Alternating Fixed Points in Boolean Equation Systems as Preferred Stable Models

K. Narayan Kumar[1,2], C.R. Ramakrishnan[1], and Scott A. Smolka[1]

[1] Department of Computer Science,
State University of New York at Stony Brook
Stony Brook, New York, U.S.A.
E-mail: {kumar,cram,sas}@cs.sunysb.edu
[2] Chennai Mathematical Institute, Chennai, India.
E-mail: kumar@smi.ernet.in

Abstract. We formally characterize alternating fixed points of boolean equation systems as models of (propositional) normal logic programs. To precisely capture this relationship, we introduce the notion of a preferred stable model of a logic program, and define a mapping that associates a normal logic program with a boolean equation system such that the solution to the equation system can be "read off" the preferred stable model of the logic program. We show that the preferred model cannot be calculated a-posteriori (i.e. compute stable models and choose the preferred one) but its computation rather is intertwined with the stable-model computation itself. This definition reveals a very natural relationship between the evaluation of alternating fixed points in boolean equation systems and the Gelfond-Lifschitz transformation used in stable-model computation.

For alternation-free boolean equation systems, we show that the logic programs we derive are stratified, while for formulas with alternation, the corresponding programs are non-stratified. Consequently, our mapping of boolean equation systems to logic programs preserves the computational complexity of evaluating the solutions of special classes of equation systems (e.g., linear-time for the alternation-free systems, exponential for systems with alternating fixed points).

1 Introduction

Model checking [1, 11, 2] is a verification technique aimed at determining whether a system specification possesses a property expressed as a temporal logic formula. Model checking has enjoyed wide success in verifying, or finding design errors in, real-life systems. An interesting account of a number of these success stories can be found in [3].

Model checking has spurred interest in evaluating *alternating fixed points* as these are needed to express system properties of practical import, such as those involving subtle fairness constraints. Probably, the most canonical temporal logic for expressing alternating fixed points is the modal mu-calculus [10, 7], which makes explicit use of the dual fixed-point operators μ (least fixed point) and

ν (greatest fixed point). A variety of temporal logics can be encoded in the mu-calculus, including Linear Temporal Logic (LTL), Computation Tree Logic (CTL) and its derivative CTL*.

Fixed-point operators may be nested in mu-calculus formulas and different fixed-point formulas may be mutually dependent on each other. Alternating fixed-point formulas are those having a least fixed point that is mutually dependent on a greatest fixed point.

Recently, it has been demonstrated that logic programming (LP) can be successfully applied to the construction of practical and versatile model checkers [12, 5]. Central to this approach is the connection between models of temporal logics and models of logic programs. For example, the XMC model checker [13] verifies an alternation-free modal mu-calculus formula by evaluating the perfect model of an equivalent stratified logic program. While the relationship between models of alternating modal mu-calculus formulae and stable models of logic programs has been conjectured [8], there has been no formal characterization of this connection. Establishing this relationship is the focus of this paper.

The model-checking problem for the modal mu-calculus can be formulated in terms of solving Boolean Equation Systems (BESs); see, e.g., [14]. A BES is a system of mutually dependent equations over boolean-valued variables, where each equation is designated as a greatest or least fixed point. To capture the solutions of a BES in terms of models of logic programs, we introduce the notion of *preferred stable models* of normal logic programs, and describe a mapping from BESs to propositional normal logic programs such that the solution to a BES can be obtained from the preferred stable model of the corresponding logic program. The mapping also ensures that alternation-free BESs are mapped to stratified logic programs, and is thus a conservative extension of the one used by the XMC system. This preserves the linear-time complexity of model checking alternation-free formulas.

The rest of this paper develops along the following lines. Section 2 begins with an overview of BESs and their relationship to logic programs; these ideas are formalized in Section 3. Section 4 introduces the notion of a preferred stable model of a normal logic program, while Section 6 formally establishes the relationship between solutions to BESs and preferred stable models. Our concluding remarks are offered in Section 7. The paper also contains two appendices to make it self-contained. Appendix A reviews the standard connection between model checking in the modal mu-calculus and solving BESs. Appendix B contains the proof of the main theorem of the paper.

