ELSEVIER

Theoretical Computer Science 266 (2001) 489–512

Theoretical Computer Science

www.elsevier.com/locate/tcs

A sound and complete procedure for a general logic program in non-floundering derivations with respect to the 3-valued stable model semantics

Susumu Yamasaki *, Yoshinori Kurose

Department of Computing and Information Technology, Faculty of Engineering, Okayama University, 1-1, Tsushima-Naka, 3-chome, Okayama 700, Japan

Received September 1998; revised March 2000; accepted May 2000
Communicated by M. Nivat

Abstract

This paper presents a sound and complete procedure with respect to the 3-valued stable model semantics. The procedure is regarded as an extension of Eshghi and Kowalski abductive proof procedure, involving finite or countably infinite SLD resolution, as well as finite or countably infinite negative recursion caused by negation as failure. The procedure makes use of the set of negative literals (the set of abducibles) for the negation as failure to be implemented. The set of abducibles is not only applicable to the extraction of explanations for abduction as in Kakas et al. (J. Logic Comput. 2 (1992) 719–770) but also specified for what stable model is now computed in the procedure, because a 3-valued stable model is not always the least (that is, the well-founded model). The procedure also contains nondeterminism in the choices of what ground negative literals are used for negation as failure. By the assumptions of some adequate choices, the procedure is well-defined inductively.

Keywords: General logic program; 3-valued stable model; Extension of Eshghi and Kowalski procedure

1. Introduction

This paper proposes an extended version of Eshghi and Kowalski abductive proof procedure [10], with respect to the 3-valued stable model [22]. There are three major backgrounds:

* Corresponding author. Tel.: 81-86-251-8179; fax: 81-86-251-8256.
E-mail address: yamasaki@momo.it.okayam-u.ac.jp (S. Yamasaki).

0304-3975/01/$ - see front matter
PII: S0304-3975(00)00201-2

(1) The relation between SLDNF resolution and the well-founded model [12] is well established, by introducing infinite failure [21, 23]. Because SLDNF resolution makes use of the negation as failure:

$\leftarrow\sim A$ succeeds if $\leftarrow A$ fails, and

$\leftarrow\sim A$ fails if $\leftarrow A$ succeeds,

it may involve infinite negative recursions. It may also contain infinite positive recursion caused by resolution. To cope with infinity, there are refined techniques such as the negative context to avoid infinite negative loop, the resolution to avoid both positive and negative loops, the treatment of function-free programs, and so on [2, 3, 5]. Even for the generalized class of programs, similar methods are established [4]. Note that the well-founded model is the least 3-valued stable model. When we deal with a 3-valued stable model (which is not always the least), we have a question of what procedure is expected to be sound and complete with respect to the 3-valued stable model.

(2) Eshghi and Kowalski abductive proof procedure (E–K procedure, for short) is invented, based on SLDNF resolution. The procedure is essentially significant in the construction of abduction framework as in [16]. For a given general logic program as a theory with a query, some set of negative literals (that is, some set of abducibles) is extracted such that the theory should satisfy the given constraint with the abducibles. The set of abducibles is provided as the set by which negation as failure can be executed. The 2-valued stable model [13] was firstly a candidate for the abductive procedure to be sound with. However, E–K procedure is not always sound with respect to the 2-valued stable model, nor to the generalized stable model [16]. It is essentially concerned with the 3-valued logic and sound with respect to the preferred extension (Dung's semantics) [6, 8]. Through the soundness proof by [8] and/or the relations among model theories as in [17, 27], we see that E–K procedure is sound with respect to the 3-valued stable model. In [14] a finite-failure stable model is presented with respect to which a modified version of E–K procedure is not only sound but also complete. Pereira et al. [20] presents a top-down derivation procedure for the 3-valued stable model semantics, equipped with loop checking techniques, where a grounded program is dealt with. It is a complete procedure for the case that the procedure terminates. However, the ways for the infinite positive and negative recursions are implicit.

(3) Kakas and Toni [18] proposes an abstract computational framework where various argumentation semantics can be computed via different parametric abstract proof theories. The abstract proof theory is required to be a concrete top-down proof procedure for query evaluation. Our procedure can be coherent with the concretization of the abstract procedure with respect to the 3-valued stable model. When we take the 3-valued stable model from denotational semantics viewpoint rather than from the argumentation theory, we can have a concrete procedure to compute the model. We prefer to a concrete, complete procedure, with respect to the 3-valued stable model defined as a denotation.

Based on the backgrounds, we construct a sound and complete procedure except the floundering case, in which countable infinity may be caused by resolutions as well as by negative recursions (applications of negation as failure). It consists of two derivations, a succeeding derivation and a failing derivation. It is an extension of E–K procedure in two aspects:

(i) The extended procedure can realize the effect of the alternating operator [1, 11].
(ii) A failing derivation is recursively defined on the basis of a finite or an infinite sequence of goal sets, where the sequence is enumerated with some choices in selections of literals and in selections of resolution derivations for applications of negation as failure. A succeeding derivation is a finite sequence of goals, where failing derivations are recursively called for.

The paper is organized as follows. In Section 2, terminologies about the model theories of general logic programs are re-organized. Section 3 presents a new procedure, which is expected to be sound and complete in non-floundering derivations with respect to the 3-valued stable model. Section 4 is concerned with the soundness of the procedure. In Section 5, we have the completeness of the procedure. In Section 6, we give remarks on the presented procedure in the floundering cases and on the implementation techniques concerning the procedure.

2. Preliminaries

In this section, fundamental definitions are introduced as well as model theories of general logic programs in 3-valued logic approach, including the alternating fixpoint [11] and 3-valued stable model [22].

2.1. General logic programs

A general logic program is a set of rules (clauses) of the form $A_0 \leftarrow A_1 \dots A_m \sim A_{m+1} \dots \sim A_n$ $(n \geqslant m \geqslant 0)$, where $A_0, A_1, \dots, A_m$ are atoms (positive literals) and $\sim A_{m+1}, \dots, \sim A_n$ are negations of atoms (negative literals). A literal is a positive literal or a negative literal. The rule is also expressed as $A_0 \leftarrow L_1 \dots L_n$, where $L_1, \dots, L_n$ are literals. The rule containing no negative literals is said to be a definite clause, while the set of definite clauses is called a definite program. In the rule $A_0 \leftarrow A_1 \dots A_m \sim A_{m+1} \dots \sim A_n$, A_0 is the head and $A_1 \dots A_m \sim A_{m+1} \dots \sim A_n$ the body. The goal (that is, the query) is an expression of the form $\leftarrow L_1 \dots L_n$, where $L_1, \dots, L_n$ are literals. The goal containing only negative literals is said to be a negative goal. The empty clause containing no head nor body is denoted by $\square$. The empty clause is regarded as a negative goal.

A substitution (see [9] for detailed algebraic properties) is a function from the set of variables *Var* to the set of terms *Term*. A substitution φ may be expressed as $\{x_1 \mid t_1, \dots, x_n \mid t_n\}$, where $t_i = \varphi(x_i)$, $1 \leqslant i \leqslant n$, and $\varphi(x) = x$ if $x \neq x_i$ $(1 \leqslant i \leqslant n)$. We assume that $\{y \mid \varphi(y)$ is defined such that $\varphi(y) \neq y\}$ is finite. The empty substitution

is denoted by ε. That is, $\varepsilon(x)=x$ for any $x\in Var$. An application of a substitution θ to an expression E (say, a literal, a term, a rule or a goal), $E\theta$, denotes the expression obtained by substituting all the variables in E for terms according to θ. $E\theta$ is said to be an instance of E. If $E\theta$ contains no variables, $E\theta$ is called a ground instance.

The composition of substitutions θ and φ (denoted by $\theta\varphi$) is defined by letting $(\theta\varphi)(x)=\theta(x)\varphi$ for $x\in Var$. It is easy to see $(\varphi\psi)\theta=\varphi(\psi\theta)$ for substitutions φ,ψ,θ. Note $\varepsilon\theta=\theta\varepsilon=\theta$. Also we see $(E\theta)\varphi=E(\theta\varphi)$ for an expression E and substitutions θ,φ.

For a substitution φ and an expression E, we define a restriction of φ with respect to E, $\varphi|_E$, to be

$$\varphi|_E(x)=\begin{cases}\varphi(x) & \text{if } x \text{ occurs in } E,\\ x & \text{otherwise}\end{cases}$$

for $x\in Var$.

An expression containing no variables is said to be a ground expression. P_{gr} stands for the set of all ground instance rules obtained from a rule of P. For a general logic program P, the Herbrand base B_P is the set of ground atoms constructed by the function and predicate symbols occurring in P. To refer to the set of negative literals constructed by function and predicate symbols occurring in P, we define

$$B_P^{\sim}=\{\sim a \mid a\in B_P\}.$$

We also define $K^+=\{a \mid \sim a\in K\}$ for $K\subseteq B_P^{\sim}$.

2.2. 3-Valued Herbrand models

We take the 3-valued logic in which t (true), f (false) and u (undefined) are truth values, and the following truth value table is defined. t, f and u are also used to denote atoms whose values are t, f and u, respectively.

Definition 2.1. Given a general logic program P, a 3-valued Herbrand interpretation of P is $(IT,IF)\in 2^{B_P}\times 2^{B_P}$ such that $IT\cap IF=\emptyset$.

