A Characterization of Stable Models using a Non-Monotonic Operator

Frank Teusink
Centre for Mathematics and Computer Science
P.O. Box 4079 1009 AB Amsterdam
The Netherlands
frankt@cwi.nl

Abstract

Stable models seem to be a natural way to describe the beliefs of a rational agent. However, the definition of stable models itself is not constructive. It is therefore interesting to find a constructive characterization of stable models, using a fixpoint construction. The operator we define, is based on the work of –among others– F. Fages. For this operator, every total stable model of a general logic program will coincide with the limit of some (infinite) sequence of interpretations generated by it. Moreover, the set of all stable models will coincide with certain interpretations in these sequences. Furthermore, we will characterize the least fixpoint of the Fitting operator and the well-founded model, using our operator.

1 Introduction

Stable models, as introduced in [GL88] and extended to three-valued models in [Prz90], seem to be a natural candidate for providing general logic programs with a meaning. However, their definition is not constructive. The aim of this paper is to find a constructive characterization of stable models for general logic programs, using sequences of interpretations generated by iterating a non-deterministic non-monotonic operator. The non-deterministic behaviour of this operator is captured by using the notion of selection strategies. Our operator is based on the ideas of F. Fages [Fag91]. The main difference with the approach of Fages is, that our operator is less non-deterministic than his. As a result, our operator is more complex, but this enables us to define a notion of (transfinite) fairness with which we can characterize a class of stabilizing strategies that contain all total stable models. Moreover, the additional structure in our operator allows us to define various classes of strategies with nice properties. The difference of our operator with respect to the *backtracking fixpoint* introduced by D. Saccà and C. Zaniolo in [SZ90] is twofold: we find all stable models, instead of only all total stable models, and, when an inconsistency occurs, we use a non-deterministic choice over

all possibilities for resolving that inconsistency, while their operator uses backtracking, which is just one particular possibility.

In the next section we give a short introduction on general logic programs and interpretations, and introduce some notations that will be used throughout the paper. Section 3 contains an explanation of (three-valued) well-supported models and stable models, and a generalization of Fages' Lemma, which establishes the equivalence between a subset of the set of (three-valued) well-supported models and the set of (three-valued) stable models. In section 4 we will introduce our operator $\mathcal{S}_P$, and prove that the sequences generated by this operator consist of well-supported interpretations. After this, we will show in sections 5, 6, 7 and 8 how to find total stable models, (three-valued) stable models, the least fixpoint of the Fitting operator and the well-founded model, respectively, using our operator. In section 9, we will take a short look at the complexity of the operator.

This paper also appears as a technical report [Teu93] at CWI. This technical report contains the full proofs of all theorems and lemmas presented in this paper.

2 Preliminaries and notations

A *general logic program* is a finite set of clauses $R : A \leftarrow L_1 \wedge \ldots \wedge L_k$, where A is an atom and L_i ($i \in [1..k]$) is a literal. A is called the *conclusion* of R, and $\{L_1, \ldots, L_k\}$ is called the *set of premises* of R. We write $concl(R)$ and $prem(R)$ to denote A and $\{L_1, \ldots, L_k\}$, respectively. For semantic purposes, a general logic program is equivalent to the (possibly infinite) set of ground instances of its clauses. In the following, we will only work with these infinite sets of ground clauses, and call them *programs*.

We use $\mathcal{B}_P$ to denote the Herbrand Base of a program P; A, A' and A_i represent typical elements of $\mathcal{B}_P$. Furthermore, $\mathcal{L}_P$ is the set of all literals of P; L, L' and L_i represent typical elements of $\mathcal{L}_P$. We use the following notations:

- for a literal L, $\neg L$ is the positive literal A, if $L = \neg A$, and the negative literal $\neg A$, if $L = A$, and
- for a set of literals S, we write
  - $\neg S$ to denote the set $\{\neg L \mid L \in S\}$,
  - $S^+ = \{A \mid A \in S\}$ to denote the set of all atoms that appear in positive literals of S,
  - $S^- = \{A \mid \neg A \in S\}$ to denote the set of all atoms that appear in negative literals of S, and
  - $S^\pm = S^+ \cup S^-$ to denote the set of all atoms that appear in literals of S.

A *two-valued* interpretation of a program P maps the elements of $\mathcal{B}_P$ on *true* or *false*. In this paper, we will use *three-valued* interpretations, in which an atom can also be mapped on *unknown*. They are defined as follows:

Definition 2.1 Let P be a program. An *interpretation* I of P is a set of elements from $\mathcal{L}_P$. An atom is *true* in I, if it is an element of I^+, it is *false* in I, if it is an element of I^-, and it is *unknown* in I, if it is not an element of $I^\pm$. If some atom is both *true* and *false* in I, then I is called *inconsistent*. If all atoms in $\mathcal{B}_P$ are either *true* or *false* (or both) in I, then I is called *total*. □

Example 2.2 Consider program P_1 consisting of the clauses $p(a) \leftarrow \neg p(b)$, $p(b) \leftarrow \neg p(a)$ and $q(b) \leftarrow q(b)$. We have that $\mathcal{B}_{P_1}$ is the set $\{p(a), p(b), q(a), q(b)\}$. There are $2^8 = 256$ interpretations of P_1, $3^4 = 81$ of them are consistent, $3^4 = 81$ of them are total, and $2^4 = 16$ of them are consistent and total.

Note, that a consistent total interpretation can be seen as a two-valued interpretation, because then no atom is both *true* and *false* and, because $I^\pm = \mathcal{B}_P$, there no atom is unknown.