2 Overview

2.1 Boolean Equation Systems

A Boolean Equation System is a sequence of fixed-point equations over boolean variables, with an associated sequence of signs that specifies the direction of the fixed points. The i-th equation is of the form $X_i = \alpha_i$ where α_i is a positive

boolean formula over variables $\{X_1, X_2, \dots\}$ and constants 0 and 1. The i-th sign, σ_i is μ if the i-th equation is a least fixed-point equation and ν if it is a greatest fixed-point equation. We use $\langle X_1 = \alpha_1, X_2 = \alpha_2, \dots, X_n = \alpha_n \rangle$ to denote the sequence of equations in a BES of size n, and $\langle \sigma_1, \sigma_2, \dots, \sigma_n \rangle$ to denote its associated sign map. In a BES of size n, X_1 is called the *innermost* variable (and the equation $X_1 = \alpha_1$ the innermost fixed point), and X_n is called the *outermost* variable. A BES of size n is said to be *closed* if all variables occurring in α_i for all $1 \leq i \leq n$ are drawn from $\{X_1, X_2, \dots, X_n\}$.

In the following, we use ϕ to range over BESs, and $\mathcal{E}$ (possibly subscripted) to represent specific BESs. Let ϕ be a BES $\langle X_1 = \alpha_1, X_2 = \alpha_2, \dots, X_n = \alpha_n \rangle$. We use ϕ_i to denote the subsystem $\langle X_1 = \alpha_1, X_2 = \alpha_2, \dots, X_i = \alpha_i \rangle$. Thus $\phi = \phi_n$, and ϕ_0 denotes the empty BES $\langle\ \rangle$.

A solution of a BES is a truth assignment to the variables $\{X_1, X_2, \dots\}$ satisfying the fixed point equations such that the outer equations have higher priority over inner equations. More precisely, a solution of a BES of size i, $\langle X_1 = \alpha_1, X_2 = \alpha_2, \dots, X_i = \alpha_i \rangle$ is a valuation of a fixed point for the outermost variable X_i and is maximal (relative to its sign σ_i) among all solutions for the subsystem $\langle X_1 = \alpha_1, X_2 = \alpha_2, \dots, X_{i-1} = \alpha_{i-1} \rangle$.

Example 2.1. Consider the BES $\mathcal{E}_1 = \langle X_1 = X_1 \wedge X_2, X_2 = X_1 \vee X_2 \rangle$ with sign $\langle \mu, \mu \rangle$. We first consider all solutions for the subsystem $X_1 = X_1 \wedge X_2 \rangle$. Note that, although $(X_1 = 1, X_2 = 1)$ is a fixed point for $X_1 = X_1 \wedge X_2$, it is not a least fixed point (as specified by σ_1) and hence not a solution for $\langle X_1 = X_1 \wedge X_2$. The solutions for the subsystem are $(X_1 = 0, X_2 = 0)$ and $(X_1 = 0, X_2 = 1)$. Both valuations are fixed points for $X_2 = X_1 \vee X_2$, but $(X_1 = 0, X_2 = 0)$ is the least fixed point (as specified by σ_2), and hence is the solution for $\mathcal{E}_1$. □

Example 2.2. The signs of both equations in $\mathcal{E}_1$ were identical; for a more complex example, consider the BES $\mathcal{E}_2 = \langle X_1 = X_1 \wedge X_2, X_2 = X_1 \vee X_2, X_3 = X_3 \wedge X_2 \rangle$ with sign $\langle \mu, \mu, \nu \rangle$. Following the evaluation of $\mathcal{E}_1$, we see that the solutions of $\langle X_1 = X_1 \wedge X_2, X_2 = X_1 \vee X_2 \rangle$ are $(X_1 = 0, X_2 = 0, X_3 = 0)$ and $(X_1 = 0, X_2 = 0, X_3 = 1)$. Of these, only $(X_1 = 0, X_2 = 0, X_3 = 0)$ is a fixed point for $X_3 = X_3 \wedge X_2$ and hence is the solution for $\mathcal{E}_2$. □

In $\mathcal{E}_2$, the inner subsystem's solutions were independent of the values assigned to the outer variable X_3. This property does not hold for every BES, as shown in the following example.

