example 2.1.2 (continued from Example 2.1.1).** By unfolding C_3 at atom ‘reach(X, N)’ in its body, program P_i = \{C_1, C_2, C_4, C_5\} ∪ DB is obtained, where

\[
\begin{align*}
C_4: & \quad \text{br}(X, Y, N) \leftarrow \text{arc}(X, N), \text{reach}(Y, N). \\
C_5: & \quad \text{br}(X, Y, N) \leftarrow \text{arc}(X, X_i), \text{reach}(X_i, N), \text{reach}(Y, N).
\end{align*}
\]

**Definition 2.1.3 (folding).** Let C be a clause in P_i of the form:

\[ A \leftarrow K, L \]

and D a clause in P_{new}[^1] of the form:

\[ B \leftarrow K'. \]

Suppose that there exists a substitution \( \theta \) satisfying the following conditions:

(F1) \( K'\theta = K. \)

(F2) Let \( X_1, ..., X_j, ..., X_m \) be internal variables of D, namely, appearing only in the body \( K' \) of D but not in B. Then, each \( X_i\theta \) is a variable in C such that it appears in none of A, L and B\( \theta \). Furthermore, \( X_i\theta \neq X_j\theta \) if \( i \neq j \).

(F3) D is the only clause in P_{new} whose head is unifiable with B\( \theta \).

(F4) Either the predicate of A is an old predicate, or C is the result of applying unfolding at least once to a clause in P_0.

[^1] Note that D is not necessarily in P_i.
Then, let $C'$ be a clause of the form: $A \leftarrow B\emptyset, L$ and let $P_{i+1}$ be $(P_i - \{C\}) \cup \{C'\}$. $C$ is called the folded clause and $D$ is called the folding clause.

**Example 2.1.3 (continued from Example 2.1.2)** By folding the body of $C_5$ by $C_3$, program $P_2 = \{C_1, C_2, C_4, C_6\} \cup DB$ is obtained, where

$$C_6: \quad \text{br}(X, Y, N) \leftarrow \text{arc}(X, X_1), \text{br}(X_1, Y, N).$$

**2.1.1. Previous results**

**Definition 2.1.4 (transformation sequence).** Let $P_0$ be an initial program and $P_{i+1}$ $(i \geq 0)$ a program obtained from $P_i$ by applying either unfolding or folding. Then, the sequence of programs $P_0, P_1, \ldots, P_N$ is called a transformation sequence starting from $P_0$.

For the above unfold/fold transformation, Tamaki and Sato proved the following result [16].

**Theorem 2.1.1 (Tamaki and Sato [16]).** The least Herbrand model $M_{P_i}$ of any program $P_i$ in a transformation sequence starting from initial program $P_0$, is identical to that of $P_0$.

Recently, Kawamura and Kanamori [7] showed that Tanakai–Sato’s transformation also preserves answer substitutions for any goal.

**Definition 2.1.5 (success set).** Let $P$ be a (definite) program. The set of all the atom-substitution pairs $(A, \sigma)$, such that there exists a successful SLD-derivation of $P \cup \{\leftarrow A\}$ with computed answer $\sigma$, is called the success set of $P$, and is denoted by $SS(P)$.

**Theorem 2.1.2 (Kawamura and Kanamori [7]).** The success set $SS(P_i)$ of any program $P_i$ in a transformation sequence starting from initial program $P_0$, is identical to that of $P_0$.

**Example 2.1.4 (continued from Examples 2.1.1 and 2.1.3).** Since $\text{br}(a, c, e) \in M(P_0)$ holds, $\text{br}(a, c, e)$ is also in $M(P_2)$ from Theorem 2.1.1. More precisely, $(\text{br}(X, Y, N), \sigma = \{X/a, Y/c, N/e\})$ is in $SS(P_0)$, thus, that pair is also in $SS(P_2)$, from Theorem 2.1.2.

**2.2. Modified folding rule and preservation of FF**

**2.2.1. Modified folding rule**

This paper also considers the finite failure set (by SLD-resolution) of a program.
Definition 2.2.1 (finite failure (FF) set). Let \( P \) be a (definite) program. The set of all atoms \( A \) such that there exists a finitely failed SLD-tree for \( P \cup \{ \leftarrow A \} \), is called the (SLD) finite failure set of \( P \), and is denoted by \( \text{FF}(P) \).

The partial correctness of the transformation w.r.t. FF is easily shown.

Proposition 2.2.1 (partial correctness w.r.t. FF). Let \( P_0, \ldots, P_N \) be a transformation sequence. Then, \( \text{FF}(P_N) \subseteq \text{FF}(P_0) \) for all \( N \geq 0 \).

Proof. Let \( G \) be a definite goal, and suppose that \( P_N \cup G \) has a finitely failed SLD tree. From the soundness of SLD-resolution [3], \( \text{comp}(P_N) \vdash G \). It is easy to see that \( \text{comp}(P_0) \vdash \text{comp}(P_N) \) holds. Thus, \( G \) is also a logical consequence of \( \text{comp}(P_0) \). Then, from the completeness of SLD-resolution [5], \( P_0 \cup G \) has a finitely failed SLD-tree.

Tamaki-Sato's unfold/fold transformation, however, does not preserve the total correctness w.r.t. FF. That is, \( \text{FF}(P_0) \subseteq \text{FF}(P_i) \) for all \( i \) (\( N \geq i > 0 \)) does not hold in general.

Example 2.2.1 (continued from Example 2.1.1, 2.1.3). The failure set of the original program \( P_0 \) is not preserved. For example, \( \text{br}(a, b, e) \in \text{FF}(P_0) \), while \( \text{br}(a, b, e) \) is not contained in \( \text{FF}(P_2) \). In fact, any SLD-tree for \( P_2 \cup \{ \leftarrow \text{br}(a, b, e) \} \) has an infinite branch. Thus, \( \text{FF}(P_0) \not\subseteq \text{FF}(P_2) \).

We now give a modified transformation rule which also preserves the total correctness w.r.t. FF. In order to specify such a rule, we need several definitions.

Definition 2.2.2 (inherited atom). Let \( P_0, \ldots, P_N \) be a transformation sequence starting from \( P_0 \), and \( C \) a clause in \( P_i \) (\( N \geq i \geq 0 \)) whose head is a new atom. Then, an atom in the body of \( C \) is called an atom inherited from \( P_0 \) if one of the following conditions is satisfied:

(i) \( C \) is a clause in \( P_{new} \). Then, each atom in the body of \( C \) is inherited from \( P_0 \).

(ii) Let \( C \) be the result of unfolding in \( P_i \). Suppose that \( C_+ \) in \( P_{-1} \) is the unfolded clause of the form: \( A \leftarrow B, B_1, \ldots, B_n \), and that \( C_- \) in \( P_{i-1} \) is one of the unfolding clauses of the form: \( B' \leftarrow K \). Thus, \( C \) is of the form: \( A \theta \leftarrow K \theta, B_1 \theta, \ldots, B_n \theta \), where \( \theta \) is an mgu of \( B \) and \( B' \). Then, each atom \( B_j \theta \) (\( 1 \leq j \leq n \)) in \( C \) is inherited from \( P_0 \) if \( B_j \) in \( C_+ \) is inherited from \( P_0 \).

(iii) Let \( C \) be the result of folding in \( P_i \). Suppose that \( C_+ \) in \( P_{i-1} \) is the folded clause of the form: \( A \leftarrow K, B_1, \ldots, B_n \), and that \( D \) in \( P_{new} \) is the folding clause of the form: \( B \leftarrow K' \). Thus, \( C \) is of the form: \( A \leftarrow B \theta, B_1, \ldots, B_n \), where \( \theta \) is an mgu such that \( K' \theta = K \). Then, each atom \( B_j \) (\( 1 \leq j \leq n \)) in \( C \) is inherited from \( P_0 \) if \( B_j \) in \( C_+ \) is inherited from \( P_0 \).

\(^2\) Note that the converse does not hold in general, that is, \( \text{comp}(P_N) \not\subseteq \text{comp}(P_0) \).
Intuitively, an inherited atom is (a possibly instantiated version of) an atom such that it was in the body of some clause in \( P_{\text{new}} \) and no unfolding has been applied to it.

**Example 2.2.2.** In Example 2.1.1, both "reach(X, N)" and "reach(Y, N)" in the body of \( C_3 \) are inherited atoms. In the body of clause \( C_4 \) (Example 2.1.2), atom "reach(Y, N)" is inherited from \( P_0 \), while neither "arc(X, X_i)" nor "reach(X_i, N)" is inherited from \( P_0 \).

Now, we can define a *modified* folding rule.

**Definition 2.2.3 (modified folding).** Let \( C \) and \( D \) be defined similarly in Definition 2.1.3, namely, \( C \) is a clause in \( P_i \) of the form: \( A \leftarrow K, L \) and \( D \) is a clause in \( P_{\text{new}} \) of the form: \( B \leftarrow K' \). Suppose that there exists a substitution \( \theta \) satisfying the following conditions:

1. (F1), (F2) and (F3) are the same as those defined in Definition 2.1.3.
2. (F4') Either the predicate of \( A \) is an old predicate, or there is no atom in \( K \) which is inherited from \( P_0 \).

**Example 2.2.3 (continued from Example 2.1.2).** Consider clause \( C_5 \) in Example 2.1.2. As noted in Example 2.2.2, atom "reach(Y, N)" in its body is inherited from \( P_0 \), thus the modified folding does not allow it to be folded by \( C_5 \). Instead, by unfolding \( C_5 \) at atom "reach(Y, N)" in its body, program \( P_2^\text{m} = \{C_1, C_2, C_4, C_7\} \cup DB \) is obtained, where

\[
C_7: \quad br(X, Y, N) \leftarrow arc(X, X_i), arc(Y, Y_i), reach(X_i, N), reach(Y_i, N).
\]

Now, atom "reach(Y_i, N)" in the body of \( C_7 \) is not inherited from \( P_0 \), so that the modified folding is now applicable to \( C_7 \). That is, by folding the body of \( C_7 \) by \( C_3 \), program \( P_3^\text{m} = \{C_1, C_2, C_4, C_8\} \cup DB \) is obtained, where

\[
C_8: \quad br(X, Y, N) \leftarrow arc(X, X_i), arc(Y, Y_i), br(X_i, Y_i, N).
\]

Hereafter, except in Section 4, by *folding* we mean the modified folding defined in Definition 2.2.3, and by a *transformation sequence*, we mean the one obtained by applying either unfolding or modified folding.

### 2.2.2. Preservation of FF for definite clauses

In this subsection, we show that the unfold/fold transformation (using modified folding) guarantees the total correctness w.r.t. \( FF \) for definite programs. We need one more definition and a lemma.

**Definition 2.2.4 (\( P_{\text{new}} \)-expansion).** Let \( A \) be an atom. Suppose that \( \bar{A} \) is defined as follows:

1. When \( A \) is an old atom, \( \bar{A} \) is \( A \) itself.
2. When \( A \) is a new atom, \( \bar{A} \) is either \( A \), or a sequence of atoms "\( B_1, \theta, \ldots, B_n, \theta \)" such that there exists a (variant of a) clause in \( P_{\text{new}} \) of the form: \( A_0 \leftarrow B_1, \ldots, B_n \) and \( \theta \) is an mgu of \( A \) and \( A_0 \).
Then, $\bar{A}$ is called a $P_{new}$-expansion of $A$.

Similarly, let $L$ be a sequence of atoms of the form: $A_1, \ldots, A_k$. Then a sequence of atoms $\bar{A}_1, \ldots, \bar{A}_k$ is called a $P_{new}$-expansion of $L$, and is denoted by $\bar{L}$.

**Example 2.2.4** (continued from Example 2.1.1). Since "reach($X, Y$)" is an old atom, a $P_{new}$-expansion of "reach($X, Y$)" is itself. On the other hand, a $P_{new}$-expansion of $br(a, Y, N)$ is either itself, or a sequence of atoms "reach($a, N$), reach($Y, N$)".

**Lemma 2.2.1** ($P_0$-simulation of SLD-derivation in $P_N$). Let $P_0, \ldots, P_N$ be a transformation sequence. Let $G$ be a goal, and suppose that there exists an SLD-derivation $Dr$ of $P_N \cup \{G\}$, $G_0 = G_1, \ldots, G_k, \ldots$ using input clauses in $P_N$ and substitutions $\theta_1, \ldots, \theta_k, \ldots$. Then, there exists an SLD-derivation $Dr_0$ of $P_0 \cup \{G\}$, $F_0 = F_1, \ldots, F_l, \ldots$ using input clauses in $P_0$ and substitutions $\sigma_1, \ldots, \sigma_l, \ldots$, satisfying the following conditions:

(i) For each $k (k \geq 0)$, there exists some $l(\geq 0)$ such that $F_l \sigma_1 \ldots \sigma_l$ is an $P_{new}$-expansion of $G_k \theta_1 \ldots \theta_k$, and

(ii) the restriction of $\sigma_1 \ldots \sigma_l$ to the variables in $G$ is the same as that of $\theta_1 \ldots \theta_k$.