Note that a 3-valued Herbrand interpretation of P, (IT,IF), is a 2-valued Herbrand interpretation if $IT\cup IF=B_P$.

For a 3-valued Herbrand interpretation $I=(IT,IF)$, the truth value of a ground expression is defined recursively as follows:

(1) A ground atom a is true in I if $a\in IT$, and false in I if $a\in IF$. Otherwise its truth value is undefined.
(2) A ground literal $\sim a$ is true in I if $a\in IF$, and false in I if $a\in IT$. Otherwise its truth value is undefined.
(3) A ground body $a_1\ldots a_m\sim a_{m+1}\ldots\sim a_n$ of a rule is true in I if $a_1,\ldots,a_m\in IT$ and $a_{m+1},\ldots,a_n\in IF$, and false in I if $a_i\in IF$ for some a_i $(1\leqslant i\leqslant m)$ or $a_j\in IT$ for some a_j $(m+1\leqslant j\leqslant n)$. Otherwise its truth value is undefined.

Table 2.1
Truth value table

$\wedge$	t	f	u
t	t	f	u
f	f	f	f
u	u	f	u

$\sim$	
t	f
f	t
u	u

$\leftarrow$	t	f	u
t	t	t	t
f	f	t	f
u	f	t	t

$\leftrightarrow$	t	f	u
t	t	f	f
f	f	t	f
u	f	f	t

(4) The value of the ground rule $a \leftarrow a_1 \ldots a_m \sim a_{m+1} \ldots \sim a_n$ is defined by means of "$\leftarrow$" in Table 2.1.
(5) A set of ground rules is true in I if each rule is true in I, and false in I if some rule is false in I.

If a set of ground rules is true in a 3-valued (2-valued) Herbrand interpretation I, then I is said to be a 3-valued (2-valued) Herbrand model of the set.

$\leqslant_k \subseteq (2^{B_P} \times 2^{B_P}) \times (2^{B_P} \times 2^{B_P})$ is defined to be

$$(IT_1, IF_1) \leqslant_k (IT_2, IF_2) \text{ iff } IT_1 \subseteq IT_2 \text{ and } IF_1 \subseteq IF_2.$$

$\leqslant_t \subseteq (2^{B_P} \times 2^{B_P}) \times (2^{B_P} \times 2^{B_P})$ is defined to be

$$(IT_1, IF_1) \leqslant_t (IT_2, IF_2) \text{ iff } IT_1 \subseteq IT_2 \text{ and } IF_2 \subseteq IF_1.$$

We have the well-founded model, following [7, 12].

Definition 2.2. A set S of ground atoms is an unfounded set of P with respect to a 3-valued Herbrand interpretation I if each atom $a \in S$ satisfies the following conditions:

For each rule $a \leftarrow a_1 \ldots a_m \sim a_{m+1} \ldots \sim a_n \in P_{\mathrm{gr}}$, either
(1) the body $a_1 \ldots a_m \sim a_{m+1} \ldots \sim a_n$ is false in I, or
(2) some positive literal b of the body $a_1 \ldots a_m \sim a_{m+1} \ldots \sim a_n$ occurs in S.

Because the union of unfounded sets is also an unfounded set, there is the greatest unfounded set of P with respect to I. It is denoted by $GU(I)$.

Definition 2.3. $V_P : 2^{B_P} \times 2^{B_P} \to 2^{B_P} \times 2^{B_P}$ is defined to be

$$V_P(I) = (T_P^+(I), GU(I)),$$

where

$$T_P^+(I) = \{a \mid \exists (a \leftarrow a_1 \ldots a_m \sim a_{m+1} \ldots \sim a_n) \in P_{\mathrm{gr}}: [a_1 \ldots a_m \sim a_{m+1} \ldots \sim a_n \text{ is true in } I]\}.$$

Note that V_P is monotonic with respect to $\leqslant_k$, and thus has the least fixpoint. The least fixpoint of V_P is denoted by $lfp(V_P)$. $lfp(V_P)$ is referred to as the well-founded model.

An alternating fixpoint is presented in relation with the well-founded model [1, 11].

Definition 2.4. Let $K \subseteq B_P$ for a general logic program P. We define

$$P[K] = \{a \leftarrow a_1 \dots a_m \bar{a}_{m+1} \dots \bar{a}_n \mid a \leftarrow a_1 \dots a_m \sim a_{m+1} \dots \sim a_n \in P_{\mathrm{gr}}\} \cup \{\bar{b} \leftarrow \mid b \in K\}.$$

S_P is defined to be

$$S_P(K) = U_{P[K]} \uparrow \omega \cap B_P,$$

where $U_Q \uparrow \omega = \bigcup_{i \in \omega} U_Q \uparrow i$ for a definite program Q, defined by

$$U_Q \uparrow i = \begin{cases} \emptyset & (i = 0), \\ U_Q(U_Q \uparrow (i-1)) & (i > 0) \end{cases}$$

such that

$$U_Q(J) = \{a \mid \exists (a \leftarrow a_1 \dots a_n) \in Q_{\mathrm{gr}} : [a_1, \dots, a_n \in J]\}$$

for $J \subseteq B_Q$.

Definition 2.5. Let P be a general logic program. Then we define $\Theta_P : 2^{B_P} \to 2^{B_P}$ to be $\Theta_P(K) = \overline{S_P(\overline{S_P(K)})}$.

Note that Θ_P is the same operator as A_P in [11], being monotonic with respect to $\subseteq$, so that there is a fixpoint of Θ_P. We call Θ_P to be an alternating operator. Also there is a least fixpoint of Θ_P, which is denoted by $lfp(\Theta_P)$. It is essentially concerned with the well-founded model. Let $lfp(\Theta_P)$ be Θ^*, and $S_P(\Theta^*)$ be Θ^+.

Theorem 2.6 (Van Gelder [11]). *Let (WT, WF) be the well-founded model of a general logic program P. Then $\Theta^+ = WT$ and $\Theta^* = WF$.*

We now review a 3-valued stable model of a general logic program in terms of $\leqslant_t$.

Definition 2.7 (*Giordano et al.* [14], *Przymusinski* [21,22]). Let $I = (T, F)$ be a 3-valued Herbrand interpretation. For a general logic program P, a program P/I is a set of rules obtained from P_{gr} by performing the following three operations:

(1) Removing from P_{gr} all rules which contain a negation of an atom $\sim a$ in their bodies such that $a \in T$,
(2) Replacing in all remaining rules those negations of atoms $\sim a$ such that $a \notin T \cup F$ by u,
(3) Removing from all the remaining rules those negations of atoms $\sim a$ such that $a \in F$.

Because P/I does not involve any negation of an atom, it has the least 3-valued Herbrand model with repect to $\leqslant_t$, which is denoted by $\Psi(P/I)$.

Definition 2.8 (*Przymusinski* [22]). A 3-valued Herbrand interpretation I is a 3-valued stable model of a general logic program P iff $\Psi(P/I)=I$.

Note that the 2-valued stable model [13] is a 3-valued stable model (T,F) such that $T\cup F=B_P$. The following theorem is primary [22].

Theorem 2.9. *Each general logic program has the least* 3*-valued stable model with respect to* $\leqslant_k$*, and is equal to its well-founded model.*

The following theorem is equivalent to Theorem 3.2 of [27].

Theorem 2.10. *Assume that* $I=(S_P(F),F)$ *is a* 3*-valued Herbrand interpretation of a general logic program* P*. Then* $F=\Theta_P(F)$ *iff* I *is a* 3*-valued stable model of* P*.*

3. Extension of Eshghi and Kowalski procedure

We now present an extension of Eshghi and Kowalski abductive procedure, which is sound and complete with respect to the 3-valued stable model semantics.

As an illustration for the extended procedure, assume

$$P=\{p(x)\leftarrow\sim q(x),\ q(y)\leftarrow\sim p(f(y))\},$$

where p, q are predicate symbols, f is a 1-place function symbol and x, y are variables. $(\{q(f^i(a))\mid i\in\omega\},\{p(f^i(a))\mid i\in\omega\})$ is a 3-valued stable model, where f^i stands for i times applications of f and a is a 0-place function symbol not in P. For a failing goal $\leftarrow p(a)$, an infinite negative recursion

$$\leftarrow p(a),\leftarrow\sim q(a),\leftarrow q(a),\leftarrow\sim p(f(a)),\leftarrow p(f(a)),\ldots$$

occurs so that $\sim p(f(a)),\sim p(f(f(a))),\ldots$ should be dealt with as abducibles in the procedure for completeness. The E–K procedure does not deal with such an infinite set of abducibles, however, we need extend it to a complete procedure with respect to the 3-valued stable model.

The (extended) procedure contains succeeding and failing derivations, where:

(1) The infinite sequence caused by negative recursive calls are admitted as well as the infinite sequence caused by SLD resolution.
(2) The infinite sequence is enumerated on the basis of nondeterministic choices, where the effect of the sequence may be expressed by the alternating operator Θ_P.

3.1. Infinite resolution derivation

We deal with the set of all goals derived by SLD resolution, which may be infinite.