3 Well-Supported and Stable Models

In this section we will introduce well-supported models and stable models. Our definition of well-supported models is an extension (to three-valued models) of the definition given in [Fag91]. Our definition of three-valued stable models follows the definition given in [Prz90]. First, we will introduce *well-supported models*, because they follow quite naturally from the intuitive idea of the meaning of a program. After this we will give the definition of *stable models*, which is quite elegant. In the remainder of this section we generalize of Fages' Lemma [Fag91] (which states that the class of total stable models and the class of total well-supported models coincide) to three-valued models.

So, let's take a look at the intuitive idea of the meaning of a program. First of all, an interpretation should be consistent; it doesn't make sense to have atoms that are both true and false. Furthermore, one can see a clause in a program as a statement saying that the conclusion of that clause should be true if that clause is applicable.

Definition 3.1 Let P be a program, let I be an interpretation of P and let R be a clause in P. R is *applicable* in I, if $prem(R) \subseteq I$. R is *inapplicable* in I, if $\neg prem(R) \cap I \neq \emptyset$. We call $\neg prem(R) \cap I$ the *blocking-set* of R in I. □

Now, a *model* of a program P is a consistent interpretation I of P such that, for every clause in P that is applicable in I, the conclusion of that clause is true in I and an atom is false in I only if all clauses with that atom as conclusion are inapplicable in I. Note, that we have to state explicitly that I has to be consistent, because in our definition an interpretation can be inconsistent.

In a model of P, atoms can be true, even when there is no reason for that atom being true. However, an atom should only be true, if there is some kind of "explanation" for the fact that that atom is true. This concept of "explanation" will be formalized using the notion of *support order*.

Definition 3.2 Let P be a program and let I be an interpretation of P. A partial order $<$ on the elements of $\mathcal{L}_P$ is a *support order* on I, if, for all $A \in I^+$, there exists a clause R in P with conclusion A such that R is applicable in I and, for all $A' \in prem(R)^+$, $A' < A$. □

Example 3.3 Consider a model $M = \{p(a), \neg p(b), q(b)\}$ of program P_1 (example 2.2). Any partial order in which $p(b) < p(a)$ and $q(b) < q(b)$ is a support order on M.

If, for some positive literal L that is true in M, we gather all literals L' such that $L' <^* L$ ($<^*$ is the transitive closure of $<$), then this set constitutes some kind of explanation for the fact that L is true in M.

Example 3.4 Consider program P_2 consisting of the clauses $p \leftarrow q \wedge r$, $q \leftarrow$ and $r \leftarrow \neg s$. One of the models of P_2 is $\{p, q, r, \neg s\}$, and $\{q < p, r < p\}$ is a support order on this model. We can read this support order as follows: p is true because r and q are true, q is always true, r is true because s is false, and s is false because there is no reason why s should be true.

However, such an explanation can be rather awkward, either because it refers to the conclusion itself, or because it contains an infinite number of literals.

Example 3.5 Consider program P_3 consisting of the clauses $p \leftarrow q$ and $q \leftarrow p$. One of the models of P_3 is $\{p, q\}$, and $\{p < q, q < p\}$ is a support order on this model. However, the explanation 'p is true because q is true and q is true because p is true', is not a meaningful explanation for the fact that p is true.

Example 3.6 Consider program P_4 consisting of the clauses $p(x) \leftarrow p(s(x))$ and $p(0) \leftarrow$. One of the models of P_4 is $\{p(s^i(0)) \mid i \geq 0\}$, and the partial order $\{p(s^i(0)) < p(s^{i+1}(0)) \mid i \geq 0\}$ is a support order on this model. However, any explanation for the fact that $p(0)$ is true in M_4, would be infinite. This seems to be rather counterintuitive.

Models for which every support order contains these cyclic or infinite explanations, should not be considered as giving a correct meaning to a program.

This can be achieved by using the fact that a support order is well-founded if and only if it doesn't contain cyclic or infinite explanations. Now, we can give the definition of *well-supported models*.

Definition 3.7 Let P be a program, and let M be a model of P. M is a *well-supported model* of P, if there exists a well-founded support order on M. □

Example 3.8 Consider the program P_1 (example 2.2). The interpretations $\{p(a), \neg p(b), \neg q(a), \neg q(b)\}$ and $\{p(a), \neg p(b), \neg q(a), \neg q(b)\}$ are well-supported models of P_1.

Another characterization of the meaning of a program is given by the definition of *stable models*. In the two-valued case, this definition uses the fact that the meaning of positive logic programs (in which the bodies of the clauses contain only positive literals) is well understood; it is given by the unique *two-valued minimal model* of the program. This definition of stable models has been generalized by T. Przymusinski to three-valued stable models [Prz90]. In this definition, he uses the notion of (three-valued) truth-minimal models, and a program transformation.

Definition 3.9 Let P be a positive program and let M be a model of P. M is a *truth-minimal model* of P, if there does not exist a model M' (other than M) of P such that $M'^{+} \subseteq M^{+}$ and $M'^{-} \supseteq M^{-}$. □

Definition 3.10 Let P be a program and let I be an interpretation of P. The program $\frac{P}{I}$ is obtained from P by replacing every negative literal L in the body of a clause in P that is true (resp. false; resp. unknown) in I by the proposition **t** (resp. **f**; resp. **u**). □

Now, we are able to give the definition of a *stable model*.

Definition 3.11 Let P be a program and let M be an interpretation of P. M is a *stable model* of P, if M is a truth-minimal model of $\frac{P}{M}$. □

Example 3.12 Consider the program P_1 (example 2.2), and the model $\{p(a), \neg p(b), \neg q(a), \neg q(b)\}$ of P_1. M is a stable model of P_1, because it is a truth-minimal model of the program $\frac{P_1}{M} = \{p(a) \leftarrow \mathbf{t}, p(b) \leftarrow \mathbf{f}, q(b) \leftarrow q(b)\}$.