(iii) (fairness) Furthermore, if the SLD-derivation $G_0 = G_1, \ldots, G_k, \ldots$ is fair, then so is the SLD-derivation $F_0 = F_1, \ldots, F_l, \ldots$.

$Dr_0$ is called a $P_0$-simulation of $Dr$.

The proof is shown in Appendix A.3, where a stronger version of the lemma is proved.

**Example 2.2.5.** Consider an SLD-derivation $Dr_3$ of $P_3'^n \cup \{G_0 = \leftarrow br(a, b, e)\}$, where $P_3'^n$ was given in Example 2.2.3. See the right-hand side in Fig. 1. $Dr_3$ has a $P_0$-simulation $F_0 = G_0, F_1, F_2, \ldots, F_6$, which is shown in the left-hand side in the

\[
\begin{align*}
F_0 : \leftarrow & \ br(a, b, e) \\
& \mid C_3 \\
F_1 : \leftarrow & \ reach(a, e), reach(b, e) \\
& \mid C_2 \\
F_2 : \leftarrow & \ arc(a, X_1), reach(X_1, e), reach(b, e) \\
& \mid C_2 \\
F_3 : \leftarrow & \ arc(a, X_1), reach(X_1, e), arc(b, Y), reach(Y_1, e) \\
& \mid C_2 \\
F_4 : \leftarrow & \ arc(a, X_1), reach(X_1, e), \ reach(d, e) \\
& \mid C_2 \\
F_5 : \leftarrow & \ arc(a, X_1), arc(X_1, X_2), reach(X_2, e), reach(d, e) \\
& \mid C_2 \\
F_6 : \leftarrow & \ arc(a, X_1), arc(X_1, X_2), reach(X_2, e), arc(d, Y_2), reach(Y_2, e) \\
& \mid C_2 \\
& \leftarrow \ fail \\

G_0 : \leftarrow & \ br(a, b, e) \\
& \mid C_3 \\
G_1 : \leftarrow & \ arc(a, X_1), arc(b, Y_1), br(X_1, Y_1, e) \\
& \mid C_2 \\
G_2 : \leftarrow & \ arc(a, X_1), br(X_1, d, e) \\
& \mid C_2 \\
G_3 : \leftarrow & \ arc(a, X_1), arc(X_1, X_2), arc(d, Y_2), br(X_2, Y_2, e) \\
& \mid C_2 \\
& \leftarrow \ fail
\end{align*}
\]

Fig. 1. $P_0$-simulation (left) of an SLD-derivation of $P_3'^n \cup \{\leftarrow br(a, b, e)\}$ (right).
We can now show the total correctness w.r.t. FF for definite programs.

**Proposition 2.2.2** (Total correctness w.r.t. FF). Let \( P_0, \ldots, P_N \) be a transformation sequence. Then, \( FF(P_0) \subseteq FF(P_N) \) for all \( N \geq 0 \).

**Proof.** For simplicity of explanation, we assume here that \( G \) is a ground atom (a more general case is shown in Proposition 3.3.1). Suppose that an SLD-tree of \( P_0 \cup \{ \leftarrow G \} \) is finitely failed. Suppose further that \( P_N \cup \{ \leftarrow G \} \) has a fair SLD-tree which is not finitely failed. Obviously, no SLD-derivation \( P_N \cup \{ \leftarrow G \} \) ever succeeds; otherwise, a \( P_0 \)-simulation of such a derivation would also succeed, which is a contradiction. Let \( BR \) be any nonfailed infinite branch in the fair SLD-tree for \( P_N \cup \{ \leftarrow G \} \). From Lemma 2.2.1, there exists a fair SLD-derivation \( Dr_0 \) of \( P_0 \cup \{ \leftarrow G \} \) which is a \( P_0 \)-simulation of \( BR \). Thus, \( Dr_0 \) is a nonfailed fair infinite derivation. From the result in [8], \( G \) is in the SLD finite failure set of \( P_0 \) iff every fair SLD-tree for \( P_0 \cup \{ \leftarrow G \} \) is finitely failed. Thus, \( Dr_0 \) should be finitely failed, which is a contradiction. \( \square \)

### 3. Unfold/fold transformation of stratified programs

#### 3.1. Preliminaries

We now consider an extension of the unfold/fold transformation from definite programs to stratified programs.

**Definition 3.1.1** (stratified program). A general logic program \( P \) is stratified if its predicates can be partitioned into levels so that, in every program clause, \( p \leftarrow L_1, \ldots, L_n \), the level of every predicate in a positive literal is less than or equal to the level of \( p \) and the level of every predicate in a negative literal is less than the level of \( p \) [1].

Throughout this paper, we assume that the levels of a stratified program are \( 1, \ldots, r \) for some integer \( r \), where \( r \) is the minimum number satisfying the above definition. In this case, \( P \) is said to have the maximum level \( r \) and is denoted \( P = \mathcal{P}^1 + \cdots + \mathcal{P}^r \), where \( \mathcal{P}^i \) is a set of clauses whose head predicates have level \( i \). Note that \( \mathcal{P}^1 \) is a set of definite clauses. When \( L \) is a literal whose predicate has level \( i \), we denote it \( \text{level}(L) = i \). Furthermore, the stratum [12] of a goal is defined as follows. For any positive atom \( A \), let \( \text{stratum}(A) = \text{level}(A) \) and \( \text{stratum}(\neg A) = \text{stratum}(A) + 1 \). Suppose that \( G \) is a goal of the form: \( \leftarrow L_1, \ldots, L_n \), where \( n \geq 0 \) and \( L_i \)'s are literals. Then, \( \text{stratum}(G) \) is 0 if \( G \) is empty, and \( \max\{\text{stratum}(L_i); 1 \leq i \leq n\} \), otherwise.
As in the previous section, we need to define an initial program, unfolding/folding and a transformation sequence for stratified programs. Although they are almost the same as the previous ones, we impose further restrictions on an initial stratified program.

Definition 3.1.2 (initial (stratified) program). An initial (stratified) program \( P_0 \) is a stratified program satisfying the following conditions:
- (I1) and (I2) are the same as those defined in Definition 2.1.1, and
- (I3) The definition of each new predicate consists of exactly one clause.

Condition (I3) above guarantees that a stratified program is also stratified after the unfold/fold transformation as shown below (Proposition 3.1.1), and most cases found in the literature seem to satisfy this condition.

Unfolding, (modified) folding and a transformation sequence are the same as those defined in Definition 2.1.2, Definition 2.2.3 and Definition 2.1.4, respectively. First, we have to confirm that our unfold/fold transformation preserves a stratification of an initial program.

Proposition 3.1.1 (preservation of stratification). Let \( P_0, \ldots, P_N \) be a transformation sequence. Then, if \( P_0 \) is a stratified program, so is \( P_i \) (\( N \geq i \geq 0 \)).

Proof. Let \( p \) be a new predicate, and let \( C \in P_{new} \) be its definition of the form: 
\[ p \leftarrow L. \]
Then, we define the level of \( p \) by:
\[ \text{level}(p) = \max \{ \text{level}(B_j) \mid B_j \in L \} \]
Then, the proposition is obvious from the definitions of unfolding and folding. \( \square \)

Example 3.1.1. Consider the following program:

- \( C_9: \quad \text{path}(X, [X]) \leftarrow \text{node}(X) \);
- \( C_{10}: \quad \text{path}(X, [X \mid L]) \leftarrow \text{arc}(X, Y), \text{path}(Y, L) \);
- \( C_{11}: \quad \text{good_list}([\_]) \);
- \( C_{12}: \quad \text{good_list}([X \mid L]) \leftarrow \neg \text{bad}(X), \text{good_list}(L) \);
- \( C_{13}: \quad \text{good_path}(X, L) \leftarrow \text{path}(X, L), \text{good_list}(L) \);

where predicates \( \text{node} \) and \( \text{arc} \) are supposed to be defined by a set of unit clauses, and the definition of predicate \( \text{bad} \) is not material, but its level is assumed to be less than 
\[ \max \{ \text{level}(\text{path}), \text{level}(\text{good_list}) \} \]

Suppose that a graph is given whose relationship of nodes and arcs is specified by the predicates \( \text{node} \) and \( \text{arc} \), respectively. Then, predicate \( \text{good_path}(X, L) \) can be thought of as finding a path \( L \) such that it starts from node \( X \) and each node of \( L \) is a "good" (or not "bad") one.
Let $P_{aux}$ be a set of definitions of predicates $node$, $arc$ and $bad$, and let $P^{\text{tr}}_0$ be $\{C_9, C_{10}, C_{11}, C_{12}, C_{13}\} \cup P_{aux}$. Moreover, let $P_{old} = \{C_9, C_{10}, C_{11}\} \cup P_{aux}$, and $P_{new} = \{C_{13}\}$. Then, $P^{\text{tr}}_0$ satisfies the conditions of an initial stratified program.

By unfolding $C_{13}$ at atom “path($X$, $L$)” in its body, the following clauses $\{C_{14}, C_{15}\}$ are obtained, where

$$C_{14}: \text{good}_{-}\text{path}(X, [X]) \leftarrow \text{node}(X), \text{good}_{-}\text{list}([X]).$$

$$C_{15}: \text{good}_{-}\text{path}(X, [X | L]) \leftarrow \text{arc}(X, Y), \text{path}(Y, L), \text{good}_{-}\text{list}([X | L]).$$

Both $C_{14}$ and $C_{15}$ can be further unfolded, and we have:

$$C_{16}: \text{good}_{-}\text{path}(X, [X]) \leftarrow \text{node}(X), \neg \text{bad}(X).$$

$$C_{17}: \text{good}_{-}\text{path}(X, [X | L]) \leftarrow \text{arc}(X, Y), \text{path}(Y, L), \neg \text{bad}(X), \text{good}_{-}\text{list}(L).$$

By folding the body of $C_{17}$ by $C_{13}$, program $P^{\text{tr}}_3 = P^{\text{tr}}_0 - \{C_{13}\} \cup \{C_{16}, C_{18}\}$ is obtained, where

$$C_{18}: \text{good}_{-}\text{path}(X, [X | L]) \leftarrow \text{arc}(X, Y), \neg \text{bad}(X), \text{good}_{-}\text{path}(Y, L).$$

### 3.2. Partial correctness of transformation

The success set ($SS$) and the finite failure ($FF$) set of a stratified program are defined similarly to those of a definite program. That is, $SS$ ($FF$) of a stratified program is defined by replacing “SLD-derivation (SLD-tree)” in Definition 2.1.5 (Definition 2.2.1) with “SLDNF-derivation (SLDNF-tree)” [9], respectively.

In this subsection, we show the partial correctness of our transformation w.r.t. both $SS$ and $FF$.

**Proposition 3.2.1** (partial correctness w.r.t. $SS$ and $FF$). Let $P_0, \ldots, P_N$ be a transformation sequence. Then, for $i = 0, \ldots, N - 1$,

(SS): if $SS(P_i) = SS(P_0)$, then $SS(P_{i+1}) \subseteq SS(P_i)$.

(FF): if $FF(P_i) = FF(P_0)$, then $FF(P_{i+1}) \subseteq FF(P_i)$.

The proof of the above proposition is shown in Appendix A.2.

### 3.3. Total correctness of transformation

#### 3.3.1. Total correctness w.r.t. $FF$

We now show the total correctness of our unfold/fold transformation. We prove the total correctness w.r.t. $FF$ first. As in the case for definite programs, we show Lemma 2.2.1 for stratified programs, replacing “SLD-derivation” in it with “SLDNF-derivation”. That is,
Lemma 3.3.1 (P_o-simulation of SLDNF-derivation in P_N). Let P_0, . . . , P_N be a transformation sequence. Let G be a goal, and suppose that there exists an SLDNF-derivation Dr of P_N ∪ {G}, G_0 = G, . . . , G_k, . . . using input clauses in P_N and substitutions θ_1, . . . , θ_k, . . . Then, there exists an SLDNF-derivation Dr_o of P_o ∪ {G}, F_0 = G, . . . , F_i, . . . using input clauses in P_o and substitutions σ_1, . . . , σ_i, . . ., satisfying the following conditions:

(i) For each k (k ≥ 0), there exists some l(≥ 0) such that F_lσ_i . . . σ_l is a P_new-expansion of G_kθ_1 . . . θ_k, and

(ii) the restriction of σ_1 . . . σ_i to the variables in G is the same as that of θ_1 . . . θ_k.

(iii) (fairness) Furthermore, if the SLDNF-derivation G_0 = G, . . . , G_k, . . . is fair, then so is the SLDNF-derivation F_0 = G, . . . , F_i, . . .

Dr_o is called a P_o-simulation of Dr.

The proof is given in Appendix A.3. Now we can show the total correctness w.r.t. FF.

Proposition 3.3.1 (total correctness w.r.t. FF). Let P_0, . . . , P_N be a transformation sequence, where P_0 is an initial stratified program. Then, for all N ≥ 0, FF(P_0) ⊆ FF(P_N).