$resol_P(F)$ denotes the set of goals derived from a goal set F by using a general logic program P. That is,

$$resol_P(F) = \{\ g' \mid g' = \leftarrow L_1\theta \ldots L_{i-1}\theta L'_1\theta \ldots L'_k\theta L_{i+1}\theta \ldots L_m\theta$$
$$\text{for some } \leftarrow L_1 \ldots L_{i-1}AL_{i+1} \ldots L_m \in F,$$
$$\text{for some } A' \leftarrow L'_1 \ldots L'_k \in P, \text{ and}$$
$$\text{for some most general unifier } \theta \text{ of } A \text{ and } A'\ \}.$$

To describe the whole behaviour caused by SLD resolution for a goal set F, we make use of the set $Res_P(F)$:

$$Res_P(F) = \bigcup_{i\in\omega} resol^i_P(F),$$

where $resol^i_P(F)$ is defined recursively:

$$resol^i_P(F) = \begin{cases} F & (i = 0), \\ resol_P(resol^{i-1}_P(F)) & (i > 0). \end{cases}$$

We say that g is derivable from F if $g \in Res_P(F)$. The derivation denoted by $Res_P(F)$ is concerned with the mapping S_P in the following sense:

$U_Q \uparrow \omega$ denotes the success set caused by SLD resolution for Q. Therefore $S_P(K)$ represents the success set for P on the assumption that $\sim a$ is already derivable for $a \in K$.

We have relations between the derivation Res_P and the mapping S_P, which we can see by means of soundness and completeness of SLD resolution. (See [19] for soundness and completeness of SLD resolution with respect to $U_Q \uparrow \omega$.)

(1) By soundness of SLD resolution,

$$\leftarrow\sim b_1 \ldots \sim b_n \in Res_P(\{\leftarrow a\}) \text{ for a ground negative goal } \leftarrow\sim b_1 \ldots \sim b_n$$
$$\Rightarrow a \in S_P(\{b_1, \ldots, b_n\}).$$

(2) By completeness of SLD resolution,

$$a \in S_P(\Delta^+) \Rightarrow \exists(\leftarrow\sim B_1 \ldots \sim B_m) \in Res_P(\{\leftarrow a\}), \exists\sigma : [\{B_1\sigma, \ldots, B_m\sigma\} \subseteq \Delta^+].$$

3.2. A procedure for the 3-valued stable model

We extend the failing derivation of Eshghi and Kowalski abductive procedure in two respects:

(1) The infinite failure caused by SLD resolution is taken.
(2) The negative recursion for the failing derivations through succeeding derivations is arranged so that an infinite sequence of goal sets is extracted.

The structure of the failing derivation is illustrated: Take an arbitrary negative goal $\leftarrow\sim B_1 \ldots \sim B_n$ derivable from a goal set F. For F to fail, $\leftarrow\sim B_1 \ldots \sim B_n$ must

fail such that there is a succeeding derivation from some ground goal $\leftarrow B_i$. The choice of B_i from $\leftarrow \sim B_1 \ldots \sim B_n$ is nondeterministic. For some ground negative goal $\leftarrow \sim C_1 \ldots \sim C_m$ derivable from $\leftarrow B_i$, each $\leftarrow C_j$ must fail, where each $\sim C_j$ should be in memory. The choice of $\leftarrow \sim C_1 \ldots \sim C_m$ among derivable negative goals from $\leftarrow B_i$ is nondeterministic. We then regard

$$\{\leftarrow C_1, \ldots, \leftarrow C_m\}$$

as an acquired candidate subset to the next goal set. (If $m = 0$ then none is acquired.) These acquisitions may form the sequence of goal sets. When no more goal set is to be acquired, the sequence is regarded as completed.

The failing derivation is constructed by means of a completed sequence of goal sets, while the succeeding derivation may involve failing derivations.

$GoalS$ stands for the power set of a set $Goal$ of goals, namely 2^{Goal}, $NegG$ for a set of negative goals and Ab for a set of ground negative literals, that is, a set of abducibles. (As in abduction framework [16], we call a set of ground negative literals to be a set of abducibles.) Given a goal g and a set $\Delta \subseteq Ab$, we denote the goal obtained by deleting the literals in Δ from g, by $g - \Delta$.

The following procedure involves nondeterministic choices. The choices are implicitly made so that the procedure represents a relation.

$$Fail \subseteq GoalS \times 2^{Ab} \times (GoalS \cup \{error, fl\}) \times (2^{Ab} \cup \{\bot\})$$

is defined as follows.

procedure $Fail(F, \Delta, F', \Delta')$;
  begin
    $F' := \emptyset$;
    $\Delta' := \Delta$;
    if $Res_P(F) \cap NegG \neq \emptyset$ **then**
      for each $g \in Res_P(F) \cap NegG$
        begin
          $g := g - \Delta$;
          if $g = \square$ **then begin** $F' := error$; $\Delta' := \bot$; **exit end**;
          choose some $\sim A$ in g;
          if A contains some variable
              then begin $F' := fl$; $\Delta' := \bot$; **exit end**;
          choose some $g' \in Res_P(\{\leftarrow A\}) \cap NegG$;
          $g' := g' - \Delta$;
          if g' contains some variable
            then begin $F' := fl$; $\Delta' := \bot$; **exit end**;
          $F' := F' \cup \{\leftarrow B \mid \sim B$ is contained in $g'\}$;
          $\Delta' := \Delta' \cup \{\sim B \mid \sim B$ is contained in $g'\}$
        end
  end

If we have got $Fail(F, \Delta, fl, \perp)$, then we encounter a case of floundering in a derivation. When we have $Fail(F, \Delta, error, \perp)$, the failing derivation is never constructed, because $\square$ is obtained. In the procedure $Fail$, for any negative goal g derived from the given goal set, if some ground negative goal g' is obtained in a succeeding derivation from a goal $\leftarrow A$, where $\sim A$ appears in the goal g, then the negative literals included in the negative goal g' can be added as new abducibles.

We now define the succeeding and failing derivations recursively.

(1) A succeeding derivation:
Assume $(g_0, \Delta_0) \in Goal \times 2^{Ab}$. A succeeding derivation is a sequence

$$(g_0, \Delta_0, \theta_0), \ldots, (g_n, \Delta_n, \theta_n),$$

where
(i) $g_0, \ldots, g_n$ are goals,
(ii) $\Delta_0 \subseteq \Delta_1 \subseteq \ldots \subseteq \Delta_n \subseteq Ab$,
(iii) $\theta_0 = \varepsilon$ and $\theta_1, \ldots, \theta_n$ are substitutions such that g_{i+1}, Δ_{i+1} and θ_{i+1} are obtained from g_i and Δ_i by one of the following rules.
(a) $g_{i+1} \in resol_P(\{g_i\})$ by means of a most general unifier θ such that $\theta_{i+1} = \theta$, and $\Delta_{i+1} = \Delta_i$.
(b) $g_{i+1} = g_i - \Delta_i$, $\Delta_{i+1} = \Delta_i$, and $\theta_{i+1} = \varepsilon$.
(c) $g_{i+1} = g_i - \{\sim A\}$, $\Delta_{i+1} = \Delta'$, and $\theta_{i+1} = \varepsilon$, if $\sim A$ is a ground literal such that $\sim A \notin \Delta_i$ and there is a failing derivation:

$$(\{\leftarrow A\}, \Delta_i \cup \{\sim A\}) \rightsquigarrow_{Fail} \Delta'.$$

(See this notation in the failing derivation.)

Notation: If, for the succeeding derivation

$$(g_0, \Delta_0, \theta_0), \ldots, (g_n, \Delta_n, \theta_n),$$

$g_n = \square$ and $\theta = (\theta_0\theta_1\theta_2 \ldots \theta_n)|_{g_0}$, then we denote the derivation by $(g_0, \Delta_0) \overset{\theta}{\rightsquigarrow}_{Suc} \Delta_n$.

(2) A failing derivation:
Assume $(F_0, \Delta_0) \in GoalS \times 2^{Ab}$. A failing derivation is

$$(F_0, \Delta_0), \ldots, (F_\alpha, \Delta_\alpha),$$

where
(i) $F_0, \ldots, F_\alpha \in GoalS$,
(ii) $\Delta_0, \ldots, \Delta_\alpha \in 2^{Ab}$ such that (F_β, Δ_β) is defined by means of the procedure $Fail$ (defined as above) and induction:

$$(F_\beta, \Delta_\beta) = \begin{cases} Fail(F_n, \Delta_n, F_{n+1}, \Delta_{n+1}) & \text{if } \beta \text{ is a successor ordinal } n+1, \\ \left(\bigcup_{\gamma<\beta} F_\gamma, \bigcup_{\gamma<\beta} \Delta_\gamma \right) & \text{if } \beta \text{ is a limit ordinal.} \end{cases}$$

Notation: If, for the failing derivation $(F_0, \Delta_0), \ldots, (F_\alpha, \Delta_\alpha)$, we have $\Delta_{\alpha+1} = \Delta_\alpha$, where

$$Fail(F_\alpha, \Delta_\alpha, F_{\alpha+1}, \Delta_{\alpha+1}),$$

then the derivation is denoted by $(F_0, \Delta_0) \rightsquigarrow_{Fail} \Delta_\alpha$. Note that $F_{\alpha+1} = \emptyset$ by the procedure *Fail*.