The following lemma shows that the class of stable models coincides with a subclass of the well-supported models. This lemma is an generalization of the lemma by F. Fages [Fag91], which proves that two-valued stable models and two-valued well-supported models coincide. The proof we give, closely follows the proof given by F. Fages. First, we have to introduce the notion of *(greatest) unfounded set*.

Definition 3.13 Let P be a program and let I be an interpretation of P. Let S be a subset of $\mathcal{B}_P - I^{\pm}$. S is an *unfounded set* of I, if all clauses R in P such that $concl(R) \in S$ are inapplicable in $I \cup \neg S$. The *greatest unfounded set* $U_P(I)$ of I is the union of all unfounded sets of I. □

Lemma 3.14 (Equivalence) *Let P be a program and let M be an interpretation of P. M is a stable model of P iff M is a well-supported model of P such that $U_P(M) = \emptyset$.*

Proof (sketch):
($\Rightarrow$) Let M be a stable model of P. We can find M^+ by applying the immediate consequence operator on $\frac{P}{M}$. Using this operator, we can define an order on M^+. This order is a well-founded support order. Therefore, M is a well-supported model of P.
($\Leftarrow$) Let M be a well-supported model of P such that $U_P(M)$ is empty. M is a model of P and therefore M is a model of $\frac{P}{M}$. Now, by the fact that M is a well-supported model of P, there does not exists a model M' of $\frac{P}{M}$ such that $M'^+ \subset M^+$ and $M'^- \supseteq M^-$, and by the fact that $U_P(M)$ is empty, there does not exists a model M' of $\frac{P}{M}$ such that $M'^+ \subseteq M^+$ and $M'^- \supset M^-$. Therefore, there does not exists a model M' of $\frac{P}{M}$ such that $M'^+ \subseteq M^+$ and $M'^- \supseteq M^-$. Thus M is a stable model of P. □

4 The operator $\mathcal{S}_P$

In this section, we define the operator $\mathcal{S}_P$. This operator is inspired on the operator J_P^a of Fages, but there are some major differences.

The idea is, to generate all *total* stable models of a program, by starting from the empty interpretation. At each step, we try to extend an interpretation I to a new interpretation I', that brings us "nearer" to a total stable model. For this, we use the following strategies:

1. If there exists a clause R that is applicable in I and $concl(R)$ is not an element of I, then we add $concl(R)$ to I (after all, we are looking for a model).

2. If there exists an atom A such that all clauses R that have A as conclusion, are inapplicable in I, and $\neg A$ is not an element of I, then we add $\neg A$ to I (after all, we are working towards a total interpretation).

3. If the previous two strategies fail, we can do little more that blindly select an atom from $\mathcal{B}_P - I^{\pm}$, and add it, or its negation, to I. However, in contrast with the two previous strategies, this strategy is flawed, in the sense that, even when I is a subset of some stable model, I' is not guaranteed to be a subset of a stable model. In fact, continuing the procedure with I' can lead to an inconsistent interpretation.

4. If I is inconsistent, then we should try to find a consistent interpretation I'. However, we don't want to throw away I completely. We know that the inconsistency was caused by some literal chosen by strategy 3. We will maintain "possible reasons for inconsistency" with our interpretation, in order to identify a literal in I that could be the reason for the inconsistency, and find a new consistent interpretation I' by removing from I all literals that were added to the interpretation due to the presence of this literal.

Note, that with all four strategies one could have more than one way to generate the next interpretation. For example, if there are two reasons for the inconsistency of an interpretation, there are two possibilities for resolving that inconsistency. As a result, our operator will be non-deterministic.

We have to maintain "reasons for inconsistency" with our interpretation. Moreover, we will maintain a support order with our interpretation, to help us prove various properties. This leads to the following definition of *j-interpretations*.

Definition 4.1 A *j-triple*, is a triple $\langle L, \tau, \psi \rangle$, such that L is an element of $\mathcal{L}_P$, and τ and ψ are subsets of $\mathcal{L}_P$. A *j-interpretation* J of P is a set of j-triples such that for every literal in $\mathcal{L}_P$, J contains at most one j-triple with that literal as the first element. We call τ the *support-set* of L and ψ the *culprit-set* of L. For a set S of j-triples, we will use $\overline{S}$ to denote the set of literals $\{L \mid \langle L, \tau, \psi \rangle \in S\}$. □

Note, that our support-set differs from the justification in a justified atom of Fages, because it can be infinite, and it is defined on literals instead of atoms. Moreover, our support-set is intended to contain a set of premises for a positive literal, and a set of elements of blocking-sets for negative literals, whereas the justifications of Fages contain a complete explanation for the fact that an atom is true. Using the support-sets in a j-interpretation J, we can define a partial order on the literals in $\overline{J}$.

Definition 4.2 Let J be a j-interpretation. We define $<_J$ to be the partial order such that $A' <_J A$ iff $\langle A, \tau, \psi \rangle \in J$ and $A' \in \tau^+$ (note, that A is a positive literal). □

In the interpretations on which S_P will operate, the culprit-set will contain the "possible reasons for inconsistency" and the partial order $<_J$ will be a support order on $\overline{J}$.

In the definition of the operator S_P, we will use the *conflict-set*, *choice-set* and *culprit-set* of a j-interpretation J. The *conflict-set* of a j-interpretation J contains j-triples for every literal L for which there are one or more reasons for adding them to J, according to strategies 1 and 2.