Proof. Suppose that an SLDNF-tree of P_0 ∪ {←A} is finitely failed. Obviously, no SLDNF-derivation P_0 ∪ {←A} ever succeeds. Furthermore, it does not flounder, from the proposition shown by Shepherdson [14], which says that, if a query Q flounders under a computation rule, then it cannot fail under any computation rule.

Suppose that P_N ∪ {←A} has a fair SLDNF-tree which is not finitely failed. Let BR_N be any nonfailed branch in that fair SLDNF-tree for P_N ∪ {←A}.

From Lemma 3.3.1, there exists a fair SLDNF-derivation BR_0 for P_0 ∪ {←A} which is a P_o-simulation of BR_N. BR_0 neither succeeds nor flounders as noted above. Thus, BR_0 is a nonfailed fair infinite derivation. Then, we can show that comp(P_0) ∪ {∃A} has a model, using similar methods in the proofs of completeness of negation as failure rule by [9, 21], which is a contradiction. □

3.3.2. Total correctness w.r.t. SS

Finally, we state the total correctness w.r.t. SS, whose proof is given in Appendix A.4.

Proposition 3.3.2 (total correctness w.r.t. SS). Let P_0, . . . , P_N be a transformation sequence, where P_0 is an initial stratified program. Then, SS(P_0) ⊆ SS(P_N) for all N ≥ 0.

4. On preservation of perfect model semantics

The semantics we have considered is somewhat operational, in that the success set and the finite failure set of a stratified program are given by specific procedures
such as SLD(NF)-resolution. In this section, we consider more declarative semantics, that is, the standard (minimal Herbrand) model $M_p$ [1, 17], or, more generally, the perfect model semantics for stratified programs introduced by Przymusinski [12].

It seems to be a more direct extension from Tamaki-Sato's original unfold/fold rules to consider transformation rules preserving the equivalence of $M_p$ or the perfect model semantics, since their framework preserves the least Herbrand model for a definite program. Recall that, Tamaki-Sato's unfold/fold transformation does not preserve the finite failure set. However, from the viewpoint of the perfect model semantics, it poses no problems, since a goal: "←$G$" which has neither a successful SLD-derivation nor a finite failed SLD-tree is simply considered to be false. We assume familiarity with the perfect model semantics (see [12]).

Definitions of an initial program, unfolding rule and folding rule are the same as those in Definition 3.1.2, Definition 2.1.2 and Definition 2.1.3, respectively. Note that we do not have to consider the modified folding rule. A transformation sequence is also defined similarly to Definition 2.1.4. Then, we have the following proposition.

**Proposition 4.1** (preservation of perfect model semantics). *The perfect model semantics of any program $P_i$ in a transformation sequence starting from initial program $P_0$, is identical to that of $P_0$.*

The proof is given in Appendix A.5.

5. Conclusion

There have been several studies on equivalence-preserving transformation of logic programs. Tamaki and Sato's result [16] and its elaboration by Kawamura and Kanamori [7] are already described in Section 2.1.1. Maher extensively studied various formulations of equivalence for definite programs [10]. In that paper, he considered a transformation system similar to that of Tamaki and Sato, and stated that his unfold/fold rules preserve logical equivalence of completions, while, as stated in Section 2.2.1, those of Tamaki-Sato do not preserve it. Kanamori and Horiuchi [6] proposed a framework for transformation and synthesis based on generalized unfold/fold rules. Their system was shown to preserve the minimum Herbrand model semantics, but the finite failure set is not preserved in general. In a very recent paper, Gardner and Shepherdson [4] proposed a framework for unfold/fold transformation of normal programs, where negative literals are allowed in the bodies of clauses, and they showed that their transformation preserves procedural equivalence based on SLDNF-resolution. Their work, however, is not comparable with our version, nor with that of [16] and [7]; their folding rule [4] specifies that, when a program $P_{i+1}$ is obtained from $P_i$ by folding $C \in P_i$ by $D$, $D$ should be in $P_i$, while, in our framework like [16] and [7], $D$ is not necessarily in $P_i$. 
Compared with previous work, the contributions of this paper will be summarized as follows:

(1) The modified folding rule for a definite program was proposed. The unfolding rule together with the modified folding rule was shown to preserve the finite failure set (by SLD-resolution) of a program as well as the success set. This guarantees a safer use of Tamaki–Sato’s transformation when negation as failure rule is used.

(2) The unfold/fold rules for stratified programs were proposed. The modified folding rule has made it possible to extend the applicability of unfold/fold transformation rules to a stratified program, so that they preserve both the success set and the finite failure set of a stratified program by SLDNF-resolution.

(3) Preservation of equivalence of the perfect model semantics was discussed. We showed that unfold/fold rules by Tamaki and Sato can be extended to rules for a stratified program and preserve the equivalence of the perfect model semantics.

Appendix A.

A.1. Preliminaries

In the following proofs, for the case of understanding and simplicity, we sometimes use such a representation that unifiers in SLD(NF)-resolution appear only implicitly and instead, we write the equations corresponding to the unifiers explicitly. For example, let $G_0 = \leftarrow B_1, \ldots, B_k, \ldots, B_n$ be a goal in an SLD(NF)-resolution, where $B_k$ is the selected (positive) atom and $C$ is an input clause $H \leftarrow \Gamma$. Then, the derived goal $G_1$ from $G_0$ and $C$ is written:

$\leftarrow B_1, \ldots, B_{k-1}, \Gamma, B_{k+1}, \ldots, B_n, B_k = H.$

Namely, an mgu $\theta$ of $B_k$ and $H$ is not applied to $G_1$, but the equation $B_k = H$ corresponding to $\theta$ is added at the end of the goal. This formulation of SLD-resolution was proposed and studied by [18]. Since properties of this formulation play a crucial role in our proofs, we cite here the relationship between a usual SLD-derivation and the above formulation [18].

Consider an SLD-derivation $Dr$. Let $(A_0, A_1, \theta_0, \ldots, A_n, \theta_0 \circ \cdots \circ \theta_n)$ be the list of selected atoms in the goals of $Dr$, written in the order in which they have been selected and let $(H_0, H_1, \ldots, H_n)$ be the list of corresponding heads of the input clauses used in the derivation and $(\theta_0, \theta_1, \ldots, \theta_n)$ the list of the mgu’s such that

$A_0 \theta_0 = H_0 \theta_0,$

$A_n \theta_0 \circ \theta_1 \circ \cdots \circ \theta_n \circ \theta_n = H_n \theta_n.$

We assume the process of standardizing the variables in the input clauses apart as usual. Then, $\theta_0, \ldots, \theta_{n-1}$ do not affect $H_n$, so that $H_n \theta_n = H_n \theta_0 \circ \theta_1 \circ \cdots \circ \theta_n$. The sequence of identities built by the SLD-derivation can therefore be rewritten as

$A_0 \theta_0 = H_0 \theta_0,$

$A_n \theta_0 \circ \theta_1 \circ \cdots \circ \theta_n = H_n \theta_0 \circ \theta_1 \circ \cdots \circ \theta_n.$
It then shows that the SLD-derivation attempts to compute an mgu \( \theta = \theta_0 \circ \theta_1 \circ \cdots \circ \theta_n \) (if it exists) which is a solution to the set of equations:

\[ \mathcal{S} = \{ A_0 = H_0, A_1 = H_1, \ldots, A_n = H_n \}. \]

On the other hand, when we consider a variant \( Dr' \) of \( Dr \) where each mgu is not applied to a goal but an equation corresponding to the mgu appears explicitly, we have exactly the same set of equations \( \mathcal{S} \) in the last goal of \( Dr' \).

Due to the unification theorem [13, 11], \( \mathcal{S} \) gives the same mgu as \( \theta \) modulo renaming if and only if it exists. Moreover, the order in which the substitutions are computed is immaterial. It is easy to see that this discussion can be extended to the case of SLDNF-derivation.

Based on this observation, we sometimes utilize the following notation. Let \( r \) be a sequence of literals, \( \mathcal{S} \) a set of equations such that it gives an mgu \( \theta \). Then, an expression \( F \) of the form \( l^\equiv \) is denoted also by "\( r, \mathcal{S} \)". We call \( r \) the literal part of \( F \), while \( \mathcal{S} \) is called the equation part of \( F \). As an example of this formulation, we prove the following lemma.

**Lemma A.1.1.** Let \( C_+ \) (resp. \( C_- \)) be a clause in a program \( P \) of the form: \( H \leftarrow B_+ \), \( L \) (resp. \( B_- \leftarrow K \)) such that \( B_+ \) is unifiable with \( B_- \) by an mgu \( \theta \), and \( C_+ \) shares no variables with \( C_- \). Let \( G \) be a goal \( \leftarrow A, \Delta \), where \( A \) is an atom unifiable with \( H \), and \( \Delta \) is a (possibly empty) sequence of literals, and variables in \( G \) appear neither in \( C_+ \) nor in \( C_- \). Consider an SLDNF-derivation of \( P \cup \{ G \} \) consisting of goals \( G_0 = G, G_1, G_2 \), where \( G_1 \) (resp. \( G_2 \)) is derived from \( G_0 \) (resp. \( G_1 \)) and \( C_+ \) (resp. \( C_- \)), selecting \( A \) (resp. possibly an instantiated version of \( B_+ \)).

On the other hand, let \( C \) be the result of applying unfolding to \( C_+ \) at \( B_+ \) by \( C_- \), i.e., \( C \) is the clause of the form: \( H \theta \leftarrow K \theta, L \theta \). Consider a resolvent \( G'_1 \) of \( G \) and \( C \), selecting \( A \). Then \( G_2 \) is equivalent to \( G'_1 \) modulo variable renaming.

**Proof.** Using the above-mentioned notation, \( G_1 \) and \( G_2 \) can be written as follows:

\[
G_1: \quad \leftarrow B_+, L, \Delta, H = A
\]

\[
G_2: \quad \leftarrow K, L, \Delta, B_+ = B_-, H = A.
\]

On the other hand, \( G'_1 \) is of the form

\[
G'_1: \quad \leftarrow K\theta, L\theta, \Delta, H\theta = A.
\]

Since the equation \( B_+ = B_- \) gives the substitution \( \theta \) and variables among \( A \) and \( \Delta \) are not affected by \( \theta \), \( G_2 \) can be rewritten as \( \leftarrow K\theta, L\theta, \Delta, H\theta = A \), which is equivalent to \( G'_1 \). \( \square \)

We prove one more technical lemma.

**Lemma A.1.2.** Let \( C \) be a clause of the form: \( H \leftarrow J, K \) and \( D \) a clause of the form: \( B \leftarrow J_0 \) such that \( J_0 \theta = J \) for some substitution \( \theta \), and \( C, D \) and \( \theta \) satisfy the conditions
of folding (F1)-(F4) in Definition 2.1.3. Let \( D' \) be a variant of \( D \) of the form: \( B' \leftarrow J_0' \) such that variables in \( D' \) appear neither in \( C \) nor in \( D \). Then, \( J \) is a variant of \( J_0 \tau \), where \( \tau \) is an mgu of \( B' \) and \( B \theta \) such that \( B' \tau = B \theta \). Moreover, \( J \) is different from \( J_0 \tau \) only with respect to those variables in \( C \) which occur only in \( J \) but neither in \( H \) nor in \( K \).

**Proof.** Let \( x_i \) (resp. \( x'_i \)) be those internal variables which occur only in \( J_0 \) (resp. \( J_0' \)) but not in \( B \) (resp. \( B' \)), and let \( y_j \) (resp. \( y'_j \)) be those variables which occur in \( B \) (resp. \( B' \)) \((i, j \geq 0)\). We thus denote \( J_0 \) (resp. \( J_0' \)) by \( J_0(x_i; y_j) \) (resp. \( J_0(x'_i; y'_j) \)). From the conditions of folding, substitution \( \theta \) can be written in the form: \( \theta = \theta_{iv} \cup \theta_{iv} \), where \( \theta_{iv} = \{x_i/z_i\} \) and \( \theta_{iv} = \{y_j/t_j\} \) such that

1. \( \theta_{iv} \) is a renaming substitution and each variable \( z_j \) appears only in \( J \) but in none of \( H \), \( K \), and \( \gamma \theta \), and
2. \( t_j \) does not contain any \( z_i \).

Therefore, \( \tau \) is equivalent to \( \theta_{iv} = \{y_j/t_j\} \). Thus, \( J_0 \tau = J_0 \theta_{iv} = J_0(x_i; t_j) \). On the other hand, \( J = J_0 \theta = J_0(x_i; y_j) \{x_i/z_i\} \cup \{y_j/t_j\} = J_0(z_i; t_j) \). Comparing \( J_0(x'_i; t_j) \) with \( J_0(z_i; t_j) \), the lemma follows. \( \square \)

### A.2. Proofs of partial correctness

Instead of proving Proposition 3.2.1, we show a more general proposition. For this, we also generalize the definitions of the success set and the finite failure set of a given program as follows.

**Definition A.2.1 (success set).** Let \( P \) be a program and \( \Gamma \) a sequence of literals. The set of all pairs \((\Gamma, \sigma)\) such that there exists a successful SLDNF-derivation of \( P \cup \{\neg \Gamma\} \) with computed answer substitution \( \sigma \), is called the **success set** of \( P \), and is denoted by \( SS(P) \).