Pereira et al. [20] demonstrates a procedure to compute the 3-valued stable model of a grounded program. As an illustration, assume a grounded general logic program

$$\begin{aligned} P = \{ \quad & p \leftarrow q \sim s, \\ & q \leftarrow\sim r, \\ & r \leftarrow\sim q, \\ & s \leftarrow s \quad \}, \end{aligned}$$

where $(\{p,q\},\{r,s\})$, $(\{r\},\{p,q,s\})$ and $(\emptyset,\{s\})$ are different 3-valued stable models. We see a derivation for p:

(1) $p \Rightarrow q \sim s$ (by resolution with $p \leftarrow q \sim s$), followed by the subderivation of consecutive sequences:

(2) $q \Rightarrow \sim r$ (by resolution with $q \leftarrow \sim r$); $\sim r \Rightarrow q$ (by resolution with $r \leftarrow \sim q$), which reach the loop for q and incorporate the derivation to be continued:

(3) $\sim s \Rightarrow \sim s$ (by resolution with $s \leftarrow s$), which reaches the loop for $\sim s$.

It follows from the derivation for p that p should be included in $\{p,q\}$ of the 3-valued stable model $(\{p,q\},\{r,s\})$. The method has the characteristics of loop checking techniques for a grounded program.

We now have an example, where an infinite expansion of abducibles may occur. Even if such a case is involved, our procedure is well defined.

Example 3.1. Let

$$\begin{aligned} P = \{ \quad & p(x) \leftarrow q(a), \\ & q(y) \leftarrow\sim r(y), \\ & r(z) \leftarrow\sim q(f(z)) \quad \}, \end{aligned}$$

where p, q, r are predicate symbols, x, y, z are variables, and a, f are 0-place, 1-place function symbols, respectively. For the goal set

$$F_0 = \{\leftarrow p(x')\}\,(x' : \text{a variable})$$

with the set of negative literals $\emptyset$, the goal sets

$$F_1 = \{\leftarrow q(f(a))\} \quad \text{and} \quad F_2 = \{\leftarrow q(f(f(a)))\}$$

are provided, where some derivations from $\leftarrow r(a)$ and $\leftarrow r(f(a))$ are taken. It is easy to see that

$$\leftarrow\sim q(f(a)) \in Res_P(\{\leftarrow r(a)\}) \cap NegG, \text{ and}$$

$$\leftarrow\sim q(f(f(a))) \in Res_P(\{\leftarrow r(f(a))\}) \cap NegG, \text{ respectively.}$$

Hence we have

$$Fail(\{\leftarrow p(x')\}, \emptyset, \{\leftarrow q(f(a))\}, \{\sim q(f(a))\}).$$

As well, we have

$$Fail(\{\leftarrow q(f(a))\}, \{\sim q(f(a))\}, \{\leftarrow q(f(f(a)))\}, \{\sim q(f(a)), \sim q(f(f(a)))\}).$$

Taking

$$F_\omega = \{\leftarrow p(x'), \leftarrow q(f(a)), \leftarrow q(f(f(a))), \ldots\},$$
$$\Delta_\omega = \{\sim q(f(a)), \sim q(f(f(a))), \ldots\},$$

we have

$$Fail(F_\omega, \Delta_\omega, \emptyset, \Delta_\omega).$$

That is

$$(\{\leftarrow p(x')\}, \emptyset) \rightsquigarrow_{Fail} \Delta_\omega.$$

With a modification of floundering as in [19, 21], we re-formulate the notion of floundering for our procedure.

Definition 3.2. Given a general logic program P, a failing derivation

$$(F_0, \Delta_0), \ldots, (F_\alpha, \Delta_\alpha)$$

flounders if the derivation is not to be followed up, owing to "fl" caused by $Fail(F_\alpha, \Delta_\alpha, fl, \perp)$. A succeeding derivation

$$(g_0, \Delta_0, \theta_0), \ldots, (g_n, \Delta_n, \theta_n)$$

flounders if the derivation is not to be followed up, taking a nonground negative literal in g_n, or reaching a floundering failing derivation from $\{\leftarrow A\}$ for some ground negative literal $\sim A$ in g_n, owing to the rule (c).

4. Soundness of procedure with respect to 3-valued stable model semantics

In this section, we have proofs for soundness of the presented procedure with respect to the 3-valued stable model. We show that the failing derivation could be at most ω-sequence under some condition.

4.1. Alternating mapping in relation to succeeding and failing derivations

By the relation of $S_P(\Delta^+) = U_{P[\Delta^+]} \uparrow \omega \cap B_P$ to the derivability of Res_P from a general logic program P with a set of ground negative literals Δ, we represent the effects of succeeding and failing derivations in terms of the mapping S_P.

Lemma 4.1. *Assume that* $(\leftarrow L_1 \dots L_n, \Delta_0) \overset{\theta}{\rightsquigarrow}_{Suc} \Delta$. *Then*

$$\forall L_i, \forall \sigma : [[(L_i\theta)\sigma = a \in B_P \Rightarrow a \in S_P(\Delta^+)] \wedge [(L_i\theta)\sigma = \sim a \in B_P^{\sim} \Rightarrow a \in \Delta^+]].$$

Proof. It is proved by induction on the steps k of the derivation.

(1) In case that $k = 0$, $n = 0$ and it obviously holds.

(2) Assume that the lemma holds for the case that $k \leqslant l$. Suppose that $k = l + 1$. $\leftarrow L_1 \dots L_n$ is reduced to some

$$\leftarrow L_1\theta_1 \dots L_{i-1}\theta_1 M_1\theta_1 \dots M_m\theta_1 L_{i+1}\theta_1 \dots L_n\theta_1 \in resol_P(\{\leftarrow L_1 \dots L_n\})$$

or to $\leftarrow L_1 \dots L_{j-1}L_{j+1} \dots L_n$, where $L_j = \sim a$ is a ground negative literal. In the latter case, the induction step is easily completed by the induction hypothesis for $\leftarrow L_1 \dots L_{j-1}$ $L_{j+1} \dots L_n$, while we see that $a \in \Delta^+$. In the former case, by applying the induction hypothesis, we have

$$\begin{aligned}
\forall L_j\ (\neq L_i), \forall \sigma :\ & [[(L_j\theta_1)\theta'\sigma = a \in B_P \Rightarrow a \in S_P(\Delta^+)] \\
& \wedge [(L_j\theta_1)\theta'\sigma = \sim a \in B_P^{\sim} \Rightarrow a \in \Delta^+]] \\
\wedge \forall M_k, \forall \rho :\ & [[(M_k\theta_1)\theta'\rho = b \in B_P \Rightarrow b \in S_P(\Delta^+)] \\
& \wedge [(M_k\theta_1)\theta'\rho = \sim b \in B_P^{\sim} \Rightarrow b \in \Delta^+]],
\end{aligned}$$

where $\theta' = (\theta_2\theta_3 \dots \theta_k)|_{(\leftarrow L_1\theta_1 \dots L_{i-1}\theta_1 M_1\theta_1 \dots M_m\theta_1 L_{i+1}\theta_1 \dots L_n\theta_1)}$. For $j = 1, \dots, i-1, i+1, \dots, n$,

$$\begin{aligned}
&(L_j\theta_1)\theta'\sigma = L_j\theta\sigma = a \in B_P \Rightarrow a \in S_P(\Delta^+), \text{ and} \\
&(L_j\theta_1)\theta'\sigma = L_j\theta\sigma = \sim a \in B_P^{\sim} \Rightarrow a \in \Delta^+.
\end{aligned}$$

Note that $(L_j\theta_1)\theta'\sigma = (L_j\theta)\sigma$. As well, it is easy to see that if $(L_i\theta)\rho = (L_i\theta_1)\theta'\rho \in B_P$ then $(L_i\theta)\rho \in S_P(\Delta^+)$, because

$$\begin{aligned}
\forall M_k, \forall \rho :\ & [[(M_k\theta_1)\theta'\rho = a \in B_P \Rightarrow a \in S_P(\Delta^+)] \\
& \wedge [(M_k\theta_1)\theta'\rho = \sim a \in B_P^{\sim} \Rightarrow a \in \Delta^+]].
\end{aligned}$$

This completes the induction step. □

Lemma 4.2. *Assume that* $(F_0, \Delta_0) \rightsquigarrow_{Fail} \Delta_\alpha$. *Then*

$$\forall g \in F_0 : [g = \leftarrow L_1 \ldots L_n \Rightarrow \forall \sigma, \exists\, L_i : [[L_i\sigma = a \in B_P \Rightarrow a \notin S_P(\overline{S_P(\Delta_\alpha^+)})] \wedge [L_i\sigma = \sim\ a \in B_P^{\sim} \Rightarrow a \in S_P(\Delta_\alpha^+)]]].$$

Proof. Assume that for some $g = \leftarrow L_1 \ldots L_n \in F_0$,

$$\exists \sigma, \forall L_i : [[L_i\sigma = a \in B_P \wedge a \in S_P(\overline{S_P(\Delta_\alpha^+)})] \vee [L_i\sigma = \sim\ a \in B_P^{\sim} \wedge a \in \overline{S_P(\Delta_\alpha^+)}]].$$

Because $L_i\sigma = a \in S_P(\overline{S_P(\Delta_\alpha^+)})$, or $L_i\sigma = \sim a$ such that $a \in \overline{S_P(\Delta_\alpha^+)}$, by the completeness of SLD resolution, $Res_P(\{g\})$ contains a negative goal $\leftarrow \sim A_1 \ldots \sim A_m$ such that

$$\exists \theta, \forall\ A_i : [A_i\theta \in \overline{S_P(\Delta_\alpha^+)}].$$

In the procedure $Fail(F_0, \Delta_0, F_1, \Delta_1)$, there may be two cases.