Definition 4.3 Let P be a program and let J be a j-interpretation of P. The *conflict-set* $Conflict_P(J)$ of J is the set of j-triples $\langle L, \tau, \psi \rangle$ such that

- $L \notin \overline{J}$,
- if $L = A$, then there exists a clause R in P with conclusion A that is applicable in $\overline{J}$ such that $\tau = prem(R)$,
- if $L = \neg A$, then every clause R in P with conclusion A is inapplicable in $\overline{J}$, and for every clause R in P with conclusion A exists a literal L_R in the blocking-set of R in $\overline{J}$ such that $\tau = \{L_R \mid R \in P \wedge concl(R) = A\}$, and
- $\psi = \bigcup\{\psi' \mid \langle L', \tau', \psi' \rangle \in J \wedge L' \in \tau\}$.

□

For a j-triple $\langle L, \tau, \psi \rangle$ in $Conflict_P(J)$, τ contains the reason for adding L to J, and ψ contains all literals that could be the cause of L being an element of $\overline{Conflict_P(J)}$, while $\neg L$ is an element of $\overline{J}$.

The *choice-set* of J contains j-triples that could be added to J on behalf of strategy 3. The support-sets and choice-sets of these j-triples reflect the fact that there is no real support for adding these literals to J.

Definition 4.4 Let P be a program and let J be a j-interpretation of P. The *choice-set* $Choice_P(J)$ of P is the set

$$\{\langle L, \emptyset, \{L\} \rangle \mid L \in \neg(\mathcal{B}_P - \overline{J^{\pm}})\}$$

□

The *culprit-set* of an inconsistent j-interpretation J, is the set of all "possible reasons for inconsistency"; that is, the set of literal that are common to the culprit-sets of all literals L in $\overline{J}$ whose negation $\neg L$ is also an element of $\overline{J}$.

Definition 4.5 Let P be a program and let J be a j-interpretation of P. The *culprit-set* $Culprit_P(J)$ of J is the set

$$\bigcap\{\psi \cup \psi' \mid \langle A, \tau, \psi \rangle \in J \wedge \langle \neg A, \tau', \psi' \rangle \in J\}$$

□

Note, that if $\overline{J}$ is consistent then $Culprit_P(J) = \emptyset$. We are now capable of defining our operator S_P.

Definition 4.6 For a general logic program P, we define the operator S_P as follows:

$$S_P(J) = \begin{cases} J - \{\langle L, \tau, \psi \rangle \mid \rho_1 \in \psi\} & \text{, if } Culprit_P(J) \neq \emptyset \\ J \cup \{\rho_2\} & \text{, if } Conflict_P(J) \neq \emptyset \\ J \cup \{\rho_3\} & \text{, if } Choice_P(J) \neq \emptyset \\ J & \text{, otherwise} \end{cases}$$

where $\rho_1 \in Culprit_P(J)$
$\rho_2 \in Conflict_P(J)$
$\rho_3 \in Choice_P(J)$

□

Note, that in this definition the order of the conditions is relevant (i.e. a rule is only applied if its condition is satisfied *and* the conditions of all previous rules failed).

The operator as we defined it, is non-deterministic, in the sense that it non-deterministically chooses an element (ρ_1, ρ_2 or ρ_3) from a set of candidates. Because we want to manipulate this non-deterministic behaviour, we extend the operator with a *selection strategy*, that encapsulates this non-deterministic behaviour of $\mathcal{S}_P$.

Definition 4.7 Let P be a program. A *selection strategy* ρ for P is a non-deterministic function that, for a j-interpretation J of P, chooses ρ_1 among $Culprit_P(J)$, ρ_2 among $Conflict_P(J)$ and ρ_3 among $Choice_P(J)$. □

Note, that ρ can be deterministic if we consider more information. For instance, we could use a selection strategy that bases its choices for some j-interpretation J on the way in which J was generated (i.e. previous applications of $\mathcal{S}_P$). We will use the notation $\mathcal{S}_P^\rho$ to indicate that we are using the operator on a program P with a selection strategy ρ for P.

As said before, we want to find a stable model for P by starting from the empty interpretation. In order to do this, we have to define the (ordinal) powers of $\mathcal{S}_P^\rho$.

Definition 4.8 Let P be a program and let ρ be a selection strategy for P. Let $\mathcal{S}_P^\rho$ be the operator as defined. we define the powers of $\mathcal{S}_P^\rho$ inductively:

$$\mathcal{S}_P^\rho \uparrow^\alpha = \begin{cases} \emptyset & \text{, if } \alpha = 0 \\ \mathcal{S}_P^\rho(\mathcal{S}_P^\rho \uparrow^{\alpha-1}) & \text{, if } \alpha \text{ is a successor ordinal} \\ \bigcup_{\beta<\alpha} \bigcap_{\beta \leq \gamma < \alpha} \mathcal{S}_P^\rho \uparrow^\gamma & \text{, if } \alpha \text{ is a limit ordinal} \end{cases}$$

□

The definition for zero and successor ordinals are quite standard. The definition for limit ordinal is the same as the one used by Fages; it states that at a limit ordinal α, we retain only the j-triples that where persistent in the preceding sequence of j-interpretations; that is, for every j-triple in $\mathcal{S}_P^\rho \uparrow^\alpha$, there exists an ordinal β smaller that α, such that, for all $\gamma \in [\beta..\alpha)$, this j-triple is an element of $\mathcal{S}_P^\rho \uparrow^\gamma$.

Using the powers of $\mathcal{S}_P^\rho$, we define the following infinite sequence of j-interpretations.

Definition 4.9 Let P be a program and let ρ be a selection strategy for P. The *sequence for P and ρ* is the infinite sequence of j-interpretations $\Gamma_P^\rho \equiv J_0, \ldots, J_\alpha, \ldots$, where $J_\alpha = \mathcal{S}_P^\rho \uparrow^\alpha$, for all ordinals α. □

We will now work towards a proof of the fact that certain fixpoints of $\mathcal{S}_P$ are stable models of P. First, we have to prove that the application of $\mathcal{S}_P$ on a j-interpretation results in a j-interpretation, and that every element of a sequence is a j-interpretation.