**Definition A.2.2 (finite failure (FF) set).** Let \( P \) be a program and \( \Gamma \) a sequence of literals. The set of all \( \Gamma \) such that there exists a finitely failed SLDNF-tree for \( P \cup \{\neg \Gamma\} \), is called the **(SLDNF) finite failure set** of \( P \), and is denoted by \( FF(P) \).

Moreover, we use the following notation convention. Let \( G \) be a goal of the form: \( \neg A \), where \( A \) is a (possibly empty) sequence of literals. Then, when \( (A, \sigma) \in SS(P) \) (resp. \( A \in FF(P) \)) holds, we denote it simply by \( (G, \sigma) \in SS(P) \) (resp. \( G \in FF(P) \)).

**Proposition A.2.1 (partial correctness w.r.t. SS and FF).** Let \( P_0, \ldots, P_n \) be a transformation sequence. Then,

- \( \text{(SS)}: \) If \( SS(P_i) = SS(P_0) \), then \( SS(P_{i+1}) \subseteq SS(P_i) \) for \( i = 0, \ldots, N - 1 \).
- \( \text{(FF)}: \) If \( FF(P_i) = FF(P_0) \), then \( FF(P_{i+1}) \subseteq FF(P_i) \) for \( i = 0, \ldots, N - 1 \).

**Proof.** The proof is by mutual induction on \( s = stratum(G_0) \) of goal \( G_0 \). It is obvious when \( s = 0 \). Suppose that the proposition has been proved for all goals \( G_0 \) whose \( stratum(G_0) \leq s \), where \( s \geq 0 \). We first prove (SS).
Suppose there exists an SLDNF-refutation $D_{r+1}$ of $P_{r+1} \cup \{G_0\}$ with the computed answer substitution $\sigma$, where $\text{stratum}(G_0)$ is $s + 1$. The proof is by induction on the length $^3$ of the SLDNF-refutation of $P_{r+1} \cup \{G_0\}$. Let $G_0 = \leftarrow A$, $\Delta$ where $A$ is a literal and $\Delta$ is a (possibly empty) sequence of literals. Suppose further that $A$ is the selected literal in $G_0$.

When $A = \sim A'$ is a negative atom, $A$ should be ground and there exists a finitely failed SLDNF-tree for $P_{r+1} \cup \{\leftarrow A'\}$ and $G_0$ has the successor $G_1 = \leftarrow \Delta$. Since $\text{stratum}(\leftarrow A')$ is less than $\text{stratum}(G_0)$, $P_r \cup \{\leftarrow A'\}$ has a finitely failed SLDNF-tree from the induction hypothesis on the partial correctness w.r.t. FF. Let $D_r$ be an SLDNF-derivation of $P_r \cup \{G_0\}$. Then $G_0$ has the successor $G_1$ also in $D_r$. From the induction hypothesis of the length of an SLDNF-refutation, $(G_1, \sigma) \in SS(P_r)$, thus $(G_0, \sigma) \in SS(P_r)$.

Next, suppose that $A$ is a positive atom. Let $C$ be the input clause.

Case 1: $C$ is inherited from $P_r$. Then, the proof is obvious from the induction hypothesis.

Case 2: $C$ is the result of unfolding. Let $C_+ \in P_r$ be the unfolded clause of the form: $H \leftarrow B_+, L$ and $C_- \in P_r$ be the unfolding clause of the form: $B_- \leftarrow K$. Then, $C$ can be written as $H \theta \leftarrow K \theta, L \theta$, where $\theta$ is an mgu of $B_+$ and $B_-$. Then, in the SLDNF-refutation $D_{r+1}$ of $P_{r+1} \cup \{G_0\}$, $G_0$ has the successor $G_1$ of the form:

$$G_1: \leftarrow K \theta, L \theta, \Delta, A = H \theta$$

and $(G_1, \sigma) \in SS(P_{r+1})$. On the other hand, consider an SLDNF-derivation $D_r$ of $P_r \cup \{G_0\}$. Using $C_-$ as an input clause, $G_0$ has the successor $G'_1$ of the form:

$$G'_1: \leftarrow B_+, L, \Delta, A = H.$$

Again, using $C_-$ as an input clause, $G'_1$ has the successor $G'_2$ of the form:

$$G'_2: \leftarrow K, L, \Delta, B_+ = B_-, A = H.$$

Since $(G_1, \sigma) \in SS(P_r)$ from the induction hypothesis and $G_1$ is equivalent (modulo renaming) to $G'_1$ from Lemma A.1.1, it is shown that $(G_0, \sigma) \in SS(P_r)$.

Case 3: $C$ is the result of folding. Let $C_+ \in P_r$ be the folded clause of the form: $H \leftarrow J, K$ and $D \in P_{new}$ be the folding clause of the form: $B \leftarrow J_0$, where $J_0 \theta = J$ for some substitution $\theta$. Then $C$ is $H \leftarrow B \theta, K$. In the SLDNF-refutation $D_{r+1}$ of $P_{r+1} \cup \{G_0\}$, $G_0$ has the successor

$$G_1: \leftarrow B \theta, K, \Delta, A = H.$$

Since $(G_1, \sigma) \in SS(P_r)$ from the induction hypothesis and $SS(P_r) = SS(P_0)$ from the assumption of the proposition, it follows that $(G_1, \sigma) \in SS(P_0)$. Let $G_2$ be the derived goal from $G_1$ and an input clause $D'$ in $P_0$ with $B \theta$ as the selected atom, where $D'$ is a variant of $D$, say, $D' = B' \leftarrow J_0'$ such that no variables in $D'$ appear elsewhere. Then, $G_2$ is of the form:

$$G_2: \leftarrow J_0', K, \Delta, A = H, B \theta = B'.$$

$^3$ The length of SLDNF-refutation is defined in a similar way to that of SLD-refutation (see [9]).
From the folding condition (F3), \( D' \) is the only clause in \( P_0 \) which is unifiable with \( B_{\theta} \), so \( (G_2, \sigma) \) is also in \( SS(P_0) \). Again, from the assumption of the proposition, \( (G_2, \sigma) \) is also in \( SS(P) \).

On the other hand, consider an SLDNF-derivation \( D_{r_i} \) of \( P_i \cup \{ G_0 \} \). Using \( C_+ \) as an input clause, \( G_0 \) has the successor \( G_i^1 \) of the form:

\[
G_i^1: \quad \leftarrow J_k, \Delta, A = H.
\]

From Lemma A.1.2, \( G_i^1 \) is a variant of \( G_i \). It follows that \( (G_i, \sigma) \in SS(P_i) \), thus \( (G_0, \sigma) \) is also in \( SS(P) \).

**Proof of (FF):** Let \( G_0 = \leftarrow A, \Delta \) be a goal, and suppose that there exists a finitely failed SLDNF-tree \( T_{r_{i+1}} \) for \( P_{i+1} \cup \{ G_0 \} \). We show that \( P_i \cup \{ G_0 \} \) also has a finitely failed SLDNF-tree \( T_{r_i} \). The proof is by induction on the size (the number of nodes) of \( T_{r_{i+1}} \). Suppose that \( A \) is the selected atom in \( G_0 \) of \( T_{r_{i+1}} \).

**Induction Basis:** Suppose that the size of \( T_{r_{i+1}} \) is 1. Then the following two cases are to be considered.

(i) \( A \) is a positive atom and there is no clause in \( P_{i+1} \) whose head is unifiable with \( A \). When \( P_i \) has no clause whose head is unifiable with \( A \), the proposition is obvious. Otherwise, since only unfolding might change the head of a clause during unfold/fold transformation, there exists only one clause \( C \) in \( P_i \) such that the head of \( C \) is unifiable with \( A \) and that, for the head \( H \) of each unfolded clause of \( C \), \( H \) is not unifiable with \( A \). In this case, it is straightforward to show that there exists a finitely failed SLDNF-tree for \( P_{i+1} \cup \{ \leftarrow A \} \).

(ii) \( A \) is a ground negative literal, say, \( \neg A' \), and there exists a successful SLDNF-derivation of \( P_{i+1} \cup \{ \leftarrow A' \} \). From the partial correctness w.r.t. \( SS \), \( (A', \varepsilon) \in SS(P) \), where \( \varepsilon \) is an empty substitution. Thus \( G_0 \in FF(P_i) \).

**Induction Step:** Suppose that the proposition has been proved for any goal whose finitely failed SLDNF-tree has the size less than \( t (\geq 1) \) and that the size of a finitely failed SLDNF-tree for \( P_{i+1} \cup \{ G_0 \} \) is \( t + 1 \).

(i) When \( A \) is a negative atom, say, \( \neg A' \), \( A' \) should be ground and there exists a finitely failed SLDNF-tree for \( P_{i+1} \cup \{ \leftarrow A' \} \). In this case \( P_{i+1} \cup \{ \leftarrow \Delta \} \) also has a finitely failed SLDNF-tree. Then, the proposition holds from the induction hypothesis.

(ii) Suppose that \( A \) is a positive atom. In the SLDNF-tree \( T_{r_i} \) for \( P_i \cup \{ G_0 \} \), let \( A \) be the selected atom in \( G_0 \). Let \( G_{i1}, \ldots, G_{ik} \) be the children of \( G_0 \) in \( T_{r_i} \), and \( C_1, \ldots, C_k \) the corresponding input clauses. We show that there exists a finitely failed SLDNF-tree for \( P_i \cup \{ G_{ij} \} \) for each \( j (j = 1, \ldots, k) \).

When \( C_j \) is inherited to \( P_{r_{i+1}} \), the proposition is obvious. When folding is applied to some \( C_j \), it is easy to see that the proposition holds from the similar discussion in the above proof of (SS) (case 3). Thus, we prove the proposition when unfolding is applied to some \( C_j \).

Let \( C_j \) be of the form: \( H \leftarrow B_{\theta}, L \) and suppose that unfolding is applied to \( B_{\theta} \). Let \( C_1', \ldots, C_n' \) \((n > 0)\) be all the clauses in \( P_i \) such that \( C_l' \) \((1 \leq l \leq n)\) is \( B_{\theta}' \leftarrow K_l \) and \( B_{\theta}' \) is unifiable with \( B_{\theta} \) by an mgu, say, \( \theta_l \). Then, the result of unfolding is
\( P_{i+1} = (P_i - \{C_j\}) \cup \{C_1, \ldots, C_n\} \), where \( C_i = H\theta_i \leftarrow K\theta_i, L\theta_i \). Note that \( G_{ij} \) is denoted by

\[
G_{ij} : \leftarrow B_+, L, \Delta, H = A.
\]

Consider an SLDNF-derivation of \( P_i \cup \{G_{ij}\} \) with \( B_+ \) as the selected atom. Since \( C_1, \ldots, C_n \) are all the clauses in \( P_i \) whose heads are unifiable with \( B_+ \), \( G_{ij} \) has the children \( G_{21}, \ldots, G_{2n} \), where

\[
G_{2j} : \leftarrow K_i, L, \Delta, H = A, B_+ = B'_j
\]

assuming that any variable in \( C'_j \) does not appear elsewhere. On the other hand, consider the finitely failed SLDNF-tree \( Tr_{i+1} \) for \( P_{i+1} \cup \{G_0\} \). Recall that \( A \) is the selected atom. Each \( C'_j \) has two cases: either \( A \) is unifiable with the head \( H\theta_i \) of \( C'_j \) or not. When it is not unifiable, the set of equations: \( \{H = A, B_+ = B'_j\} \) has no solution. Thus goal \( G_{2j} \) is finitely failed. Otherwise, \( G_0 \) in \( Tr_{i+1} \) has a child

\[
G_{ij}^{i+1} : \leftarrow K_i, L\theta_i, \Delta, H\theta_i = A
\]

which is finitely failed in \( P_{i+1} \). From the induction hypothesis on the size of a finitely failed tree, \( G_{ij}^{i+1} \) has a finitely failed SLDNF-tree also in \( P_i \). Since \( B_+ = B'_j \) gives the substitution \( \theta_i \), \( G_{ij}^{i+1} \) is equivalent to \( G_{2j} \). Thus, \( G_{2j} \) has also a finitely failed SLDNF-tree in \( P_i \). This completes the proof. \( \square \)

A.3. Proof of Lemma 3.3.1.

In this subsection, we give a proof of Lemma 3.3.1. For this, we first prove the following lemma, which says that, for a one-step SLDNF-derivation of \( P_N \cup \{\leftarrow A\} \), there exists a “corresponding” (possibly several steps) SLDNF-derivation of \( P_0 \cup \{\leftarrow A\} \), where \( P_0, \ldots, P_N \) is a transformation sequence and \( A \) is an atom. In the following, for a clause \( C \) of the form: \( H \leftarrow B_1, \ldots, B_m \), we denote its head \( H \) by \( head(C) \) and its body \( B_1, \ldots, B_m \) by \( body(C) \).