(1) In case that $Res_P(\{g\}) \cap NegG = \emptyset$, $Res_P(\{g\})$ contains no negative goal, which is a contradiction to the assumption.

(2) In case that $Res_P(\{g\}) \cap NegG \neq \emptyset$, some ground negative literal $\sim A_i \notin \Delta_0$ is chosen such that some ground negative goal g' is derivable from $\{\leftarrow A_i\}$, because we are now with the non-floundering derivation. Assume that $g' - \Delta_0 = \leftarrow \sim b_1 \ldots \sim b_l$. Then we have

$$\{\sim b_1, \ldots, \sim b_l\} \subseteq \Delta_1.$$

It follows that $A_i \in S_P(\Delta_1^+)$. By monotonicity of S_P, $A_i \in S_P(\Delta_\alpha^+)$. It contradicts that $A_i = A_i\theta \in \overline{S_P(\Delta_\alpha^+)}$. Therefore, the lemma holds. □

4.2. Soundness of succeeding and failing derivations

We have basic lemmas to present the preservation of consistency in succeeding and failing derivations, when the initial set of abducibles is empty.

Lemma 4.3. *Assume that* $(F_0, \Delta_0) \rightsquigarrow_{Fail} \Delta_\alpha$, *where* $\Delta_0 = \emptyset$. *Then*

(1) $S_P(\Delta_\alpha^+) \cap \Delta_\alpha^+ = \emptyset$.

(2) $\Delta_\alpha^+ \subseteq \Theta_P(\Delta_\alpha^+)$.

Proof. (1) Assume the failing derivation $(F_0, \Delta_0), \ldots, (F_\alpha, \Delta_\alpha)$. By the definition of the failing derivation, for $1 \leqslant \beta \leqslant \alpha$,

$$a \in \Delta_\beta^+ \ \Leftrightarrow\ \exists \beta' : [[1 \leqslant \beta' \leqslant \beta] \wedge [\leftarrow a \in F_{\beta'}]].$$

Suppose that $a \in S_P(\Delta_\alpha^+) \cap \Delta_\alpha^+$. Because $a \in \Delta_\alpha^+$ implies $\leftarrow a \in F_\beta$ for some $1 \leqslant \beta \leqslant \alpha$, $[a \in S_P(\Delta_\alpha^+)] \wedge [\leftarrow a \in F_\beta]$.

(i) In case that $\alpha = \beta$: Because $a \in S_P(\Delta_\alpha^+)$, for some ground negative goal $g \in Res_P(\{\leftarrow a\}) \cap NegG$, $g - \Delta_\alpha = \square$ as long as we have a non-floundering derivation.

That is, the empty clause $\square$ is got with Δ_α. It contradicts that $\leftarrow a \in F_\beta$, where F_β is a goal set for the failing derivation.

(ii) In case that $\beta < \alpha$:

(a) Assume that α is finite. Because $a \in S_P(\Delta_\alpha^+)$, we see that

$$\leftarrow \sim B_1 \ldots \sim B_n \in Res_P(\{\leftarrow a\}) \cap NegG$$

such that $\{B_1\rho, \ldots, B_n\rho\} \subseteq \Delta_\alpha^+$ for some substitution ρ. On the other hand, owing to the failing derivation not providing fl, some ground negative goal is derivable from some ground $\leftarrow B_i$ such that B_i is in $S_P(\Delta_{\beta+1}^+) \subseteq S_P(\Delta_\alpha^+)$, but not in Δ_β^+. That is,

$$B_i \in S_P(\Delta_\alpha^+) \cap \Delta_{\beta'}^+ \quad \text{and} \quad \leftarrow B_i \in F_{\beta'}$$

for some β' such that $\beta < \beta'$. It follows that

$$[b \in S_P(\Delta_\alpha^+) \wedge \leftarrow b \in F_\gamma] \Rightarrow \exists \gamma' : [[\gamma < \gamma'] \wedge \exists c : [c \in S_P(\Delta_\alpha^+) \wedge \leftarrow c \in F_{\gamma'}]].$$

By induction,

$$\exists d : [d \in S_P(\Delta_\alpha^+) \wedge \leftarrow d \in F_\alpha],$$

which is reduced to the case of (i).

(b) Assume that $\alpha \geqslant \omega$: For some limit ordinal α' and some finite ordinal k, $\alpha = \alpha' + k$.

(b-1) If $\beta \geqslant \alpha'$, then the proof is completed for the same reason as in (a).

(b-2) If $\beta < \alpha'$, then $\leftarrow a \in F_\beta \subseteq F_{\alpha'}$. This case is reduced to the case of (a).

(2) Assume that $a \in \Delta_\alpha^+$. It follows that $\leftarrow a \in F_\beta$ for some $1 \leqslant \beta \leqslant \alpha$. Because $(F_\beta, \Delta_\beta) \rightsquigarrow_{Fail} \Delta_\alpha$, by Lemma 4.2, $a \notin S_P(\overline{S_P(\Delta_\alpha^+)})$. That is, $a \in \Theta_P(\Delta_\alpha^+)$ so that $\Delta_\alpha^+ \subseteq \Theta_P(\Delta_\alpha^+)$. $\square$

Lemma 4.4. *Given a general logic program P, assume that* $(g_0, \emptyset) \rightsquigarrow_{Suc}^{\theta} \Delta$. *Then*

(1) $S_P(\Delta^+) \cap \Delta^+ = \emptyset$.

(2) $\Delta^+ \subseteq \Theta_P(\Delta^+)$.

Proof. The succeeding derivation $(g_0, \emptyset) \rightsquigarrow_{Suc}^{\varepsilon} \Delta$ contains at most finitely many failing derivations. We list l failing derivations in order by

$$(F_0^n, \Delta_0^n), \ldots, (F_{\alpha_n}^n, \Delta_{\alpha_n}^n),$$

where $0 \leqslant n \leqslant l-1$ and

$$F_0^n = \{\leftarrow a_n\},$$

$$\Delta_0^n = \Delta_n \cup \{\sim a_n\} (\sim a_n \notin \Delta_n),$$

$$\Delta_{\alpha_n}^n = \Delta_{n+1}$$

for $\Delta_0 = \emptyset$ and $\Delta_l = \Delta$.

(1) We show by induction that $S_P(\Delta_n^+) \cap \Delta_n^+ = \emptyset$.

(i) In case that $n = 0$, it is trivial, since $\Delta_0^+ = \emptyset$.

(ii) Assume that $S_P(\Delta_n^+) \cap \Delta_n^+ = \emptyset$ for $n \leqslant k$. Now assume that $b \in S_P(\Delta_{k+1}^+) \cap \Delta_{k+1}^+$. $b \in \Delta_{k+1}^+$ implies $\leftarrow b \in F_\beta^n$ for some goal set F_β^n, where $0 \leqslant \beta \leqslant \alpha_n$ and $0 \leqslant n \leqslant k$. Since $b \in S_P(\Delta_{k+1}^+)$, there is

$$\leftarrow \sim D_1 \ldots \sim D_m \in Res_P(\{\leftarrow b\}) \cap NegG$$

such that $\{\sim D_1\rho, \ldots, \sim D_m\rho\} \subseteq \Delta_{k+1}$ for some substitution ρ. Because $\leftarrow b$ is in some goal set F_β^n, some ground negative goal $\leftarrow \sim e_1 \ldots \sim e_{m'}$ is derivable from some ground goal $\leftarrow D_i$ such that $D_i \notin \Delta_\beta^n$, where $\{e_1, \ldots, e_{m'}\} \subseteq \Delta_{n+1}^+$. It follows that $D_i \in S_P(\Delta_{n+1}^+)$.

(i) In case that $n \leqslant k-1$ and $D_i \in \Delta_k^+$, $D_i \in S_P(\Delta_k^+)$. It is a contradiction to the induction hypothesis that $S_P(\Delta_k^+) \cap \Delta_k^+ = \emptyset$.

(ii) In case that $n = k$ or $D_i \in \Delta_{k+1}^+ - \Delta_k^+$:

(a) If $n = k$ then $\leftarrow b \in F_\beta^k$.

(b) If $D_i \in \Delta_{k+1}^+ - \Delta_k^+$ then $\leftarrow D_i \in F_{\beta'}^k$ for some β'.

In both cases, there is b' such that $b' \in S_P(\Delta_{k+1}^+)$ and $\leftarrow b' \in F_\gamma^k$ for some γ. As in the proof of Lemma 4.3, we see that

$$\exists c : [[\leftarrow c \in F_{\alpha_k}^k] \wedge [c \in S_P(\Delta_{k+1}^+) = S_P(\Delta_{\alpha_k}^{k+})]]$$

which will lead to the contradiction as in Lemma 4.3. That is, we have an illegal assumption that $b \in S_P(\Delta_{k+1}^+) \cap \Delta_{k+1}^+$. This completes the induction step.

(2) For any $a \in \Delta^+$, there is some goal set F_β^k such that $\leftarrow a \in F_\beta^k$ for $0 \leqslant k \leqslant l-1$ and $0 \leqslant \beta \leqslant \alpha_k$. By Lemma 4.2, $a \notin S_P(\overline{S_P(\Delta_{k+1}^+)})$. It follows that $a \in \Theta_P(\Delta_{k+1}^+) \subseteq \Theta_P(\Delta^+)$. Hence $\Delta^+ \subseteq \Theta_P(\Delta^+)$. □

The following lemma is to state that any failing derivation preserving consistency can be defined as at most ω-sequence.