Example 2.3. Consider the BES $\mathcal{E}_3 = \langle X_1 = X_1 \wedge X_2, X_2 = X_1 \vee X_2 \rangle$ with sign $\langle \nu, \mu \rangle$. We first consider all solutions for the subsystem $X_1 = X_1 \wedge X_2$. Note that, although $(X_1 = 0, X_2 = 1)$ is a fixed point for $X_1 = X_1 \wedge X_2$, it is not a greatest fixed point (as specified by σ_1) and hence not a solution for $X_1 = X_1 \wedge X_2$. The solutions for the subsystem are $(X_1 = 0, X_2 = 0)$ and $(X_1 = 1, X_2 = 1)$. Both valuations are fixed points for $X_2 = X_1 \vee X_2$, but $(X_1 = 0, X_2 = 0)$ is the least fixed point (as specified by σ_2), and hence is the solution for $\mathcal{E}_3$. □

Nesting and Alternation in BES: We say that X_i *depends on* X_j if α_i contains a reference to X_j, or to X_k such that X_k depends on X_j. A BES is said to be *nested* if there are two variables X_i and X_j such that X_i depends on X_j and $\sigma_i \neq \sigma_j$. We say that X_i and X_j are *mutually dependent* if X_i depends on X_j and vice versa. A BES is *alternation free* if X_i and X_j are mutually dependent implies $\sigma_i = \sigma_j$. Otherwise, the BES is said to contain *alternating fixed points*. For instance, the BES $\mathcal{E}_1$ has no nested fixed points, $\mathcal{E}_2$ has nested fixed points while $\mathcal{E}_3$ has alternating fixed points. Note that every BES that has alternating fixed points is also nested.

The order of equations in a nested BES is important, as the following example shows.

Example 2.4. Consider the BES $\mathcal{E}_4 = \langle X_1 = X_2 \vee X_1, X_2 = X_2 \wedge X_1 \rangle$ with sign $\langle \mu, \nu \rangle$. Note that $\mathcal{E}_4$ differs from $\mathcal{E}_3$ only in the order in which the equations are defined (and the corresponding change to the names of variables). Valuations $(X_1 = 0, X_2 = 0)$ and $(X_1 = 1, X_2 = 1)$ are solutions for the subsystem $X_1 = X_2 \vee X_1$; among these only $(X_1 = 1, X_2 = 1)$ is a fixed point for $X_2 = X_2 \wedge X_1$, and hence is the solution for $\mathcal{E}_4$. □

2.2 Boolean Equation Systems as Logic Programs

A *normal logic program* over a set of propositions $\mathcal{A}$ is a set of clauses of the form $\gamma \leftarrow \beta$ where $\gamma \in \mathcal{A}$ and β is a boolean formula in negation normal form over $\mathcal{A} \cup \{0, 1\}$. We use 0 and 1 to denote *true* and *false*, respectively. In a clause of the form $\gamma \leftarrow \beta$, γ is called the head of the clause and β its body. We use p and q (possibly subscripted) to denote propositions and P to denote normal logic programs. A *definite logic program* is a program where every clause body is a positive boolean formula. We say that a proposition p *depends on* another proposition q in a program if q appears in the body of a clause with p as the head; p *negatively depends on* q if q appears in the scope of a negation. A program is said to be *stratified* if no cycle in the transitive closure of the dependency relation contains two literals p and q such that p negatively depends on q.

We use stable models as the semantics of normal logic programs [6]. Note that stable models coincide with the standard least-model semantics for definite logic programs and the perfect-model semantics for stratified logic programs.

A BES consisting only of least fixed points (and hence, not nested) can be readily seen as equivalent to a definite propositional logic program. We can thus use logic program evaluation techniques to find the solution for such a BES.

Example 2.5. Consider the propositional program $P_1 = \{p_1 \leftarrow p_1 \wedge p_2,\ p_2 \leftarrow p_1 \vee p_2\}$. This program is equivalent to BES $\mathcal{E}_1$ where p_1 represents X_1 and p_2 represents X_2. The least model for P_1 is $\{\}$, from which we can derive $(X_1 = 0, X_2 = 0)$ as the solution for $\mathcal{E}_1$. □

For a nested but non-alternating BES we can construct a stratified propositional logic program such that the solution of the BES can be obtained from the perfect model of the logic program. The basic strategy is to convert the greatest

fixed-point equations to least fixed-point equations and using the equivalence $\nu X.\phi \equiv \neg\mu Z.\neg\phi[\neg Z/X]$. In fact, the XMC model checker is based on this strategy.

Example 2.6. Consider the propositional program $P_2 = \{p_1 \leftarrow p_1 \wedge p_2.\ p_2 \leftarrow p_1 \vee p_2.\ p_3 \leftarrow \neg q_3.\ q_3 \leftarrow q_3 \vee \neg p_2.\}$. This program is equivalent to BES $\mathcal{E}_2$ where p_i represent X_i. The perfect model for P_2 is $\{q_3\}$ from which we can derive $(X_1 = 0, X_2 = 0, X_3 = 0)$ as the solution for $\mathcal{E}_2$. □

However, for a BES with alternating fixed points, this translation yields a non-stratified logic program which may not have unique stable models.