Lemma A.3.1 (P₀-simulation of one-step SLDNF-derivation in \( P_N \)). Let \( P_0, \ldots, P_N \) (\( N \geq 0 \)) be a transformation sequence and \( G_N^N = \leftarrow A \) be a goal, where \( A \) is an atom. Let \( C \) be a clause in \( P_N \) of the form: \( H \leftarrow B_1, \ldots, B_m \) (\( m \geq 0 \)) and let \( G_N^N \) be a resolvent of \( G_N \) and \( C \), written in the form: \( \leftarrow B_1, \ldots, B_m, A = H \).

(i) Then, there exists an SLDNF-derivation \( Dr_0 \) of \( P_0 \cup \{\leftarrow A\} \) consisting of \( G_0 = \leftarrow A, \ldots, G_k \) (\( k \geq 0 \)) using input clauses in \( P_0 \) and substitutions \( \sigma_1, \ldots, \sigma_k \) such that the literal part of \( G_k \) is a \( P_{new} \)-expansion of that of \( G_N^N \). Namely, the following condition is satisfied:

\begin{itemize}
  \item[(D1)] \( G_k \) is denoted by \( \leftarrow \tilde{B}_1, \ldots, \tilde{B}_m, A = H \), where there exists a bijection \( \varphi \) from the multiset \( \{B_1, \ldots, B_m\} \) to the multiset \( \{\tilde{B}_1, \ldots, \tilde{B}_m\} \) such that

\[
\varphi(B_i) = \begin{cases} 
\tilde{B}_i & \text{if } B_i \text{ is either an old atom or a negative literal} \\
body(D_i), \text{head}(D_i) = B_i & \text{otherwise}
\end{cases}
\]

where \( D_i \in P_{new} \) is a clause whose head is unifiable with \( B_i \) (\( i = 1, \ldots, m \)) (see Fig. 2).
\end{itemize}
(ii) Moreover, when \( A \) is a new atom, there exists a variant \( C_0 \) of some clause in \( P_{\text{new}} \) such that \( C_0 \) is \( A_0 \leftarrow L_0 \) and we can construct an SLDNF-derivation \( D_{r_0} \) of \( P_0 \cup \{ \leftarrow L_0, A_0 = A \} \), consisting of \( G_0^* = \leftarrow L_0, A_0 = A \), \( \ldots \), \( G_k^* (k' \geq 0) \) using input clauses in \( P_0 \) and substitutions \( \sigma_1', \ldots, \sigma_{k'}' \), satisfying the condition (D1) replacing \( k \) with \( k' \). Furthermore,

(D2) For each \( U \) in \( L_0 \) such that \( U \) is left unresolved in the SLDNF-derivation \( D_{r_0} \), let \( B_j \) (for some \( j, m \geq j \geq 0 \)) in \( G_k \) be the possibly instantiated version of \( U \). Then, \( B_j = \varphi^{-1}(B_j) \) is an inherited atom in \( C \).

**Proof.** The proof is done by induction on the length of a transformation sequence \( N \).

**Induction Basis:** The base case \((N = 0)\) trivially holds, since it suffices to consider the SLDNF-derivation of \( P_0 \cup \{ \leftarrow A \} \) using the same clause \( C \) as its input clause. Moreover, when \( A \) is a new atom, let \( C_0 \) be \( C \) itself. Then, it is easy to see that the above (ii) is satisfied.

**Induction Step:** Suppose that the above proposition holds until \( N - 1 \). We consider the following three cases.

**Case 1:** \( C \) is inherited from \( P_{N-1} \). Then it is obvious by the induction hypothesis.

**Case 2:** \( C \) is the result of unfolding. Let \( C \in P_{N-1} \) be the unfolded clause of the form: \( H \leftarrow B_+, J \) and \( C \in P_{N-1} \) the unfolding clause: \( B_+ \leftarrow K \). Then \( C = H \theta \leftarrow K \theta, J \theta \), where \( \theta \) is an mgu of \( B_+ \) and \( B_- \). The resolvent \( G_1^N \) of \( C \) and \( G_0^N = \leftarrow A \) can be denoted by

\[
G_1^N: \leftarrow K \theta, J \theta, H \theta = A.
\]

(i) First, we show that there exists an SLDNF-derivation \( D_{r_0} \) of \( P_0 \cup \{ \leftarrow A \} \) satisfying the condition (D1). Consider an SLDNF-derivation of \( P_{N-1} \cup \{ \leftarrow A \} \). Using \( C_+ \) as an input clause, \( G_0^{N-1} = \leftarrow A \) has the successor \( G_1^{N-1} \) of the form: \( \leftarrow B_+, J, H = A \). Since the above lemma holds for \( P_{N-1} \) from the induction hypothesis, \( P_0 \cup \{ \leftarrow A \} \) has an SLDNF-derivation which satisfies the condition (D1), consisting of \( G_0 = \leftarrow A, \ldots, G_{k_i} \) for some \( k_i \geq 0 \) such that \( G_{k_i} = \leftarrow B_+, J, H = A \). Let \( \varphi_{k_i} \) be a bijection from the multiset \( \{ B_+, J \} \) to the multiset \( \{ B_+, J \} \) such that \( \varphi_{k_i} \) satisfies the condition in (D1). We consider the following two cases depending on whether \( B_+ = B_+ \) (i.e., \( B_+ \) is an old atom) or not.

\[
\begin{align*}
G_0: & \leftarrow A \\
G_1^N: & \leftarrow A \\
\vdots & \\
G_{k_i}: & \leftarrow B_+, \ldots, B_m, A = H
\end{align*}
\]

Fig. 2. \( P_0 \)-simulation (left) of an SLDNF-derivation of \( P_N \cup \{ \leftarrow A \} \) (right).
(i-1) Suppose that $\overline{B}_+ = B_+$. Consider a resolvent of $\leftarrow B_+ \land C_- \in P_{N-1}$, which is denoted by $\leftarrow K, B_- = B_+$. Since the above lemma holds for $P_{N-1}$ from the induction hypothesis, there exists an SLDNF-derivation $Dr_0(B_+)$ of $P_0 \cup \{ \leftarrow B_+ \}$ which satisfies (D1), consisting of $F_0 = \leftarrow B_+, \ldots, F_{k_0} = \leftarrow K, B_- = B_+$ for some $k_0 (\geq 0)$. Let $\varphi_{k_0}$ be a bijection from the multiset \{K\} to the multiset \{K\} such that $\varphi_{k_0}$ satisfies the condition in (D1). Thus, by concatenating the SLDNF-derivation $Dr_0(B_+)$ to $B_+$ in $G_{k_0}$, the SLDNF-derivation of $P_0 \cup \{ \leftarrow A \}$ consisting of $G_0 = \leftarrow L_1, A_1 = B_+, \ldots, G_{k_0} = \leftarrow K, B_+ = B_+$ for some $k_0 (\geq 0)$. Then the proposition follows from a discussion similar to (i-1).

(i-2) Suppose that $\overline{B}_+$ is of the form: "$L_1, A_1 = B_+$" for a variant $A_1 \leftarrow L_1$ of some clause in $P_{new}$. From the induction hypothesis for $P_{N-1}$, there exists an SLDNF-derivation $Dr_0(B_+)$ of $P_0 \cup \{ \leftarrow L_1, A_1 = B_+ \}$ which satisfies (D1), consisting of $F_0 = \leftarrow L_1, A_1 = B_+, \ldots, F_{k_0} = \leftarrow K, B_+ = B_+$ for some $k_0 (\geq 0)$. From the similar discussion to that in (i), there exists an SLDNF-derivation $Dr_0$ of $P_0 \cup \{ \leftarrow L_0, A_0 = A \}$ consisting of $G_0 = \leftarrow L_0, A_0 = A', \ldots, G_{k_1}$ for some $k_1 (\geq 0)$ such that $G_{k_1} = \leftarrow \overline{B}_+, \overline{J}, H = A$. Again, we consider the following two cases depending on whether $\overline{B}_+ = B_+$ (i.e., $\overline{B}_+$ is an old atom) or not.

(ii) Next, suppose that $A$ is a new atom. We show that there exists a clause $C_0: A_0 \leftarrow L_0 \in P_{new}$ and an SLDNF-derivation $Dr_0$ of $P_0 \cup \{ \leftarrow L_0, A_0 = A \}$ satisfying both conditions (D1) and (D2). From the induction hypothesis for $P_{N-1}$, there exists an SLDNF-derivation of $P_0 \cup \{ \leftarrow L_0, A_0 = A \}$ which satisfies the conditions (D1) and (D2), consisting of $G_0 = \leftarrow L_0, A_0 = A', \ldots, G_{k_1}$ for some $k_1 (\geq 0)$ such that $G_{k_1} = \leftarrow \overline{B}_+, \overline{J}, H = A$. Again, we consider the following two cases depending on whether $\overline{B}_+ = B_+$ (i.e., $\overline{B}_+$ is an old atom) or not.

(ii-1) Suppose that $\overline{B}_+ = B_+. \text{Consider a resolvent of } \leftarrow B_+ \land C_- \in P_{N-1}$, which is denoted by $\leftarrow K, B_- = B_+$. Again, from the induction hypothesis, there exists an SLDNF-derivation $Dr_0(B_+)$ of $P_0 \cup \{ \leftarrow B_+ \}$ which satisfies (D1), consisting of $F_0 = \leftarrow B_+, \ldots, F_{k_2} = \leftarrow K, B_- = B_+$ for some $k_2 (\geq 0)$. From the similar discussion to that in (i), there exists an SLDNF-derivation $Dr_0$ of $P_0 \cup \{ \leftarrow L_0, A_0 = A \}$ consisting of $G_0 = \leftarrow L_0, A_0 = A', \ldots, G_{k_1+k_2}$, where $G_{k_1+k_2}$ is of the form: $\leftarrow \overline{K}, \overline{J}, H = A$. The proof that $Dr_0$ satisfies (D1) is quite similar to that of the case (i). We thus only show that $Dr_0$ satisfies (D2). Let $\varphi_{k_2}, \varphi$ be the bijections defined as in (i). If there exists an atom, say $U$, in $L_0$ such that it is left unresolved in $Dr_0$, its possibly instantiated version, say, $\overline{B}_j$ should be contained in $\overline{J}$. Then such an unresolved atom $U$ exists in $Dr_0$ if and only if $B_j = \varphi_{k_1}^{-1}(\overline{B}_j)$ in $J$ is an inherited atom in $C_+$ from the induction hypothesis, which holds if and only if $B_j\theta$ is an inherited atom in $C$ from the definition of an inherited atom. Thus the condition (D2) holds for $Dr_0$. 

\[ G_{k_1+k_2}: \leftarrow \overline{K}, \overline{J}, H = A, \]
(ii-2) Next, suppose that $B_\omega$ is a new atom. As for the condition (D1), the proof is quite similar to that of (i-1). Moreover the condition (D2) is also shown from a discussion similar to (ii-1).

Case 3: $C$ is the result of folding. Let $C_+ \in P_{N-1}$ be the folded clause of the form: $H \leftarrow J_0, K$ and $D \in P_{\text{new}}$ be the folding clause: $B \leftarrow J_0\theta$, where $J_0\theta = J$ for some substitution $\theta$. Let $\theta_{\nu}$ be the restriction of $\theta$ to those variables occurring only in $J_0$. From the condition of folding (F2), $\theta_{\nu}$ is a renaming substitution of the form: $\{X_i / Z_i\} (i \geq 0)$, where $X_i$ is an internal variable in $D$ and $Z_i$ is a variable occurring only in $J$. The result of folding $C$ is $H \leftarrow B\theta, K$. The resolvent $G_1^N$ of $C$ and $G_0^N = \leftarrow A$ can be denoted by

$$G_1^N: \leftarrow B\theta, K, H = A.$$

(i) First, we show that there exists an SLDNF-derivation $D_{r_0}$ of $P_0 \cup \{\leftarrow A\}$ satisfying the condition (D1). Since $B\theta$ is a new atom, what we should prove is that $P_0 \cup \{\leftarrow A\}$ has an SLDNF-derivation $D_{r_0}$, consisting of $G_0 = \leftarrow A, \ldots, G_k$, for some $k_0 (\geq 0)$ such that

$$G_k: \leftarrow J_0, K, B = B\theta, H = A,$$

where $\tilde{D}$ is an arbitrarily chosen and fixed variant of $D$ of the form: $\tilde{B} \leftarrow \tilde{J}_0$ such that none of the variables in $\tilde{D}$ appear elsewhere. Moreover, let $t$ be a renaming substitution from variables in $D$ to those in $\tilde{D}$. Consider an SLDNF-derivation of $P_{N-1} \cup \{\leftarrow A\}$.