Lemma 4.5. *Assume a failing derivation* $(F_0, \Delta_0) \rightsquigarrow_{Fail} \Delta_\alpha$, *where* $S_P(\Delta_\alpha^+) \cap \Delta_\alpha^+ = \emptyset$. *Then* α *could be at most* ω.

Proof. If α is finite or ω, the lemma is trivial. Now assume that $\alpha > \omega$. Let

$$F_\omega = \bigcup_k F_k,$$

$$\Delta_\omega = \bigcup_k \Delta_k.$$

For any negative goal $g \in Res_P(F_i) \cap NegG$, there is a ground negative literal $\sim A$ in $g - \Delta_i$ such that some ground negative goal $\leftarrow \sim B_1 \ldots \sim B_n$ is derivable from $\{\leftarrow A\}$, where

$$\{\sim B_1, \ldots, \sim B_n\} \subseteq \Delta_{i+1}.$$

Note that $\sim A \notin \bigcup_k \Delta_k$. Otherwise, $A \in \bigcup_k \Delta_k^+ \subseteq \Delta_\alpha^+$ and $A \in S_P(\Delta_{i+1}^+) \subseteq S_P(\bigcup_k \Delta_k^+) \subseteq S_P(\Delta_\alpha^+)$. It follows that $S_P(\Delta_\alpha^+) \cap \Delta_\alpha^+ \neq \emptyset$, which contradicts the assumption of this

lemma. Therefore, $\sim A$ is included in $g - \Delta_\omega$. Because $\Delta_{i+1} \subseteq \Delta_\omega$, $\{B_1, \ldots, B_n\} \subseteq \Delta_\omega^+$ so that

$$g' - \Delta_\omega = \square \quad \text{for some } g' \in Res_P(\{\leftarrow A\}) \cap NegG.$$

Hence

$$Fail\left(F_i, \bigcup_k \Delta_k, \emptyset, \bigcup_k \Delta_k\right).$$

Note that $Res_P(\bigcup_i F_i) = \bigcup_i Res_P(F_i)$. Also $Res_P(\bigcup_i F_i) \cap NegG = (\bigcup_i Res_P(F_i)) \cap NegG$. We thus have

$$Fail(F_\omega, \Delta_\omega, \emptyset, \Delta_\omega). \qquad \square$$

For the failing derivation $(F_0, \Delta_0) \rightsquigarrow_{Fail} \Delta_\alpha$ with $\Delta_0 = \emptyset$, $S_P(\Delta_\alpha^+) \cap \Delta_\alpha^+ = \emptyset$ by Lemma 4.3, so that Lemma 4.5 is applicable. That is, the failing derivation sequence could be at most ω.

The following lemma is concerned with the preservation of consistency under the operator of Θ_P.

Lemma 4.6. *Given a general logic program P and Δ_0, we define $\Delta_\alpha (\alpha \geqslant 1)$ as follows.*

$$\Delta_\alpha^+ = \begin{cases} \Theta_P(\Delta_{\alpha-1}^+) & \text{if } \alpha \text{ is a successor,} \\ \bigcup_{\beta<\alpha} \Delta_\beta^+ & \text{if } \alpha \text{ is a limit ordinal.} \end{cases}$$

We denote the set of ground negative literals by Ab. For $X \subseteq Ab$, let $Q(X^+)$ be a predicate to represent that $S_P(X^+) \cap X^+ = \emptyset$, where we assume a complete lattice $(Ab^+, \subseteq)$.

(1) *Q is preserved under Θ_P.*

(2) *Q is inclusive, that is, $\forall H \subseteq Ab^+(chain)$: $[\forall Y_\beta \in H : [Q(Y_\beta)]$ implies $Q(\cup H)]$.*

Proof. (1) By monotonicity of S_P, we have

$$\begin{aligned} Q(\Delta_\alpha^+) &\Leftrightarrow \Delta_\alpha^+ \subseteq \overline{S_P(\Delta_\alpha^+)} \\ &\Rightarrow S_P(\Delta_\alpha^+) \subseteq S_P(\overline{S_P(\Delta_\alpha^+)}) \\ &\Rightarrow \Delta_{\alpha+1}^+ = \overline{S_P(\overline{S_P(\Delta_\alpha^+)})} \subseteq \overline{S_P(\Delta_\alpha^+)} \\ &\Rightarrow S_P(\Delta_{\alpha+1}^+) \subseteq S_P(\overline{S_P(\Delta_\alpha^+)}) \\ &\Rightarrow \Delta_{\alpha+1}^+ = \Theta_P(\Delta_\alpha^+) \subseteq \overline{S_P(\Delta_{\alpha+1}^+)} = \overline{S_P(\Theta_P(\Delta_\alpha^+))} \\ &\Rightarrow Q(\Theta_P(\Delta_\alpha^+)). \end{aligned}$$

This concludes the preservation of Q under Θ_P.

(2) Assume that $Q(\cup H)$ does not hold for some chain H, even if

$$\forall Y_\beta \in H : [Q(Y_\beta)].$$

It follows from the definition of S_P for a sufficiently large set Y_β that

$$\exists a : [a \in S_P(\cup H) \text{ and } a \in \cup H] \Rightarrow \exists Y_\beta \in H : [a \in S_P(Y_\beta) \text{ and } a \in Y_\beta].$$

This contradicts $Q(Y_\beta)$. Therefore, $Q(\cup H)$ for any chain $H \subseteq Ab^+$. □

It is shown in [27] that for the fixpoint Δ_f of Θ_P such that $S_P(\Delta_f^+) \cap \Delta_f^+ = \emptyset$, we have a 3-valued stable model $(S_P(\Delta_f^+), \Delta_f^+)$. Collecting above lemmas together, we have the soundness theorem with respect to the 3-valued stable model.

Theorem 4.7. *Given a general logic program P, the succeeding and failing derivations are sound with respect to the* 3*-valued stable model. That is,*
(1) *If* $(g_0, \emptyset) \rightsquigarrow^{\theta}_{Suc} \Delta$ *for* $g_0 = \leftarrow L_1 \ldots L_m$, *then there exists a fixpoint* Δ_f *of* Θ_P *such that*
(i) $\Delta^+ \subseteq \Delta_f^+$,
(ii) $S_P(\Delta_f^+) \cap \Delta_f^+ = \emptyset$, *and*
(iii)

$$\forall L_i, \forall \sigma: [[(L_i\theta)\sigma = a \in B_P \Rightarrow a \in S_P(\Delta_f^+)] \wedge [(L_i\theta)\sigma = \sim a \in B_P^{\sim} \Rightarrow a \in \Delta_f^+]].$$

(2) *If* $(F_0, \emptyset) \rightsquigarrow_{Fail} \Delta_\alpha$ *for* $F_0 = \bigcup_i \{\leftarrow L_1^i \ldots L_{m_i}^i\}$, *then there exists a fixpoint* Δ_f *of* Θ_P *such that*
(i) $\Delta_\alpha^+ \subseteq \Delta_f^+$,
(ii) $S_P(\Delta_f^+) \cap \Delta_f^+ = \emptyset$, *and*
(iii)

$$\forall i, \forall \sigma, \exists L_k^i: [[L_k^i\sigma = a \in B_P \Rightarrow a \in \Delta_f^+] \wedge [L_k^i\sigma = \sim a \in B_P^{\sim} \Rightarrow a \in S_P(\Delta_f^+)]].$$

Proof. (1) (i) By Lemma 4.4(2), $\Delta^+ \subseteq \Theta_P(\Delta^+)$. Since Θ_P is monotonic, there is a fixpoint Δ_f of Θ_P such that $\Delta^+ \subseteq \Delta_f^+ = \Delta_\alpha^+$ for some ordinal α, where Δ_α is defined as in Lemma 4.6 for $\Delta_0 = \Delta$.

(ii) By Lemma 4.4(1), $S_P(\Delta^+) \cap \Delta^+ = \emptyset$. Using Lemma 4.6 and fixpoint induction, we see that $S_P(\Delta_f^+) \cap \Delta_f^+ = \emptyset$.

(iii) This is straightforward from Lemma 4.1 and monotonicity of S_P. This concludes the proof.
(2) Replacing Δ in the proof of (1) by Δ_α, Lemma 4.1 by Lemma 4.2 and Lemma 4.4 by Lemma 4.3, respectively, we have the proof. □

5. Completeness of procedure with respect to 3-valued stable model semantics

This section presents the completeness of the procedure in the non-floundering derivation with respect to the 3-valued stable model semantics.

For the empty set, we have the well-founded model $(S_P(\Theta_P^{\alpha}(\emptyset)), \Theta_P^{\alpha}(\emptyset))$ of a general logic program P, where $\Theta_P^{\alpha}(\emptyset)$ is a least fixpoint of Θ_P for some ordinal α. On the other hand, for some set $\Delta^+ \neq \emptyset$ such that $\Delta^+ \subseteq \Theta_P(\Delta^+)$ and $S_P(\Delta^+) \cap \Delta^+ = \emptyset$, we have $(S_P(\Theta_P^{\beta}(\Delta^+)), \Theta_P^{\beta}(\Delta^+))$ as a 3-valued stable model, where $\Theta_P^{\beta}(\Delta^+)$ is a fixpoint of Θ_P for some ordinal β. That is, based on the alternating fixpoint approach, some nonempty set is needed for the fixpoint which is a 3-valued stable model, but not with the well-founded model.