Lemma 4.10 *Let P be a program and let ρ be selection strategy for P. If J is a j-interpretation, then $\mathcal{S}_P^\rho(J)$ is a j-interpretation.*

The proof of this lemma follows directly from the definition of $\mathcal{S}_P$ and is therefore omitted.

Lemma 4.11 *Let Γ_P^ρ be a sequence for a program P. Every element J_α of Γ_P^ρ is a j-interpretation of P.*

The proof of this lemma is by induction on α, and fairly straightforward.

We will now prove that for every j-interpretation J_α in a sequence Γ_P^ρ, the partial order $<_{J_\alpha}$ is a support order and a well-founded order.

Theorem 4.12 (Supportedness) *Let Γ_P^ρ be a sequence for a program P. For every J_α in Γ_P^ρ, the partial order $<_{J_\alpha}$ is a support order on $\overline{J}_\alpha$.*

Proof (sketch): The proof uses induction on α. If α is a successor ordinal, then we construct a support order $<_{J_\alpha}$, using the support order $<_{J_{\alpha-1}}$. If α is a limit ordinal, we use the fact that for all β smaller than α a support order $<_{J_\beta}$ exists, the fact that every j-literal in J_α was persistent in the preceding sequence of j-interpretations, and the fact that if a j-triple $\langle L, \tau, \psi \rangle$ is an element J_α, then for all $L' \in \tau$ there exists a j-triple $\langle L', \tau', \psi' \rangle$ in J_α. □

Theorem 4.13 (Well-Foundedness) *Let Γ_P^ρ be a sequence for a program P. For every J_α in Γ_P^ρ, the partial order $<_{J_\alpha}$ is well-founded.*

Proof: Suppose that $<_{J_\alpha}$ is not well-founded. Then, there exists an infinite decreasing chain $\ldots <_{J_\alpha} A_2 <_{J_\alpha} A_1 <_{J_\alpha} A_0$. Because $A_i \in \overline{J}_\alpha^+$, there exists a least ordinal β_i such that $\beta_i \leq \alpha$ and for some τ_i and ψ_i, for all $\gamma \in [\beta_i..\alpha]$, $\langle A_i, \tau_i, \psi_i \rangle \in J_\gamma$. Also, because $A_{i-1} \in \overline{J}_\alpha^+$, there exists a least ordinal β_{i-1} such that $\beta_{i-1} \leq \alpha$ and for some τ_{i-1} and ψ_{i-1}, for all $\gamma \in [\beta_{i-1}..\alpha]$, we have that $\langle A_{i-1}, \tau_{i-1}, \psi_{i-1} \rangle \in J_\gamma$. Furthermore, we have that $A_i <_{J_\alpha} A_{i-1}$, which implies that $A_i \in \tau_{i-1}$, and therefore $\beta_i < \beta_{i-1}$. As a result, we have that $\ldots < \beta_2 < \beta_1 < \beta_0$ is an infinite decreasing chain. But the $<$ order on ordinals is well-founded. Thus, the assumption that $<_{J_\alpha}$ is not well-founded is in contradiction with the fact that the $<$ order on ordinals is well-founded. Therefore, we can conclude that $<_{J_\alpha}$ is well-founded. □

We will now show that all fixpoints of $\mathcal{S}_P$ that appear in sequences are consistent.

Lemma 4.14 *Let Γ_P^ρ be a sequence for a program P. Let J_α be an element of Γ_P^ρ. If $\overline{J}_\alpha$ is inconsistent, then $\overline{J}_{\alpha+1}$ is consistent.*

Proof (sketch): The actual proof is rather long, because it involves proving two auxiliary lemmas. Therefore, we will do with a short sketch of the proof. If some $\overline{J}_\alpha$ is inconsistent, then there exists exactly one A such that both A and $\neg A$ are elements of $\overline{J}_\alpha$. By the definition of $\mathcal{S}_P$, we have that $\overline{J}_{\alpha+1}$

is consistent if $Culprit_P(J)$ is non-empty. This is true if one of the culprit-sets of A and $\neg A$ is non-empty. We then show that the inconsistency of $\overline{J}_\alpha$ implies that at least one of the two culprit-sets is non-empty. □

Theorem 4.15 (Fixpoint Consistency) *Let Γ_P^ρ be a sequence for a program P. Let J_α be an element of Γ_P^ρ. If J_α is a fixpoint of $\mathcal{S}_P$, then $\overline{J}_\alpha$ is consistent.*

Proof: Suppose $\overline{J}_\alpha$ is inconsistent. Then, by lemma 4.14, $J_{\alpha+1}$ is consistent. But then $J_\alpha \neq J_{\alpha+1}$. This is in contradiction with the fact that J_α is a fixpoint of $\mathcal{S}_P$. □

5 Total stable models as limit fixpoint of $\mathcal{S}_P$

We will now take a look at the fixpoints of $\mathcal{S}_P$ that appear in the sequence of P (we will call them *limit fixpoints*), and prove that they are the total stable models of P. First, we have to define the class of sequences that will contain a fixpoint: *stabilizing sequences*.