(ii) Next, suppose that $A$ is a new atom. We show that there exists a clause $C_0$: $A_0 \leftarrow L_0$ in $P_{\text{new}}$ and an SLDNF-derivation $D_{r'_0}$ of $P_0 \cup \{\leftarrow L_0, A_0 = A\}$ satisfying the both conditions (D1) and (D2). From the induction hypothesis for $P_{N-1}$ and the similar discussion in (i), there exists an SLDNF-derivation $D_{r'_0}$ of $P_0 \cup \{\leftarrow L_0, A_0 = A\}$ which satisfies the conditions (D1) and (D2), consisting of $G_0 = \leftarrow L_0, A_0 = A, \ldots, G_k$, for some $k_1 (\geq 0)$ such that $G_k = \leftarrow J^*, K, H = A$, where $J^*$ is defined in the above (i). The condition (D1) is shown similarly to the case (i). As for the condition (D2), note that an unresolved atom in $D_{r'_0}$ (if any), say $U$, is not contained in $J^*$; otherwise, let $B' \in J^*$ be the possibly instantiated version of $U$. Then, $B'$ is an inherited atom in $C_+',$ thus folding cannot be applied to $C_+',$ nor to $C_-$, which contradicts the assumption. Consequently, the unresolved atom $U$, if
it exists, would be contained in $K$. Again, from the induction hypothesis, it is an inherited atom in $C'$, and so is in $C$ from the definition of the inherited atom, it is also an inherited atom in $C \in P_N$, which proves the condition (D2). □

Using the above lemma, we can now show Lemma 3.3.1.

**Lemma A.3.2** ($P_0$-simulation of SLDNF-derivation in $P_N$). Let $P_0, \ldots, P_N$ be a transformation sequence and let $G$ be a goal. Suppose that there exists an SLDNF-derivation $D_{rN}$ of $P_N \cup \{G\}$, $G_0 = G, \ldots, G_k, \ldots$, using input clauses in $P_N$ and substitutions $\theta_1, \ldots, \theta_k, \ldots$. Then, there exists an SLDNF-derivation $D_{r0}$ of $P_0 \cup \{G\}$, $F_0 = G, \ldots, F_k, \ldots$, using input clauses in $P_0$ and substitutions $\sigma_1, \ldots, \sigma_k, \ldots$, satisfying the following conditions:

(i) For each $k$ ($k \geq 0$), there exists some $l_k$ ($\geq 0$) such that $F_k \sigma_1 \cdots \sigma_k$ is a $P_{new}$-expansion of $G_k \theta_1 \cdots \theta_k$; and

(ii) the restriction of $\sigma_1 \cdots \sigma_k$ to the variables in $G$ is the same as that of $\theta_1 \cdots \theta_k$.

(iii) (fairness) Furthermore, if the SLDNF-derivation $G_0 = G, \ldots, G_k, \ldots$ is fair, then so is the SLDNF-derivation $F_0 = G, \ldots, F_k, \ldots$.

$D_{r0}$ is called a $P_0$-simulation of $D_{rN}$.

**Proof.** The proof is done by induction on the length $k$ of SLDNF-derivation $D_{rN}$. The induction basis (i.e., $k = 0$) is obvious. In the following, we define a bijection $\varphi_k$ from the multiset of literals in $G_k$ to that of literals in $F_k$. For $k = 0$, $G_0 = F_0$ and let $\varphi_0$ be an identity. Suppose that the proposition holds until $k - 1$ ($k > 0$) and $\varphi_{k-1}$ is already defined. Let $D_{rN_{k-1}}$ be the segment of $D_{rN}$ from $G_0$ to $G_{k-1}$, and let $G_{k-1}$ be of the form: $\leftarrow A, \Delta$, where $\Delta$ is a possibly empty sequence of literals and $A$ is the selected literal in $G_{k-1}$. We first show conditions (i) and (ii).

- When $A = \neg A'$ is a negative literal, $A$ should be ground and there exists a finitely failed SLDNF-tree for $P_N \cup \{\neg A\'}$. In this case, $G_k$ is of the form: $\leftarrow \Delta$ and $\theta_k$ is an identity substitution. From the partial correctness w.r.t. FF, $P_0 \cup \{\neg A\}$ also has a finitely failed SLDNF-tree. Thus, it is easy to see that the above (i) and (ii) hold. $\varphi_k$ is $\varphi_{k-1}$ except that the selected atom $A$ is deleted from its domain.

- Otherwise (namely, $A$ is a positive atom), suppose that $A$ is an old atom. Let $C \in P_N$ be an input clause of the form: $H \leftarrow L$, where $L$ is a possibly empty sequence of literals. Then, the resolvent $G_k$ of $G_{k-1}$ and $C$ is of the form: $\leftarrow \Delta, A = H$, where $\theta_k$ is a substitution given by the equation $A = H$. On the other hand, from the induction hypothesis, there exists an SLDNF-derivation $D_{rN_{k-1}}$ corresponding to $D_{rN_{k-1}}$, which satisfies the conditions in the lemma. Suppose that $D_{rN_{k-1}}$ consists of $F_0 = G, \ldots, F_{k-1}$. Note that $F_{k-1}$ can be denoted by $\leftarrow A, \Delta$. Since $A$ is assumed to be an old atom, $A$ is equivalent to $A$. Moreover, due to Lemma A.3.1, there exists an SLDNF-derivation $D_{rA}$ of $P_0 \cup \{\neg A\}$ consisting of a sequence of goals $\leftarrow A, \ldots, \leftarrow L, A = H$, which satisfies the conditions given
therein. Thus, using $Dr_N^0$, it is easy to see that we can extend $Dr_N^{0+1}$ into an SLDNF-derivation $Dr_N^0$ so that it satisfies the conditions (i) and (ii) in the lemma. Let $\varphi_L$ be a bijection from the multiset $\{L\}$ to $\{\tilde{L}\}$. Then, using $\varphi_L$, $\varphi_k$ is obtained by extending $\varphi_k^{k-1}$ in an obvious way.

- The proof of the conditions (i) and (ii) for the case where $A$ is a new atom is done similarly to the above-mentioned case.

Finally, we show the fairness condition (iii). Suppose that $A$ is selected in some goal $G_i$ of $Dr_N$ and let $G_{i+1}$ be the resolvent of $G_i$ and an input clause $C$. Then, in $Dr_N^0$, the corresponding subgoal $\leftarrow \varphi_i(A)$ in $F_i$ is tried, obtaining some descendant node $F_{i+1}$ of $F_i$ which corresponds to $G_{i+1}$. Thus, in order to show the fairness condition, it suffices to consider the case where $A$ is a new atom. Let $\varphi_i(A)$ be of the form: $B_1, \ldots, B_k, H = A$, where $H \leftarrow B_1, \ldots, B_k$ is a variant of a clause in $P_{new}$ whose head is unifiable with $A$. If there exists any literal, say $B_j$, in $F_i$ such that it is left unresolved from $F_i$ to $F_{i+1}$, it follows from Lemma A.3.1 that it should be an inherited atom in $C$. Thus $\varphi^{i+1}_i(B_j)$ is an old atom in $G_{i+1}$.

From the fairness condition of $Dr_N^0$, its possibly instantiated version will be eventually selected, which means that $B_j$ in $Dr_N^0$ will be also eventually selected.

A.4. Proof of total correctness w.r.t. SS

In order to prove the total correctness w.r.t. SS, we need to prepare several definitions and notations. Most of the following definitions are originally given in [15] and [7] for definite programs. We extend them for general programs in a suitable manner. Let $\theta$ be a substitution and $A$ an atom. Then, the restriction of $\theta$ to the variables in $A$ is denoted by $\theta|_A$.

**Definition A.4.1 (weight of derivation).** Let $P_0$ be the initial program of a transformation sequence and $\Gamma$ a sequence of literals. Let $Dr$ be a finite SLDNF-derivation of $P_0 \cup \{\leftarrow \Gamma\}$ consisting of goals $G_0 = \leftarrow \Gamma, G_1, \ldots, G_n$. The weight of $Dr$ is defined as follows:

1. When $Dr$ ends in an empty clause (i.e., $Dr$ is a successful derivation), the weight of $Dr$ is the number of those goals in $Dr$ whose selected literals are either old atoms or negative literals.

2. When $Dr$ ends in a goal $G_n = \leftarrow B_1, \ldots, \neg B_k$ ($k \geq 1$) such that each $B_i$ ($k \geq i \geq 1$) is a nonground atom, the weight of $Dr$ is the number of those goals in $Dr$ whose selected literals are either old literals or negative literals, plus $k$ (i.e., the number of negative literals in $G_n$).

**Definition A.4.2 (weight of atom-substitution pair).** Let $P_0$ be the initial program of a transformation sequence. Let $A$ be an atom and $\sigma$ a substitution. Then, the weight of a pair $(A, \sigma)$, denoted by $w(A, \sigma)$, is defined to be the minimum of the weight of the SLDNF-derivation of $P_0 \cup \{\leftarrow A\}$, consisting of goals $G_0 = \leftarrow A, G_1, \ldots, G_n$ and substitutions $\theta_1, \ldots, \theta_n$ such that
(i) \( G_n \) is either an empty goal or a goal consisting only of negative nonground literals, and
(ii) \( \sigma \) is the restriction of \( \theta_1 \circ \cdots \circ \theta_n \) to the variables in \( A \).

Similarly, let \( \Gamma' \) be a sequence of literals and \( \tau \) a substitution. Then, the above definition is extended to the definition of the weight of a pair \((\Gamma', \tau)\), denoted by \( w(\Gamma', \tau) \), in an obvious way.

The following notion of a descent clause plays an important role in the proof of the total correctness w.r.t. SS.

**Definition A.4.3 (descent clause).** Let \( P_i \) be a program in a transformation sequence starting from an initial program \( P_0 \). Let \( C \) be a clause in \( P_i \) of the form: \( H \leftarrow L \), where \( L \) is a possibly empty sequence of literals. Suppose that \( A \) is an atom such that \((A, \sigma) \in SS(P_0)\) for some substitution \( \sigma \). Then \( C \) is called a descent clause for \((A, \sigma)\), if there exists a substitution \( \tau \) such that

(W1) \((\{L, H = A\}, \tau) \in SS(P_0)\) and the restriction of \( \tau \) to the variables in \( A \) is \( \sigma \),
(W2) \( w(A, \sigma) \geq w((L, H = A), \tau) \), and
(W3) if \( C \) satisfies the folding condition (F4), then \( w(A, \sigma) > w((L, H = A), \tau) \).

**Definition A.4.4 (weight completeness).** Let \( P_i \) be a program in a transformation sequence starting from the initial program \( P_0 \). Then, \( P_i \) is weight complete if and only if, for any atom-substitution pair \((A, \sigma) \in SS(P_0)\), there exists a descent clause in \( P_i \) for that pair.

After showing the following lemma, we proceed by proving the total correctness w.r.t. SS.

**Lemma A.4.1.** Let \( P_0, \ldots, P_N \) be a sequence of program transformation and \( C \) be a clause in \( P_i \) \((0 \leq i \leq N)\). If \( C \) does not satisfy the folding condition (F4) in Definition 2.1.3, then all the atoms in the body of \( C \) are old atoms.

**Proof.** Since \( C \) does not satisfy the condition (F4), the head of \( C \) is a new atom and unfolding has not been applied to \( C \) during the transformation. Thus, \( C \) should be inherited as it is from \( P_0 \). Then, the lemma obviously holds from the definition of an initial program \( P_0 \). \( \square \)

Following [15], the outline of the proof of the total correctness is as follows:
(1) We first show that the weight completeness is a sufficient condition for the total correctness w.r.t. SS (Lemma A.4.2).
(2) Next, the initial program \( P_0 \) of a transformation sequence is shown to be weight complete (Lemma A.4.3).
Finally, the weight completeness is preserved during program transformation (Lemma A.4.4).

**Lemma A.4.2** (weight completeness is sufficient for total correctness w.r.t. SS). Let \( P_0, \ldots, P_N \) be a sequence of program transformation. If \( P_i \) is weight complete, then \( SS(P_i) \equiv SS(P_0) \) (\( N \geq i \geq 0 \)).

**Proof.** We prove a more general proposition that, under the same condition, if \( (I', \sigma) \in SS(P_0) \), then \( (I', \sigma) \in SS(P_i) \), where \( I' \) is a (possibly empty) sequence of literals.

First, we introduce the following well-founded ordering \( > \) into the set of pairs \( (I', \sigma) \in SS(P,) \), i.e., \( (I_1', \sigma_1) > (I_2', \sigma_2) \) if and only if

1. \( w(I_1', \sigma_1) > w(I_2', \sigma_2) \), or
2. \( w(I_1', \sigma_1) = w(I_2', \sigma_2) \) and the number of new atoms in \( I_1' \) is greater than that of new atoms in \( I_2' \).

We show the lemma by induction on the above defined well-founded ordering.

As for the induction basis, i.e., when \( I' \) is empty, the lemma is obvious. Next, suppose that \( I' \) is of the form: \( A, A' \), where \( A \) is a possibly empty sequence of literals and \( A' \) is the selected atom in the initial goal \( G_0 \) of an SLDNF-refutation of \( P_0 \cup \{G_0 = \leftarrow A, \Delta \} \) with the computed answer substitution \( \sigma \). When \( A \) is a negative literal, say, \( \sim A' \), \( A' \) should be ground and there exists a finitely failed SLDNF-tree for \( P_0 \cup \{\sim A' \} \). Thus, \( G_0 \) has the child node \( G_1 = \leftarrow A \) such that \( (A, \sigma) \in SS(P_0) \).