Our derivations can begin with the empty set as the set of abducibles, even with respect to 3-valued stable models. The set of abducibles is recognized both as implementations of negations as failure, and as requirements for the denotation of the alternating fixpoint.

We have an example, where

(1) the failing derivation starts with the empty set of abducibles,

(2) the failing derivation contains a finite sequence of goal sets, and

(3) the failing derivation provides an abducible set for which we have a 3-valued stable model in terms of an alternating fixpoint.

Example 5.1. Consider a general logic program

$$\begin{aligned} P = \{\ & fail(a) && \leftarrow \sim q(x)\, term(x), \\ & q(x) && \leftarrow \sim p(x), \\ & p(f(x)) && \leftarrow \sim r(x), \\ & r(x) && \leftarrow \sim p(x), \\ & term(f(x)) && \leftarrow term(x), \\ & term(a) && \leftarrow \qquad\qquad \}, \end{aligned}$$

where *fail*, p, q, r, *term* are predicate symbols, x is a variable, and a, f are 0-place, 1-place function symbols, respectively. For $\Delta = \{\sim p(a)\}$,

$$\Theta_P^{\omega}(\Delta^+) = \{fail(f^{n+1}(a)) \mid n \in \omega\} \cup \{p(f^n(a)) \mid n \in \omega\},$$
$$\Theta_P^{\omega+1}(\Delta^+) = \{fail(f^n(a)) \mid n \in \omega\} \cup \{p(f^n(a)) \mid n \in \omega\},$$

where f^i stands for i times applications of f. $\Theta_P^{\omega+1}(\Delta^+)$ is the least fixpoint such that $(S_P(\Theta_P^{\omega+1}(\Delta^+)), \Theta_P^{\omega+1}(\Delta^+))$ is a 3-valued stable model.

On the other hand, for the goal set $\{\leftarrow fail(a)\}$ with the empty set of abducibles,

$$Fail(\{\leftarrow fail(a)\}, \emptyset, \{\leftarrow p(f^n(a)) \mid n \in \omega\}, \{\sim p(f^n(a)) \mid n \in \omega\}),$$
$$Fail(\{\leftarrow p(f^n(a)) \mid n \in \omega\}, \{\sim p(f^n(a)) \mid n \in \omega\}, \emptyset, \{\sim p(f^n(a)) \mid n \in \omega\}).$$

That is,

$$(\{\leftarrow fail(a)\},\emptyset) \rightsquigarrow_{Fail} \{\sim p(f^n(a)) \mid n \in \omega\}.$$

By applying Θ_P to the set of abducibles $\{\sim p(f^n(a)) \mid n \in \omega\}$, we have

$$\Theta_P(\{p(f^n(a)) \mid n \in \omega\}) = \Theta_P^{\omega+1}(\Delta^+).$$

We next consider the case that we suffer from the floundering:

Example 5.2. Assume a general logic program

$$\begin{aligned} P = \{ \quad & p(a) \leftarrow\sim q(x) \sim s(a), \\ & q(a) \leftarrow\sim r(a), \\ & r(a) \leftarrow\sim q(a), \\ & s(a) \leftarrow\sim s(a) \qquad \}, \end{aligned}$$

where p, q, r, s are predicate symbols, x is a variable and a a 0-place function symbol. We can see that $(\{q(a)\},\{p(a),r(a)\})$ is a 3-valued stable model such that

$$\overline{S_P(\overline{S_P(\{p(a),r(a)\})})} = \{p(a),r(a)\}, \text{ and}$$
$$S_P(\{p(a),r(a)\}) = \{q(a)\}.$$

Now we take $\Delta_0 = \emptyset$. By SLD resolution,

$$\leftarrow\sim q(x)\sim s(a) \in Res_P(\{\leftarrow p(a)\}) \cap NegG.$$

Note that

$$S_P(\emptyset) = \emptyset,$$
$$S_P(\overline{S_P(\emptyset)}) = \{p(a), q(a), r(a), s(a)\}.$$

For the 3-valued stable model $(\{q(a)\},\{p(a),r(a)\})$, $\leftarrow p(a)$ cannot fail, as long as (1) we have a negative goal

$$\leftarrow\sim q(x) \sim s(a)$$

derivable from $\leftarrow p(a)$, and
(2) we have the ground negative literal $\sim s(a)$ in the goal, where we have not got

$$(\leftarrow s(a),\emptyset) \rightsquigarrow^{\varepsilon}_{Suc} \Delta \quad \text{for any } \Delta.$$

On the other hand, it does not flounder, since $\sim s(a)$ should be chosen in the goal $\leftarrow \sim q(x) \sim s(a)$. Note that $\sim s(a)$ can be deleted from the goal $\leftarrow \sim q(x) \sim s(a)$, when $s(a) \notin S_P(\overline{S_P(\emptyset)})$ such that $\leftarrow s(a)$ fails. However, if $s(a) \in S_P(\overline{S_P(\emptyset)})$, then we could not have got

$$(\{\leftarrow s(a)\},\{\sim s(a)\}) \rightsquigarrow_{Fail} \Delta'$$

for any Δ'. It is desirable to have the floundering derivation from $\leftarrow p(a)$ in this situation. For the floundering derivation, we present Definition 3.2.

We assume the definition of floundering in Definition 3.2. We can get the completeness of the presented procedure with respect to the 3-valued stable model semantics.

Theorem 5.3. *For a given general logic program P, assume that* $\Theta_P(\Delta_f^+) = \Delta_f^+$ *and* $S_P(\Delta_f^+) \cap \Delta_f^+ = \emptyset$. *If* $\{a_1, \ldots, a_l\} \subseteq \Delta_f^+$ *and* $\Delta_0 \subseteq \Delta_f$, *then* $(\{\leftarrow a_1, \ldots, \leftarrow a_l\}, \Delta_0) \rightsquigarrow_{Fail} \Delta_\alpha$ *for some* $\Delta_\alpha \subseteq \Delta_f$, *or there is some floundering derivation for* $\{\leftarrow a_1, \ldots, \leftarrow a_l\}$.

Proof. Assume that $\{a_1, \ldots, a_l\} \subseteq \Delta_f^+$, that is, $a_1, \ldots, a_l \notin S_P(\overline{S_P(\Delta_f^+)})$.

(1) In case that $Res_P(\{\leftarrow a_1, \ldots, \leftarrow a_l\}) \cap NegG = \emptyset$:

$$Fail(\{\leftarrow a_1, \ldots, \leftarrow a_l\}, \Delta_0, \emptyset, \Delta_0)$$

so that there is a failing derivation from $(\{\leftarrow a_1, \ldots \leftarrow a_l\}, \Delta_0)$. That is, $\{\leftarrow a_1, \ldots, \leftarrow a_l\}, \Delta_0) \rightsquigarrow_{Fail} \Delta_0$.

(2) In case that $Res_P(\{\leftarrow a_1, \ldots, \leftarrow a_l\}) \cap NegG \neq \emptyset$: For any $g \in Res_P(\{\leftarrow a_1, \ldots, \leftarrow a_l\}) \cap NegG$ such that $\leftarrow \sim B_1 \ldots \sim B_n = g - \Delta_0$, in the non-floundering derivation, there is some ground negative literal $\sim B_i$ such that

$$B_i \notin \overline{S_P(\Delta_f^+)},$$

because $a_1, \ldots, a_l \notin S_P(\overline{S_P(\Delta_f^+)})$. That is, $B_i \in S_P(\Delta_f^+)$ so that in the non-floundering derivation, there is some ground negative goal $g' \in Res_P(\{\leftarrow B_i\}) \cap NegG$ and

$$g' - \Delta_0 = \leftarrow \sim C_1 \ldots \sim C_m,$$

where $C_1, \ldots, C_m \in \Delta_f^+$. If some C_i contains some variable, there is a floundering derivation. In the non-floundering case, we have $Fail(\{\leftarrow a_1, \ldots, \leftarrow a_l\}, \Delta_0, F_1, \Delta_1)$ such that

$$\{\leftarrow C_1, \ldots, \leftarrow C_m\} \subseteq F_1 \quad \text{and} \quad \{\sim C_1, \ldots, \sim C_m\} \subseteq \Delta_1 \subseteq \Delta_f.$$

For this reason, we can construct

$$Fail(\{\leftarrow a_1, \ldots, \leftarrow a_l\}, \Delta_0, F_1, \Delta_1),$$
$$Fail(F_i, \Delta_i, F_{i+1}, \Delta_{i+1}) \ (i \geqslant 1),$$

where $\Delta_i \subseteq \Delta_f$ $(i \geqslant 1)$. Note that for $\leftarrow b \in F_i$, $b \in \Delta_f^+$. By induction, we have a sequence

$$(F_0, \Delta_0), \ (F_1, \Delta_1), \ldots \ .$$

Since the cardinality of Δ_f is at most ω, with good choices in the procedure *Fail*, we have a finite or infinite sequence such that

$$(\{\leftarrow a_1, \ldots, \leftarrow a_l\}, \Delta_0) \rightsquigarrow_{Fail} \Delta_\alpha \quad \text{for some } \Delta_\alpha \subseteq \Delta_f.$$

Because $S_P(\Delta_f^+) \cap \Delta_f^+ = \emptyset$, $S_P(\Delta_\alpha^+) \cap \Delta_\alpha^+ = \emptyset$. It follows from Lemma 4.5 that α could be at most ω. □

By Theorem 5.3, we have:

Corollary 5.4. *For a given general logic program P, assume that* $\Theta_P(\Delta_f^+) = \Delta_f^+$ *and* $S_P(\Delta_f^+) \cap \Delta_f^+ = \emptyset$. *If* $a \in \Delta_f^+$, *then* $(\{\leftarrow a\}, \emptyset) \rightsquigarrow_{Fail} \Delta_\alpha$ *for some* $\Delta_\alpha \subseteq \Delta_f$, *or there is some floundering derivation for* $\{\leftarrow a\}$.