Definition 5.1 A sequence Γ_P^ρ is *stabilizing*, if there exists an ordinal α, such that, for all ordinals β greater than α, $J_\alpha = J_\beta$. The *closure ordinal* of Γ_P^ρ is the least ordinal α, such that, for all ordinals β greater than α, $J_\alpha = J_\beta$. □

Definition 5.2 Let P be a program. A j-interpretation J is a *limit fixpoint* of $\mathcal{S}_P$, if there exists a selection strategy ρ for P, such that the sequence Γ_P^ρ is stabilizing and $J = J_\alpha$, where α is the closure ordinal of Γ_P^ρ. □

Theorem 5.3 *Let P be a program. If J is a limit fixpoint of $\mathcal{S}_P$, then $\overline{J}$ is a total stable model of P.*

Proof: J is a limit fixpoint of $\mathcal{S}_P$. Therefore, there exists a selection strategy ρ such that Γ_P^ρ is stabilizing and $J = J_\alpha$, where α is the limit ordinal of Γ_P^ρ. By the Fixpoint Consistency Theorem (4.15), $\overline{J}_\alpha$ is consistent. By the construction of $\mathcal{S}_P$ and the fact that $J_\alpha = J_{\alpha+1}$, $\overline{J}_\alpha$ is a total model of P. Also, by the Supportedness Theorem (4.12) and the Well-Foundedness Theorem (4.13), $<_{J_\alpha}$ is a well-founded support order for $\overline{J}_\alpha$. Therefore, $\overline{J}$ is a total well-supported model of P. Because $\overline{J}$ is total, $U_P(\overline{J})$ is empty. From the Equivalence Lemma (3.14), we conclude that $\overline{J}$ is a total stable model of P. □

So, the limit fixpoints of $\mathcal{S}_P$ are total stable models of P. We will now show the converse: every total stable model is a limit fixpoint of $\mathcal{S}_P$. We define, for every stable model M of P, a class of selection strategies ρ such that M is contained in Γ_P^ρ.

Definition 5.4 Let P be a program and let M be a stable model of P. A *selection strategy for* M is a selection strategy that, for all J such that $\overline{J} \subset M$, selects a j-triple $\langle L, \tau, \psi \rangle$ from $Conflict_P(J)$ or $Choice_P(J)$ such that $L \in M$. □

Lemma 5.5 *Let P be a program and let M be a stable model of P. Then, there exists a selection strategy ρ for M and for some J_α in Γ_P^ρ, $M = \overline{J_\alpha}$.*

Proof (sketch): We first prove by inspection of the definition of $\mathcal{S}_P$ that, for an arbitrary stable model M of P and an arbitrary j-interpretation J of P such that $\overline{J} \subset M$, there exists a selection strategy ρ for P such that $\overline{\mathcal{S}_P^\rho(J)} \subseteq M$. From this we can conclude that there exists a selection strategy ρ for M. We then proceed by proving by induction on α that if α is the least ordinal such that, for $J_\alpha \in \Gamma_P^\rho$, $\overline{J_\alpha} \not\subset M$, then $\overline{J_\alpha} = M$. □

Theorem 5.6 (Characterization) *Let P be a program. The limit fixpoints of $\mathcal{S}_P$, coincide with the total stable models of P.*

Proof: We have from theorem 5.3 that all limit fixpoints of $\mathcal{S}_P$ contain stable models of P. Also, by lemma 5.5, there exists for every (total) stable model M of P a selection strategy ρ such that M is contained in an element of Γ_P^ρ. Because M is total, it follows that M is a limit fixpoint of $\mathcal{S}_P$. □

6 A characterization of stable models, using $\mathcal{S}_P$

In this section, we characterize the stable models of a program P, using our operator $\mathcal{S}_P$. As we have seen, the total stable models coincide with the limit fixpoints of $\mathcal{S}_P$. This means that we cannot characterize the set of all three-valued stable models as a set of fixpoints of $\mathcal{S}_P$. Instead, we identify the set of stable models of a program with some set of j-interpretations appearing in the sequences for that program.

Lemma 6.1 *Let P be a program and let M be an interpretation of P. M is a stable model of P iff there exists a j-interpretation J in a sequence for P, such that $M = \overline{J}$, $\overline{J}$ is consistent, $Conflict_P(J) = \emptyset$ and $U_P(\overline{J}) = \emptyset$.*

Proof (sketch):
($\Leftarrow$) Let J be an element of a sequence for P such that $\overline{J}$ is consistent, $\overline{Conflict_P(J)} = \emptyset$ and $U_P(\overline{J}) = \emptyset$. By the Supportedness Theorem (4.12) and the Well-Foundedness Theorem (4.13), $\overline{J}$ is a well-supported interpretation of P. Also, we know that $\overline{J}$ is consistent and that $U_P(\overline{J}) = \emptyset$. Because $Conflict_P(J) = \emptyset$, we know that $\overline{J}$ is a model of P. Finally, by the Equivalence Lemma (3.14), $\overline{J}$ is a stable model of P.
($\Rightarrow$) Let M be a stable model of P. By lemma 5.5, there exists a strategy ρ such that there exists an element J of Γ_P^ρ where $M = \overline{J}$. Clearly, M is consistent. So, we only have to prove that $Conflict_P(J) = \emptyset$ and that

$U_P(\overline{J}) = \emptyset$. The proofs of $Conflict_P(J) = \emptyset$ and $U_P(\overline{J}) = \emptyset$ are both based on the fact that M is a truth-minimal model of $\frac{P}{M}$. □

7 Relating the fixpoint of the Fitting operator to the sequences for P

In the operator S_P, we have a preference for using elements of $Conflict_P$ to extend an interpretation. The definition of $Conflict_P$ bares resemblance to the sets T_P and F_P used by the Fitting operator [Fit85]. We can identify the least fixpoint of the Fitting operator Φ_P with a special j-interpretation that appears in every sequence for P (in fact, it is the last element of the maximal prefix shared by all sequences for P). First, we give a definition of the Fitting operator.