Example 2.7. Consider the propositional program $P_3 = \{p_1 \leftarrow \neg q_1.\ q_1 \leftarrow q_1 \vee \neg p_2.\ p_2 \leftarrow p_1 \vee p_2.\}$. This program is equivalent to BES $\mathcal{E}_3$ where p_i represent X_i. There are two stable models for P_3: $\{p_1, p_2\}$, and $\{q_1\}$ which gives two candidates $(X_1 = 1, X_2 = 1)$ and $(X_1 = 0, X_2 = 0)$ as solutions for $\mathcal{E}_3$. □

The translation from BES to normal logic programs informally described by the above examples illustrates the problem: a BES has an ordered set of equations (sequence), and the corresponding normal logic program loses the order. In fact it is easy to see that the program P_3 in the example also represents the BES $\mathcal{E}_4$ where p_2 represents X_1 and p_1 represents X_2. Each of the stable models of P_3 correspond to the solutions of $\mathcal{E}_3$ and $\mathcal{E}_4$.

At first sight, it appears that one can simply "select" the appropriate stable model by applying the order information *after* the stable-model computation. The following example illustrates why such an *a posteriori* selection cannot be done.

Example 2.8. Consider BES $\mathcal{E}_5 = \langle X_1 = X_2 \wedge X_3, X_2 = X_1 \wedge X_3, X_3 = X_2 \wedge X_3 \rangle$ with sign $\langle \nu, \mu, \nu \rangle$. The corresponding logic program is $\{p_1 \leftarrow \neg q_1, q_1 \leftarrow \neg p_2 \vee q_3, p_2 \leftarrow \neg q_1 \wedge \neg q_3, p_3 \leftarrow \neg q_3, q_3 \leftarrow \neg p_2 \wedge q_3\}$. The stable models for this program are $\{p_1, p_2, p_3\}$ and $\{q_1, q_3\}$, which correspond to solutions $v_1 = (X_1 = 1, X_2 = 1, X_3 = 1)$ and $v_2 = (X_1 = 0, X_2 = 0, X_3 = 0)$, respectively. Among them, v_1 assigns to X_3 (the outermost variable) the value 1, which is closest to its default (since $\sigma_3 = \nu$). However, the solution to BES $\mathcal{E}_5$ is v_2 since v_1 is not a solution to the subsystem $\langle X_1 = X_2 \wedge X_3, X_2 = X_1 \wedge X_3 \rangle$. □

Hence the "smallest" stable model may not correspond to the solution of a BES. Thus, the order information on the BES lost in the translation to logic programs should be taken into account when the appropriate model is computed. In Section 4 we define the notion of *preferred stable models* where information on ordering of literals is taken into account in the definition of the model itself.

3 Solutions to Boolean Equation Systems

Let $\chi = \{X_1, X_2, \dots\}$ be the set of variables. The set of *positive boolean formulas* over the set χ is described by the following grammar:

$$\alpha \quad := \quad X_i \in \chi \ \mid\ \alpha_1 \wedge \alpha_2 \ \mid\ \alpha_1 \vee \alpha_2$$

A *valuation* v is a map $v : \chi \to \{0,1\}$ with 0 standing for `false` and 1 for `true`. Let $\mathcal{V}$ denote the set of valuations. Given a positive boolean formula α and a valuation v, $\alpha[v]$ denotes the boolean value obtained by evaluating α using valuation v. Given a valuation v and a boolean value a, the valuation $v[a/X_i]$ is the valuation that returns the same value as v for all X_j other than X_i and returns a for X_i.

The solution of a boolean equation system ϕ, denoted by $[\![\phi]\!]$, is defined as a function that maps valuations to valuations. The mapping is such that $[\![\phi]\!](v)$ depends on v only for the free variables of ϕ. Thus, for a closed system $[\![\ ]\!]$ defines a constant function.