Proof. Let $l = 1$ and $\Delta_0 = \emptyset$ in Theorem 5.3. Then we have this corollary. □

Theorem 5.5. *Assume for a given general logic program P that* $\Theta_P(\Delta_f^+) = \Delta_f^+$ *and* $S_P(\Delta_f^+) \cap \Delta_f^+ = \emptyset$. *If* $a \in S_P(\Delta_f^+)$, *then* $(\leftarrow a, \emptyset) \rightsquigarrow_{Suc}^{\varepsilon} \Delta_\gamma$ *for some* $\Delta_\gamma \subseteq \Delta_f$, *or there is some floundering derivation for* $\leftarrow a$.

Proof. Since $a \in S_P(\Delta_f^+)$, some ground negative goal $\leftarrow \sim c_1 \ldots \sim c_n$ is derivable from $\leftarrow a$ such that $\{c_1, \ldots, c_n\} \subseteq \Delta_f^+$, or there is a floundering derivation for $\leftarrow a$. By Theorem 5.3, for $\Delta_i \subseteq \Delta_{i+1} \subseteq \Delta_f$, we have

$$(\{\leftarrow c_i\}, \Delta_i) \rightsquigarrow_{Fail} \Delta_{i+1} \ (1 \leqslant i \leqslant n),$$

or there is some floundering derivation for $\{\leftarrow c_i\}$. Hence, the consecutive failing derivations can be included in a succeeding derivation. This concludes the proof. □

6. Concluding remarks

Based on the standpoint of taking the 3-valued stable model as a denotation, which is not always the least, we present a sound and complete procedure with respect to the model. The procedure is a concretization of abstract proof procedure in argumentation theory as in [18]. It might involve not only positive and negative loops for grounded programs as in [20], but also infinite derivations caused by both positive and negative recursions in non-floundering derivations even for non-ground programs.

We give some remarks of the presented procedure.

(1) Two kinds of choices are needed for the construction of failing derivations, while they are in relation with the floundering problems. As regards the implementations of the choice functions, it is better to have a way to exclude non-floundering ground negative literals (that is, negative subgoals) of each negative goal. Finite failure and/or infinite failure in failing derivations may be allowable, for us to erase the subgoals.

As shown in Example 5.2, we may encounter the goal like

$$\leftarrow\sim q(x) \sim s(a)$$

in some failing derivation for a general logic program P. $\sim s(a)$ should be chosen for non-floundering derivations in this case, however, $\leftarrow \sim q(x) \sim s(a)$ cannot fail, because $(\leftarrow s(a), \emptyset) \not\rightsquigarrow_{Suc}^{\varepsilon} \Delta$ for any Δ. If $s(a) \notin S_P(\overline{S_P(\emptyset)})$ by means of the mapping S_P,

then we can confirm that $\leftarrow s(a)$ fails so that $\sim s(a)$ can be erased from the goal $\leftarrow \sim q(x) \sim s(a)$ for further failing derivations. Unless $s(a) \notin S_P(\overline{S_P(\emptyset)})$, we cannot determine, without any other means, whether or not

$$(\{\leftarrow s(a)\}, \{\sim s(a)\}) \rightsquigarrow_{Fail} \Delta'$$

for some Δ'. In this case, we cannot cut $\sim s(a)$ off from the goal, to conclude that the goal flounders in the sense of floundering as introduced in [19]. That is, the exclusion of any ground negative literal from a goal, which is not needed for the failing derivation, is relevant to a floundering problem.

(2) The presented procedure takes a safeness condition for negation as failure, that is, just ground negative literals can be used for negation as failure, while the floundering problem is not avoidable. By taking the semantics as in [15, 24–26], we might have an extended version of the procedure with respect to the 3-valued non-ground stable model and the non-ground alternating fixpoint semantics.

(3) There are practical problems on the implementation of the present procedure. The infinite derivations of both positive and negative recursions are required as long as we are concerned with the completeness for the 3-valued stable model semantics. Selecting one ground negative literal in a negative goal, and selecting one derivation of a negative goal are means for the procedure to be well-defined inductively. In case of treating SLS resolution [21, 23] as an extension of SLDNF resolution with infinite failure and negative recursion, we can have XOLDTNF resolution [2], for the computation of the well-founded model, by making use of memoing of positive loop, the negative context to regard the negative literal as undefined, and bounded-term-size property for terminations. Chen and Warren [5] presents the methods of delaying the treatments for negative literals to be interpreted in a 3-valued model, as well as of terminating computations by bouded-term-size property. When we apply some variant of memoing techniques to our derivation, it will be an interesting problem to see some relations between bounded-term-size property and the termination of our procedure.

Acknowledgements

The authors are grateful to the referees regarding the revision of the paper.

References

[1] C.R. Baral, V.S. Subrahmanian, Dualities between alternative semantics for logic programming and nonmonotonic reasoning, J. Automat. Reason. 10 (1993) 399–420.
[2] W. Chen, D.S. Warren, A goal-oriented approach to computing well-founded semantics, J. Logic Programming 17 (1993) 279–300.
[3] W. Chen, D.S. Warren, Efficient top-down computation of queries under the well-founded semantics, J. Logic Programming 24 (1995) 161–199.
[4] W. Chen, D.S. Warren, Query evaluation in deductive databases with alternating fixpoint semantics, ACM Trans. Database Systems 20 (1995) 239–287.

[5] W. Chen, D.S. Warren, Tabled evaluation with delaying for general logic programs, J. ACM 43 (1996) 20–74.
[6] P.M. Dung, Negation as hypothesis: an abductive foundation for logic programming, Proc. 8th ICLP (1991) 3–17.
[7] P.M. Dung, On the relations between stable and well-founded semantics of logic programs, Theoret. Comput. Sci. 105 (1992) 7–25.
[8] P.M. Dung, An argumentation-theoretic foundation for logic programming, J. Logic Programming 22 (1995) 151–177.
[9] E. Eder, Properties of substitutions and unifiers, J. Symbolic Comput. 1 (1985) 31–46.
[10] K. Eshghi, R.A. Kowalski, Abduction compared with negation by failure, Proc. 6th ICLP (1989) 234–255.
[11] A. Van Gelder, The alternating fixpoint of logic programs with negation, J. Comput. System Sci. 47 (1993) 185–221.
[12] A. Van Gelder, K.A. Ross, J.S. Schlipf, The well-founded semantics for general logic programs, J. ACM 38 (1990) 620–650.
[13] M. Gelfond, V. Lifschitz, The stable model semantics for logic programs, Proc. 5th ICLP (1988) 1070–1080.
[14] L. Giordano, A. Martelli, M.L. Sapino, Extending negation as failure by abduction: a three-valued stable model semantics, J. Logic Programming 26 (1996) 31–67.
[15] G. Gottlob, S. Marcus, A. Nerode, G. Salzer, V.S. Subrahmanian, A non-ground realization of the stable and well-founded semantics, Theoret. Comput. Sci. 166 (1996) 221–262.
[16] A.C. Kakas, R.A. Kowalski, F. Toni, Abductive logic programming, J. Logic Comput. 2 (1992) 719–770.
[17] A.C. Kakas, P. Mancarella, Preferred extensions are partial stable models, J. Logic Programmimg 14 (1992) 341–348.
[18] A.C. Kakas, F. Toni, Computing argumentation in logic programming, J. Logic Comput. 9 (1999) 515–562.
[19] J.W. Lloyd, Foundations of Logic Programming, 2nd extended ed., Springer, Berlin, 1993.
[20] L.M. Pereira, N.A. Joaquim, J.J. Alferes, Derivation procedures for extended stable models, Proc. 12th IJCAI (1991) 863–868.
[21] T. Przymusinski, Every logic program has a natural stratification and an iterated least fixed point model, Proc. 8th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (1989) 11–21.
[22] T. Przymusinski, Well-founded semantics coincides with three-valued stable semantics, Fund. Inform. 13 (1990) 445–463.
[23] K.A. Ross, A procedural semantics for well-founded negation in logic programs, J. Logic Programming 13 (1992) 1–22.
[24] J.C. Shepherdson, A sound and complete semantics for a version of negation as failure, Theoret. Comput. Sci. 65 (1989) 343–371.
[25] S. Yamasaki, SLDNF resolution with non-safe rule and fixpoint semantics for general logic programs, Theoret. Comput. Sci. 160 (1996) 283–303.
[26] S. Yamasaki, Y. Kurose, Soundness of abductive proof procedure with respect to constraint for non-ground abducibles, Theoret. Comput. Sci. 206 (1998) 257–281.
[27] J.-H. You, L.Y. Yuan, On the equivalence of semantics for normal logic programs, J. Logic Programming 22 (1995) 211–222.