On the other hand, from the total correctness w.r.t. FF (Proposition 3.3.1), \( P_1 \cup \{\leftarrow A' \} \) also has a finitely failed SLDNF-tree. Thus, also in an SLDNF-derivation of \( P_0 \cup \{\leftarrow A, \Delta \} \), when \( A \) is selected in the initial node \( G_0 \), \( G_0 \) has the child node \( G_1 \). From the induction hypothesis on the well-founded ordering \( > \), \( (\Delta, \sigma) \) is in \( SS(P_1) \). Thus, so is \( (I', \sigma) \). Otherwise, suppose that \( A \) is a positive literal. From the definition that \( (I', \sigma) = ((A, \Delta), \sigma) \) in \( SS(P_0) \), there exists an SLDNF-derivation of \( P_0 \cup \{\leftarrow A, \Delta \} \) with a computed answer substitution \( \sigma \). Thus, it follows that its subgoal \( P_0 \cup \{\leftarrow \Delta \} \) has an SLDNF-derivation which satisfies the following conditions:

1. **(D1)** It ends in a goal \( G_n = \leftarrow \mathcal{N}, \) where \( \mathcal{N} \) is a possibly empty sequence of negative non-ground literals.
2. **(D2)** Let \( \sigma_\Delta \) be a substitution for variables in \( \leftarrow \Delta \), computed during this derivation. Then, there exists a substitution \( \sigma_1 \) such that \( (A \sigma_\Delta, \sigma_1) \in SS(P_0) \) and \( \sigma_\Delta \circ \sigma_1 = \sigma \).
3. **(D3)** Moreover, \( \mathcal{N} \sigma_1 \) is ground and \( \leftarrow \mathcal{N} \sigma_1 \) has a successful SLDNF-derivation.

In the following, we consider such an SLDNF-derivation of \( P_0 \cup \{\leftarrow \Delta \} \) that it satisfies **(D1)**–**(D3)** and the weight \( w(\Delta, \sigma_\Delta) \) is the minimum.

As \( P_i \) is weight complete and \( (A \sigma_\Delta, \sigma_1) \in SS(P_i) \), there exists a descent clause \( C \) for \( (A \sigma_\Delta, \sigma_1) \) in \( P_i \). Let \( C \) be of the form: \( H \leftarrow L \), where no variables in \( C \) appear elsewhere. Then, from the definition of the descent clause, the following conditions hold:
(W1) \(((L, H = A\sigma_\Delta), \tau) \in SS(P_0)\) and the restriction of \(\tau\) to the variables in \(A\sigma_\Delta\) is \(\sigma_1\).

(W2) \(w(A\sigma_\Delta, \sigma_1) \geq w((L, H = A\sigma_\Delta), \tau)\), and

(W3) if \(C\) satisfies the folding condition (F4), then \(w(A\sigma_\Delta, \sigma_1) > w((L, H = A\sigma_\Delta), \tau)\).

Now, consider an SLDNF-derivation \(D_r\) of \(P_0 \cup \{\leftarrow A, \Delta\}\). Then, using \(C\) as an input clause, the initial goal \(G_0 = \leftarrow A, \Delta\) has the successor \(G_1 = \leftarrow L, \Delta, A = H\). We first show that there exists an SLDNF-refutation \(D_{r_0}\) of \(P_0 \cup \{G_i\}\) with the computed answer substitution whose restriction to the variables in \(G_i\) is \(\sigma\). Resolving the subgoal \(\Delta\) in \(G_i\) first, from (D1) in the above, \(G_i\) has a descendent \(G_{i_1}\) of the form: \(\leftarrow L, N, H = A\sigma_\Delta\) with the substitution \(\sigma_\Delta\) computed from \(G_i\) to \(G_{i_1}\).

From the above condition (W1), \(P_0 \cup \{\leftarrow L, H = A\sigma_\Delta\}\) has a successful SLDNF-derivation with the computed answer substitution \(\tau\). Thus, \(G_{i_1}\) has a descendent \(G_{i_1}\) of the form: \(\leftarrow L, N, H = A\sigma_\Delta\) with the substitution \(\sigma_\Delta\) computed from \(G_i\) to \(G_{i_1}\).

When \(\tau\) is defined, \(w((A, A), \sigma) = w((L, H = A\sigma_\Delta), \sigma_1) + w(\Delta, \sigma_\Delta)\)

\[\geq w((L, H = A\sigma_\Delta), \tau) + w(\Delta, \sigma_\Delta) \quad \text{(from W2)}\]

\[= w((L, H = A), \sigma_\Delta \circ \tau).\]

When \(w((A, A), \sigma) > w((L, H = A), \sigma_\Delta \circ \tau)\) holds, from the induction hypothesis on the well-founded ordering \(\geq\), \(P_0 \cup \{G_i\}\) has a successful SLDNF-derivation with a computed answer \(\sigma_\Delta \circ \tau\). Thus \(P_0 \cup \{G_0\}\) also has a successful SLDNF-derivation with a computed \(\sigma_\Delta \circ \tau\) whose restriction to the variables in \(G_0\) is \(\sigma\). So, we have that \(((A, \Delta), \sigma) = (I, \sigma) \in SS(P_i).\)

On the other hand, when \(w((A, A), \sigma) = w((L, H = A), \sigma_\Delta \circ \tau)\) holds, from the condition (W3), \(C\) does not satisfy the folding condition (F4). From Lemma A.4.1,

\[G_0 : \leftarrow A, L \quad | \quad C : H \leftarrow L\]

\[G_i : \leftarrow L, \Delta, A = H \quad | \quad G_{i_1} : \leftarrow L, \Delta, A = H\]

\[\vdots\]

\[\vdots\]

\[\vdots\]

\[G_i : \leftarrow L, N, H = A\sigma_\Delta \quad | \quad \vdots\]

\[G_{i_1} : \leftarrow N, \quad | \quad \vdots\]

\[G_{i_2} : \leftarrow \square\]

Fig. 3. An SLDNF-derivation of \(P_0 \cup \{G_i\}\) (left) and an SLDNF-derivation of \(P_1 \cup \{G_0\}\) (right).
the predicate of $H$ is a new predicate and so is that of $A$, while all atoms in $L$ are old atoms. Thus, it follows that $((A, \Delta), \sigma) > ((L, H = A, \Delta), \sigma_\Delta \circ \tau)$. Consequently, from the induction hypothesis, $P_i \cup \{\leftarrow L, \Delta, H = A\}$ has a successful SLDNF-derivation with a computed answer $\sigma_\Delta \circ \tau$. From the same discussion as above, it is shown that $((A, \Delta), \sigma) = (I, \sigma) \in SS(P_i)$. 

**Lemma A.4.3.** The initial program $P_0$ is weight complete.

**Proof.** Let $(A, \sigma)$ be an atom substitution pair in $SS(P_0)$ and let $Dr$ be an SLDNF-refutation of $P_0 \cup \{G_0 = \leftarrow A\}$ with answer substitution $\sigma$ such that the weight of $Dr$ is $w(A, \sigma)$. Furthermore, let $C = H \leftarrow L$ be the input clause used in $G_0$. Then, $G_0$ has the child node $G_1 = \leftarrow L, H = A$ and there exists an SLDNF-refutation of $P_0 \cup \{G_1\}$ with answer substitution $\tau$ such that the restriction of $\tau$ to the variables in $A$ is $\sigma$. It is easy to see that $C$ satisfies conditions (W1) and (W2) of the definition of a descent clause. Moreover, $C$ satisfies the folding condition (F4) if and only if $H$ is an old atom. Thus, it follows that $w(A, \sigma) > w((L, H = A), \tau)$, so $C$ satisfies also the condition (W3). 

**Lemma A.4.4** (preservation of weight completeness). Let $P_0, \ldots, P_n$ be a sequence of program transformation. If $P_i$ is weight complete, then $P_{i+1}$ is also weight complete $(N - 1 \equiv t \equiv 0)$.

**Proof.** Let $(A, \sigma)$ be an atom-answer substitution pair in $SS(P_0)$. Since $P_i$ is weight complete, there exists a descent clause $C_0$ for $(A, \sigma)$ in $P_i$. We will show that there exists a descent clause for $(A, \sigma)$ also in $P_{i+1}$, by considering the following three cases.

**Case 1:** $C_0$ is in $P_{i+1}$. Then $C_0$ itself is a descent clause for $(A, \sigma)$ also in $P_{i+1}$.

**Case 2:** $C_0$ is unfolded. Let $C_0$ be $H \leftarrow B_\circ$, $J$, where $J$ is a possibly empty sequence of literals and suppose that atom $B_\circ$ is unfolded. Since $C_0$ is a descent clause for $(A, \sigma)$, there exists an SLDNF-refutation of $P_0 \cup \{\leftarrow B_\circ, J, H = A\}$ with a computed answer substitution $\tau$ such that the restriction of $\tau$ to the variables in $A$ is $\sigma$. Thus, it follows that its subgoal $P_0 \cup \{\leftarrow J, H = A\}$ has an SLDNF-derivation which satisfies the following conditions:

(D1) It ends in a goal of the form: $\leftarrow \mathcal{N}$, where $\mathcal{N}$ is a (possibly empty) sequence consisting only of negative literals. Let $\tau_1$ be a substitution computed during this SLDNF-derivation.

Then, it is easy to see that $P_0 \cup \{\leftarrow B_\circ, \tau_1\}$ has an SLDNF-refutation with a computed answer substitution $\tau_2$ such that

(D2) $\mathcal{N} \tau_2$ is ground and $P_0 \cup \{\leftarrow \mathcal{N} \tau_2\}$ has an SLDNF-refutation, and

(D3) $\tau_1 \circ \tau_2 = \tau$.

Again, from the weight-completeness of $P_i$, there exists a descent clause $C_\circ$ in $P_i$ for $(B_\circ, \tau_1, \tau_2)$. Let $C_\circ = B_\circ \leftarrow K$. Then,

(D4) $P_0 \cup \{\leftarrow K, B_\circ = B_\circ, \tau_1\}$ has an SLDNF-refutation with a computed answer substitution $\eta$ such that the restriction of $\eta$ to the variables in $B_\circ, \tau_1$ is $\tau_2$. 

Since $B_-$ and $B_+, \tau_1$ are unifiable, so are $B_+$ and $B_-$. Thus, $C_0$ is unfolded by $C_-$, obtaining an unfolded clause $C$ in $P_{\tau_1}$ of the form: $H\beta \leftarrow KB, J\beta$, where $\beta$ is an mgu of $B_+$ and $B_-$. Now, we show that $C$ is a descent clause for $(A, \sigma)$.

**Condition (W1):** Consider an SLDNF-derivation $Dr$ of $P_0 \cup \{G_0 = \leftarrow KB, J\beta, H\beta = A\}$. Due to the unification theorem, an SLDNF-derivation $Dr'$ of $P_0 \cup \{G_0' = \leftarrow K, B_+ = B_+, J, H = A\}$ is equivalent to $Dr$, as far as a computed answer substitution restricted to the variables in $A$ is concerned. From (D1), $G_0'$ has a descendent node $G_1': \leftarrow K, B_+ = B_+ \tau_1, \eta$. From (D4), $G_1'$ has a descendent node $G_2': \leftarrow \eta \gamma$. Finally, from (D2) and from the fact that the restriction of $\eta$ to the variables in $B_+ \tau_1$ is $\tau_2$, $Dr'$ has a successful SLDNF-derivation with a computed answer substitution $\tau_1 \circ \eta$. Moreover,

$$\left((\tau_1 \circ \eta)|_A = \tau_1|_A \circ \eta \right) \land \left((\tau_1 \circ \eta)|_A = \tau \right).$$

**Condition (W2):** Since $C_0$ (resp. $C_-$) is a descent clauses for $(A, \sigma)$ (resp. $(B_+, \tau_1, \tau_2)$), we have

$$w(A, \sigma) \geq w((B_+, J, H = A), \tau),$$

$$w(B_+ \tau_1, \tau_2) \geq w((K, B_+ = B_+ \tau_1), \eta).$$

Thus,

$$w(A, \sigma) \geq w((B_+, J, H = A), \tau)$$

$$= w((B_+ \tau_1, \tau_2) + w((J, H = A), \tau_1)$$

$$\geq w((K, B_+ = B_+ \tau_1), \eta) + w((J, H = A), \tau_1)$$

$$= w((K, B_+ = B_+ \tau_1, J, H = A), \tau_1 \circ \eta)$$

$$= w((KB, J\beta, H\beta = A), \tau_1 \circ \eta).$$

**Condition (W3):** Note that $C$ satisfies the condition (F4). Thus, we have to show that

$$w(A, \sigma) > w((KB, J\beta, H\beta = A), \tau_1 \circ \eta).$$

When $B_+$ is a new atom, $C$ satisfies (F4). Thus, the strict inequality in (1) holds. Otherwise (i.e., when $B_+$ is an old atom), $C_-$ satisfies (F4), thus the strict inequality in (2) holds. Consequently, in either case, it is shown that the strict inequality holds in (3). Thus, it is shown that $C$ is a descent clause for $(A, \sigma)$.