We first consider finding solutions to ϕ_i, where $\sigma_i = \mu$. Consider a function f parameterized by a valuation v defined as $f(v) = \lambda x.\alpha_i[v[x/X_i]]$. Since evaluating a formula w.r.t. a valuation results in a boolean value, $f(v)$ maps booleans to booleans. Now, the least fixed point of $f(v)$ (taken w.r.t. the natural partial order on $\{0,1\}$ with 0 less than 1), denoted by $\mathtt{lfp}(f(v))$, gives the smallest value for X_i such that the fixed-point equation $\mu : X_i = \alpha_i$ holds for the given valuation v. Note that the value assigned by v to X_i is immaterial since $f(v)$ considers only $v[x/X_i]$. Let valuation v' be such that for all inner variables X_j, $j \leq i$, $v'(X_j)$ are the fixed points of the equations in ϕ_{i+1} when the outer variables X_k, $k > i$, are substituted by $v'(X_k)$. Then $\mathtt{lfp}(f(v'))$ is the fixed-point value of X_i corresponding to the values of the outer variables as specified by v'. Thus, $[\![\phi_i]\!](v)(X_i)$, is identical to $\mathtt{lfp}(f([\![\phi_{i+1}]\!](v)))$. The semantics of greatest fixed-point equations can be explained similarly.

The solution of a system ϕ is defined by induction on the size i of the system as follows:

$$
\begin{array}{rcl}
[\![\phi_0]\!](v) & = & v \\
[\![\phi_{i+1}]\!](v) & = & \begin{cases} [\![\phi_i]\!](v[\mathtt{lfp}(\xi_{i+1})/X_{i+1}]) \text{ if } \sigma(i+1) = \mu \\ [\![\phi_i]\!](v[\mathtt{gfp}(\xi_{i+1})/X_{i+1}]) \text{ if } \sigma(i+1) = \nu \end{cases} \quad i \geq 0 \\
\multicolumn{3}{c}{\text{where } \xi_{i+1} = \lambda x.\alpha_{i+1}[[\![\phi_i]\!](v[x/X_{i+1}])]}
\end{array}
$$

3.1 Solutions as Preferred Fixed Points

It is useful to treat the solution to a BES as the "minimum valuation" that satisfies the equations in the BES. We now formalise this notion. We define a family of partial orders $\sqsubseteq_i$, $i \geq 1$, on valuations that captures our intuition that least fixed-point variables take values as close to 0 as possible and greatest fixed-point variables take values as close to 1 as possible. Further, it also captures the idea that outer variables (i.e. variables with a higher index) have higher priority then inner variables (i.e. variables with a lower index).

We say that a valuation v is a *fixed point* of the system $\langle X_1 = \alpha_1, \ldots, X_n = \alpha_n \rangle$ if $v(X_i) = \alpha_i[v]$ for $1 \leq i \leq n$.

Definition 3.1 (Fixed Points of a BES). *The set of fixed points of a BES ϕ with respect to a valuation v, denoted by $FP(v)(\phi)$, is such that*

$$FP(v)(\phi_0) = v$$
$$FP(v)(\phi_{i+1}) = \{u \mid u \in FP(v)(\phi_i) \text{ and } u(X_{i+1}) = \alpha_{i+1}[u]\} \quad \text{if } i \geq 0$$

Note that in the above definition, we ignore the signs of the equations. We now define a partial order on valuations based on the signs is used to select the preferred fixed point.

For a given sign map σ, we define the following partial orders: Let the partial order $\leq_i$ over $\{0,1\}$ be defined as $0 \leq_i 1$ iff $\sigma(i) = \mu$ and $1 \leq_i 0$ iff $\sigma(i) = \nu$. The partial order $\sqsubseteq_i$ over valuations is defined by recursion over i as follows:

$$u \sqsubseteq_1 v \iff u(X_1) \leq_1 v(X_1)$$
$$u \sqsubseteq_{i+1} v \iff u(X_{i+1}) <_{i+1} v(X_{i+1}) \text{ or } u(X_{i+1}) = v(X_{i+1}) \ \wedge \ u \sqsubseteq_i v$$

We say that $u \sqsubset_i v$ if $u \sqsubseteq_i v$ and $u \neq v$. It is easy to see that $\sqsubseteq_i$ is a partial order on any set of valuations that agree at X_j for all $j > i$.