Definition 7.1 Let P be a program. The Fitting operator Φ_P is defined as follows:

$$\begin{array}{ll} \Phi_P(I) = T_P(I) \cup F_P(I) & \\ \text{where} & T_P(I) = \{A \mid \exists_{R \in P} concl(R) = A \wedge prem(R) \subseteq I\} \\ & F_P(I) = \{\neg A \mid \forall_{R \in P} concl(R) = A \rightarrow \neg prem(R) \cap I \neq \emptyset\} \end{array}$$

□

The powers of the Fitting operator can be defined in the same way as we did for S_P. Although the definition of Fitting differs in the case of limit ordinals, we can safely use our definition, because Φ_P is monotone, and for monotone operators both definitions coincide.

Lemma 7.2 *Let Γ_P^ρ be a sequence for a program P. Let α be the least ordinal such that $Conflict_P(J_\alpha) = \emptyset$. Then, $\overline{J}_\alpha$ is the least fixpoint of the Fitting operator Φ_P.*

Proof: Let M be the least fixpoint of Φ_P. We have that $M = \Phi \uparrow^\phi (\emptyset)$, where ϕ is the closure ordinal of Φ_P. We will prove that $\overline{J}_\alpha \subseteq M$ and $\overline{J}_\alpha \supseteq M$.

1. We will prove by induction on β that if $\beta \leq \alpha$ then $\overline{J}_\beta \subseteq M$. For $J_0 = \emptyset$, the lemma holds trivially. Assume that for all $\gamma < \beta \leq \alpha$, $\overline{J}_\gamma \subseteq M$.

   If β is a successor ordinal, we have that $J_\beta = J_{\beta-1} \cup \{\langle L, \tau, \psi \rangle\}$. By induction hypothesis, we have that $\overline{J}_{\beta-1} \subseteq M$. Also, by the definition of $Conflict_P(J)$ and Φ_P, we have that $\overline{Conflict_P(J_{\beta-1})} \subseteq M$. Therefore, $\overline{J}_\beta \subseteq M$.

   If β is a limit ordinal, we have, because $\beta \leq \alpha$, that $J_\beta = \bigcup_{\gamma<\beta} J_\gamma$. By induction hypothesis, we have that $\overline{J}_\gamma \subseteq M$, for all $\gamma < \beta$. Therefore, $\overline{J}_\beta \subseteq M$.

2. We have to prove that $\overline{J}_\alpha \supseteq M$. It is enough to prove that $L \notin \overline{J}_\alpha$ implies that $L \notin M$. Suppose $L \notin \overline{J}_\alpha$. There are two cases:

   - L is positive.
     By definition of S_P and the fact that $Conflict_P(J_\alpha) = \emptyset$, we know that all clauses with conclusion L are not applicable in $\overline{J}_\alpha$. Therefore, by the definition of Φ_P, $L \notin T_P(M)$. As a result, we have that $L \notin M$, because $M = \Phi_P(M) \supseteq T_P(M)$.
   - L is negative.
     By definition of S_P and the fact that $Conflict_P(J_\alpha) = \emptyset$, we know that there exists a clause R in P with conclusion $\neg L$ such that $\neg prem(R) \cap \overline{J}_\alpha = \emptyset$. By this and the definition of Φ_P we have that $L \notin F_P(M)$, and therefore $L \notin M$.

□

8 Finding the Well-Founded Model using S_P

Although the well-founded model, as introduced in [GRS91], is a stable model, and therefore can be found using the results in section 6, we want to give special consideration to this model, because it is one of the most interesting stable models (together with the total stable models). In this section, we will show that the well-founded model of a program can be found using a special class of selection strategies, the *well-founded strategies*. First, we will give a definition of the well-founded model (for a proper definition, we refer to [GRS91]).

Definition 8.1 Let P be a program. The *well-founded model* of P is the smallest stable model of P (with respect to the knowledge ordering). □

Now, we introduce the class of *well-founded strategies*.

Definition 8.2 Let P be a program. A selection strategy ρ for P is a *well-founded strategy*, if, for all J such that ρ has to select an element of $Choice_P(J)$ and $U_P(\overline{J}) \neq \emptyset$, ρ selects an element of $U_P(\overline{J})$. □

Lemma 8.3 *Let Γ_P^ρ be the sequence for a program P and a well-founded selection strategy for P. Let α be the least ordinal such that $\overline{J}_\alpha$ is a stable model of P. Then $\overline{J}_\alpha$ is the well-founded model of P.*

Proof: Let M be the well-founded model for P. We have to prove that every well-founded strategy is a selection strategy for M. Let J be a j-interpretation such that $\overline{J} \subset M$. Clearly, $\overline{J}$ is consistent. If $Conflict_P(J)$ is non-empty, we can select an arbitrary element from $Conflict_P(J)$. (See lemma 5.5). So, suppose that $Conflict_P(J)$ is empty. Then, because $\overline{J} \subset M$, $Conflict_P(J)$ has to be non-empty. Clearly, $U_P(\overline{J}) \subseteq Conflict_P(J)$. So, to

prove that every well-founded strategy is a selection strategy for M, we have to prove that $U_P(J)$ is non-empty. Now, suppose that $U_P(J$ is empty. Then, $\overline{J}$ is smaller than M in the truth-ordering and $\overline{J}$ is a stable model of P. But this is in contradiction with the fact that M is the well-founded model of P. □

9 On the complexity of S_P

The fact that we can generate all stable models as limits of sequences of interpretations, does not mean that we are in general capable of finding them in finite time. M. Fitting has already shown in [Fit85] that the closure ordinal of his operator Φ_P could be as high as Church-Kleene ω_1, the first nonrecursive ordinal. Because our operator in some sense 'encapsulates' the Fitting operator, we cannot hope to do better with our operator. It would be interesting to define classes of programs whose stable models can be generated in an "acceptable" amount of time.

The first class of programs that comes to mind, is the class of programs P whose Herbrand Base $\mathcal{B}_P$ is finite. The following result is similar to the results obtained in [Fag91] and [SZ90]. First, we have to define a class of selection strategies whose sequences are guaranteed to be stabilizing.