**Case 3:** $C_0$ is folded. Let $C_0$ be $H \leftarrow J_+, K$ and $D \in P_{\text{new}}$ be the folding clause of the form $B \leftarrow J_-$, where $J \theta = J_+$ for some substitution $\theta$. Then, the result of folding $C \in P_{\tau_1}$ is $H \leftarrow B\theta, K$. Since $C_0$ is a descent clause for $(A, \sigma)$, there exists an SLDNF-refutation $P_0 \cup \{G_0 = \leftarrow J_+, K, H = A\}$. Let $\tau$ be its computed answer substitution such that the restriction of $\tau$ to the variables in $A$ is $\sigma$. We show that this $C$ is a descent clause for $(A, \sigma)$.

**Condition (W1):** Consider an SLDNF-derivation $Dr$ of $P_0 \cup \{G_0 = \leftarrow B\theta, K, H = A\}$. Let $D'$ be a variant of $D$, say, $D' = B' \leftarrow J'$ such that all variables in $D'$ are newly introduced ones. Then $G_0'$ has a successor $G_1'$ of the form: $\leftarrow J' \gamma, K,$
$H = A$, where $\beta$ is an mgu of $B'$ and $B\theta$ such that $B'\beta = B\theta$. From Lemma A.1.2, $G_1$ is a variant of $G_0$ such that they are different only with respect to the internal variables in $J'$. Thus $P_0 \cup \{G_0\}$ has an SLDNF-refutation with a computed answer substitution $\tau'$ such that the restriction of $\tau'$ to the variables in $A$ is $\alpha$.

**Condition (W2) and (W3):** Note that $C_0$ is a descent clause for $(A, \alpha)$ and it satisfies the condition (F4). Thus, we have

$$w(A, \alpha) > w((J', K, H = A), \tau) = w((J', \beta, K, H = A), \tau').$$

Since $B$ is a new atom, it is shown from the definition of the weight that

$$w(A, \alpha) > w((B\theta, K, H = A), \tau').$$

\[ \square \]

### A.5. Proof of preservation of perfect model semantics

In this section, we give the proof of Proposition 4.1. The important property of the perfect model semantics we will use in the following proof is that every perfect model is supported [12]. That is, we first fix a pre-interpretation (e.g., [9]) $J$ of a program $P$. Let $M$ be the perfect model of $P$ based on $J$. Then, for every $J$-ground atom $A$ in $M$, there exists a $J$-ground instance of a clause in $P$ such that its head is equal to $A$, all positive premises belong to $M$ and none of the negative premises belong to $M$.

From this property, we can consider a “ground proof derivation” analogous to an SLDNF-derivation. Namely, let $\Gamma$ be a (possibly empty) sequence of ground literals such that $M \models \Gamma$. Then, a ground proof derivation of $P \cup \{G_0 = \neg \Gamma\}$ consists of a sequence of goals: $G_0, G_1, \ldots, G_n = \Box$ (an empty goal), a sequence $C_1, \ldots, C_{n-1}$ of $J$-ground instances of clauses (called input clauses) in $P$ or negative ground literals, satisfying the following conditions:

(i) Let $G_i$ be of the form: $\leftarrow A_1, \ldots, A_m, \ldots, A_n$, where $A_m$ is the selected positive literal in $G_i$. Let $C_{i+1} \in P$ be an input clause of the form: $A_m \leftarrow L$, where $L$ is a possibly empty sequence of ground literals and $M \models L$. Then $G_{i+1}$ is $\leftarrow A_1, \ldots, A_{m-1}, L, A_{m+1}, \ldots, A_n$.

(ii) $G_i$ is $\leftarrow A_1, \ldots, A_m, \ldots, A_n$, where $A_m = \neg A'$ is the selected negative literal in $G_i$ such that $M \models \neg A'$. Then $G_{i+1}$ is $\leftarrow A_1, \ldots, A_{m-1}, A_{m+1}, \ldots, A_n$.

The purpose of defining the above ground proof derivation is to prove Proposition 4.1 by following exactly the same lines as in the proofs of Proposition 3.2.1 and Proposition 3.3.2.

In the following, for a fixed pre-interpretation $J$, we denote the perfect model of $P$ based on $J$ by $\text{PERF}(P)$. Moreover, as we did in Section A.2, we consider $\text{PERF}(P)$ as a set of all (possibly empty) sequences of ground literals $\Gamma$ such that $\text{PERF}(P) \models \Gamma$, or equivalently, $P \cup \{\neg \Gamma\}$ has a ground proof derivation. As we did in Section A.4, we also need the definitions of weights and descent clauses modified suitably for the current purpose.

**Definition A.5.1 (weight of a ground proof derivation).** Let $P_0$ be the initial program of a transformation sequence and $\Gamma$ a sequence of ground literals. Let $Dr$ be a ground
proof derivation of $P_0 \cup \{\leftarrow \Gamma\}$ consisting of goals $G_0 = \leftarrow \Gamma$, $G_1, \ldots, G_n$. The weight of $Dr$ is defined to be the number of those goals in $Dr$ whose selected literals are either old atoms or negative literals.

**Definition A.5.2 (weight of atom).** Let $P_0$ be the initial program of a transformation sequence and $A$ a ground atom such that $\text{PERF}(P_0) \models A$. Then the weight of $A$, denoted by $w(A)$, is defined to be the minimum of the weight of the ground proof derivation of $P_0 \cup \{\leftarrow A\}$. Similarly, let $\Gamma$ be a sequence of ground literals such that $\text{PERF}(P_0) \models \Gamma$. Then, the above definition is extended to the definition of the weight of $\Gamma$, denoted by $w(\Gamma)$, in an obvious way.

**Definition A.5.3 (descent clause).** Let $P_i$ be a program in a transformation sequence starting from an initial program $P_0$. Let $C$ be a ground instance of a clause in $P_i$ of the form: $A \leftarrow L$, where $L$ is a possibly empty sequence of ground literals. Suppose that $A$ is an atom such that $\text{PERF}(P_i) \models A$ holds. Then $C$ is called a descent clause for $A$ if the following conditions are satisfied:

1. $\text{PERF}(P_i) \models L$ holds,
2. $w(A) \geq w(L)$, and
3. if $C$ satisfies the folding condition (F4), then $w(A) > w(L)$.

**Definition A.5.4 (weight completeness).** Let $P_i$ be a program in a transformation sequence starting from the initial program $P_0$. Then, $P_i$ is weight complete if and only if, for any ground atom $A \in \text{PERF}(P_0)$, there exists a descent clause in $P_i$ for $A$.

**Proposition A.5.1 (preservation of perfect model semantics).** Let $P_0, \ldots, P_N$ be a transformation sequence. Then,

1. **(PC):** If $\text{PERF}(P_i) = \text{PERF}(P_0)$, then $\text{PERF}(P_{i+1}) \subseteq \text{PERF}(P_i)$ for $i = 0, \ldots, N-1$.

2. **(TC):** If $\text{PERF}(P_i) = \text{PERF}(P_0)$, then $\text{PERF}(P_{i+1}) \subseteq \text{PERF}(P_i)$ for $i = 0, \ldots, N-1$.

**Proof.** The proof is by mutual induction on $s = \text{stratum}(G_0)$ of goal $G_0 = \leftarrow \Gamma$. It is obvious when $s = 0$. Suppose that the proposition has been proved for all goals $G'_0$ whose $\text{stratum}(G'_0) \leq s$, where $s \geq 0$. We first prove (PC).

Suppose there exists a ground proof derivation $Dr_{i+1}$ of $P_{i+1} \cup \{G_0\}$. The proof is by induction on the length of the ground proof derivation of $P_{i+1} \cup \{G_0 = \leftarrow \Gamma\}$. Let $\Gamma$ be of the form: $A, \Delta$, where $A$ is a ground literal and $\Delta$ is a (possibly empty) sequence of ground literals. Suppose that $\text{stratum}(G_0)$ is $s + 1$. Suppose further that $A$ is the selected atom in $G_0$. When $A = \neg A'$ is a negative literal, $\text{PERF}(P_{i+1}) \models \neg A'$
holds and $G_0$ has the successor $G_1 = \Delta$. Since $\text{stratum}(\Delta)$ is less than $\text{stratum}(G_0)$, $\text{PERF}(P_i) \models \lnot A'$ from the induction hypothesis on the total correctness (TC). Let $D_r$ be a ground proof derivation of $P_i \cup \{G_0\}$. Then $G_0$ has the successor $G_1$ also in $D_r$. From the induction hypothesis of the length of a ground proof derivation, $\text{PERF}(P_i) \models \Delta$, thus $\text{PERF}(P_i) \models \Gamma$.

Next, suppose that $A$ is a positive atom. Then, the proof is quite similar to that of Proposition 3.2.1 (see Section A.2), except that we should consider a ground proof derivation instead of an SLDNF-refutation. So we omit the proof.

Proof of (TC): The proof of the total correctness (TC) is done quite similarly to that of Proposition 3.3.2. First, note that Lemma A.4.1 and Lemma A.4.3 hold also in this case. Thus, what we should prove are those lemmas corresponding to Lemma A.4.2 and Lemma A.4.4. Since the proofs of both lemmas are shown as in their counterparts in the previous section, we only show in the following the proof of the counterpart of Lemma A.4.2.

Lemma A.5.1 (weight completeness is sufficient for total correctness w.r.t. the perfect model semantics). Let $P_0, \ldots, P_N$ be a sequence of program transformation. If $P_{i+1}$ is weight complete, then $\text{PERF}(P_{i+1}) \supseteq \text{PERF}(P_0)$ ($N - 1 \geq i \geq 0$).

Proof. First, we introduce the following well-founded ordering $\succ$ into the set of a (possibly empty) sequence of ground literals $\Gamma$ in $\text{PERF}(P_0)$, i.e., $\Gamma_1 \succ \Gamma_2$ if and only if

1. $w(\Gamma_1) > w(\Gamma_2)$, or
2. $w(\Gamma_1) = w(\Gamma_2)$ and the number of new atoms in $\Gamma_1$ is greater than that of new atoms in $\Gamma_2$.

We show the lemma by induction on the above defined well-founded ordering.

As for the induction basis, i.e., when $\Gamma$ is empty, the lemma is obvious. Next, suppose that $\Gamma$ is of the form: $A, \Delta$, where $\Delta$ is a possibly empty sequence of ground literals and $A$ is the selected literal in $G_0$ of a ground proof derivation of $P_0 \cup \{G_0 = \Delta, A\}$. When $A$ is a negative literal, say, $\lnot A'$, $\text{PERF}(P_0) \models \lnot A'$ holds. Thus $G_0$ has the child node $G_1 = \Delta$ such that $\Delta \in \text{PERF}(P_0)$. On the other hand, from the partial correctness (PC), $\text{PERF}(P_{i+1}) \models \lnot A'$ holds. Thus, also in a ground proof derivation of $P_{i+1} \cup \{G_0 = \Delta, A\}$, when $A$ is selected in $G_0$, $G_0$ has the child node $G_1$. From the induction hypothesis on the well-founded ordering $\succ$, $\Delta$ is in $\text{PERF}(P_{i+1})$. Thus so is $\Gamma$. Otherwise, suppose that $A$ is a positive ground literal.

As $P_{i+1}$ is weight complete and $A \in \text{PERF}(P_0)$, there exists a descent clause $C$ for $A$ in $P_{i+1}$. Let $C$ be of the form: $A \leftarrow L$. Then, from the definition of the descent clause, the following conditions hold:

1. $\text{PERF}(P_0) \models L$,
2. $w(A) > w(L)$, and
3. if $C$ satisfies the folding condition (F4), then $w(A) > w(L)$.
Now, consider a ground proof derivation \( Dr \) of \( P_{i+1} \cup \{ \rightarrow A, \Delta \} \). Then, using \( C \) as an input clause, the initial goal \( G_0 = \rightarrow A, \Delta \) has the successor \( G_1 = \rightarrow L, \Delta \).

If \( w(A) > w(L) \) holds, then \( (A, \Delta) > (L, \Delta) \). Thus, from the induction hypothesis on the well-founded ordering \( > \), we have that \( \text{PERF}(P_{i+1}) \models A, \Delta \).

On the other hand, when \( w(A) = w(L) \) holds, from the condition (W3), \( C \) does not satisfy the folding condition (F4). From Lemma A.4.1, \( A \) is a new atom, while all atoms in \( L \) are old atoms. Thus it follows that \( (A, \Delta) > (L, \Delta) \). Consequently, from the induction hypothesis, it is shown that \( \text{PERF}(P_{i+1}) \models A, \Delta \). \( \square \)

Acknowledgment

This work is based on the result by Tamaki and Sato, and the succeeding work by Kawamura and Kanamori. The author would like to express deep gratitude to them for their stimulating work. The idea of modified folding arose from discussions with Kazunori Ueda and Tadashi Kanamori. The author would like to thank anonymous referees for their useful comments.

References