Definition 9.1 Let P be a program and let ρ be a selection strategy for P. We call ρ *fair* if, for all ordinals α and all ordinals β smaller than α, $J_\alpha = J_\beta$ implies that the selection made by ρ for J_α differs from the selection made by ρ for J_β. □

Lemma 9.2 *Let P be a program. If ρ is a fair strategy for P, then the sequence Γ_P^ρ is stabilizing.*

Proof: Suppose there exists a fair strategy ρ such that Γ_P^ρ is not stabilizing. Then, we have that, for all ordinals α, $J_\alpha \neq J_{\alpha+1}$. Because J_α is defined for all ordinals α, there exists at least one j-interpretation J, such that for any ordinal α, there exists an ordinal β such that $\beta > \alpha$ and $J_\beta = J$. This j-interpretation J has a set C associated with it, from which ρ makes a selection (C is one of $Culprit_P(J)$, $Conflict_P(J)$ and $Choice_P(J)$). This set C is non-empty, because otherwise we would have that $J = S_P^\rho(J)$, and is countable (but possibly infinite), because $\mathcal{B}_P$ is countable. Because ρ is fair, we have that for any two j-interpretations J_α and J_β in Γ_P^ρ such that $J_\alpha = J_\beta$ and $\alpha \neq \beta$, the element selected by ρ for J_α differs from the element selected by ρ for J_β. Therefore, there exists an ordinal γ after which every element of C has been selected once for J. But we know that there exists an ordinal δ such that $\delta > \gamma$ and $J = J_\delta$. At that point, ρ cannot make a fair selection. This is in contradiction with the fact that ρ is a fair selection rule. Therefore, if ρ is fair then Γ_P^ρ is stabilizing. □

Lemma 9.3 *Let P be a program with a finite Herbrand base $\mathcal{B}_P$. Let ρ be a fair strategy for P. The closure ordinal of the sequence Γ_P^ρ is finite.*

Proof: First, note that by lemma 9.2 Γ_P^ρ is stabilizing, and that therefore it has a closure ordinal. Because $\mathcal{B}_P$ is finite, the number of j-interpretations is finite. Furthermore, for any j-interpretation J, the sets $Conflict_P(J)$, $Choice_P(J)$ and $Culprit_P(J)$ are finite. Because of this and the fact that ρ is fair, any j-interpretation J that is not the limit fixpoint of Γ_P^ρ will occur only finitely many times in Γ_P^ρ. As a result, we have that the closure ordinal of Γ_P^ρ is finite. □

Note, that this result is not very surprising. If $\mathcal{B}_P$ is finite, the set of interpretations for P is finite, which means that one can simply enumerate the set of all interpretations of P and test which of them are stable models of P. Thus, any operator should be capable of finding a solution in finite time in this case.

There remains the question of what is the best method for finding stable models of programs in the case of finite Herbrand Bases; generating and testing all consistent interpretations of a program or using S_P with some carefully chosen family of selection strategies. We have good hope, that the second option will, in general, perform better than the first option. First of all, by inducing some order on the atoms in the Herbrand Base of a program, like Saccà and Zaniolo did with their backtracking operator in [SZ90], we can restrict ourselves to a family of 'ordered' selection strategies, in which the redundancy in partial interpretations being considered is greatly reduced (though not eliminated completely). Moreover, although in general the number of well-supported partial interpretations of a program can be greater than the number of consistent total interpretations of a program, we think that in the typical case the number of well-founded interpretations taken into consideration by S_P when using a family of ordered selection strategies will be much smaller. To reinforce this claim, we will have to take a closer look at these ordered selection strategies and implement the operator to experiment with it.

10 Conclusion

In this paper, we have presented an operator that generates sequences of interpretations. We have shown that the limits of these sequences are exactly all total stable models of a general logic program. Moreover, the set of all stable models can be identified as a subset of the interpretations generated by the operator. Furthermore, we have shown that the least fixpoint of the Fitting operator appears in all sequences generated by our operator, and that we can find the well-founded model, using a class of special selection strategies.

It would be interesting to find classes of selection strategies that can be implemented efficiently, are complete (i.e. are capable of finding all (total) stable models), and have small closure ordinals. The class of ordered strategies seems to be a good candidate, and it might be possible that we are capable of restricting this class further.

Acknowledgements

This paper was partially supported by a grant from SION, a department of NWO, the National Foundation for Scientific Research. I would like to thank Krzysztof Apt for his support, proof reading and giving valuable suggestions for improving the paper. Also, I would like to thank Elena Marchiori for proof reading the paper.

References

[Fag91] Francois Fages. A new fixpoint semantics for general logic programs compared with the well-founded and the stable model semantics. *New Generation Computing*, 9:425–443, 1991.

[Fit85] Melvin Fitting. A kriptke-kleene semantics for logic programs. *Journal of Logic Programming*, 2(4):295–312, 1985.

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proceedings on the Fifth International Conference and Symposium on Logic Programming*, pages 1070–1080, 1988.

[GRS91] Allen van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the AMC*, 38(3):620–650, july 1991.

[Prz90] Teodor C. Przymusinski. Extended stable semantics for normal and disjunctive programs. In *Proceedings on the Seventh International Conference on Logic Programming*, pages 459–477, 1990.

[SZ90] Domenico Saccà and Carlo Zaniolo. Stable models and nondeterminism in logic programs with negation. In *Proceedings of the ACM Symposium on Principles of Database Systems*, page 16, 1990.

[Teu93] Frank Teusink. A characterization of stable models using a nonmonotonic operator. Technical report, Centre for Mathermatics and Computer Science, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands, february 1993.