Definition 3.2 (Preferred Fixed Points). *The preferred fixed points of a BES ϕ with respect to a valuation v, denoted by $PFP(v)(\phi)$, is the set of valuation such that:*

$$PFP(v)(\phi_0) = v$$
$$PFP(v)(\phi_{i+1}) = \min_{\sqsubseteq_{i+1}}(PFP(v)(\phi_i) \cap FP(v)(\phi_{i+1})) \quad \text{if } i \geq 0$$

Observe from the above definitions that $PFP(v)(\phi) \subseteq FP(v)(\phi)$. Moreover, $u(X_j) = v(X_j)$ for all $j > (i+1)$ for any $u \in FP(v)(\phi_i)$. Thus, $\sqsubseteq_{i+1}$ is a linear order (as it is a lexicographic order) on the set of valuations in $PFP(v)(\phi_i) \cap FP(v)(\phi_{i+1})$. This leads us to the following proposition:

Proposition 3.3. *For every closed boolean equation system ϕ, there is a unique preferred fixed point, i.e., $|PFP(\phi)| = 1$.*

For the preferred fixed point to capture the solution of a given BES, the preference $\min_{\sqsubseteq}$ must be applied to a set of preferred fixed points of the inner equation. To see this, consider the following formula:

$$X_1 = X_2 \wedge X_3$$
$$X_2 = X_3 \wedge X_1$$
$$X_3 = X_2 \wedge X_3$$

with $\sigma(1) = \nu$, $\sigma(2) = \mu$ and $\sigma(3) = \nu$. It is easy to verify that the valuation v that assigns 1 to X_1, X_2 and X_3 is a fixed point, and is minimal w.r.t. $\sqsubseteq_3$. However, it is also easy to check that the solution of the BES assigns 0 to X_1, X_2 and X_3, and this is the preferred fixed point according to the definition given above.

Theorem 3.4. *Let ϕ be a BES of size n. Then, for all $i \leq n$, $[\![\phi_i]\!](v)$ is the preferred fixed point of ϕ_i w.r.t. v.*

The proof follows by an easy induction on i.

4 Preferred Stable Models of Normal Logic Programs

Let $P = \{p_1 \leftarrow \beta_1, p_2 \leftarrow \beta_2, \ldots, p_n \leftarrow \beta_n\}$ be a logic program. A proposition $p \notin \{p_1, p_2, \ldots, p_n\}$ is said to be *free* in P if there is some β such that p occurs in β_i.

We represent a model of a logic program by a substitution that maps propositions to truth values $\{0, 1\}$. We use w_0 to denote the substitution that maps all propositions to 0. Given a substitution w over propositions, we extend it to literals such that $w(\neg p) = \neg w(p)$ for every proposition p, where $\neg 0 = 1$ and $\neg 1 = 0$. Finally, given a program P and a substitution w, the program $P[w]$ is the one obtained by substituting all *free* propositions p in P by $w(p)$.

Definition 4.1 (Least Models for Definite Logic Programs). *The least model of a definite logic program P w.r.t. a substitution w on the free propositions of P is defined by the following equations:*

$$\begin{array}{rcl} M(\{\})(w) & = & w \\ M(\{p_i \leftarrow \beta_i\} \cup P')(w) & = & M(P')(w[b_i/p_i]) \\ & & \textit{where } b_i = \mathtt{lfp}(\lambda x.\beta_i[M(P')(w[x/p_i])]) \end{array}$$

The traditional least model of P under the closed world assumption is simply $M(P)(w_0)$.

We now recall the definition of stable model semantics for normal logic programs.

Definition 4.2 (Gelfond-Lifschitz Transformation [6]). *The Gelfond-Lifschitz transform of a propositional normal logic program P with respect to substitution w is a program $P \downarrow w$ obtained by replacing every negative literal of the form $\neg p$ in P by $\neg w(p)$.*

Note that for all P and w, $P \downarrow w$ is a definite logic program. A substitution w is a *stable model* of a program P iff it is the least model of $P \downarrow w$.

Definition 4.3 (Stable Models). *The set of stable models of a normal logic program P w.r.t. a substitution w on the free propositions of P, denoted by $SM(P)(w)$, is defined as:*

$$SM(P)(w) = \{u \mid u = M(P[w] \downarrow u)(w)\}$$

4.1 Preferred Stable Models

We now define stable models w.r.t. a *preference sequence*: a sequence $S = \langle l_1, l_2, \ldots, l_m \rangle$ of literals such that no proposition appears both positively and negatively in S. As usual, we represent by S_i the initial subsequence of S of length i. Given a substitution w which maps propositions to truth values, we extend w to literals with the usual interpretation that $w(\neg p) = \neg w(p)$ for some proposition p where $\neg 0 = 1$ and $\neg 1 = 0$.

Definition 4.4 (Preference Order $\sqsubseteq_S$). *Given two substitutions w_1 and w_2 and a preference sequence $S = \langle l_1, l_2, \ldots, l_m \rangle$, we say that w_2 is preferred over w_1 (written as $w_1 \sqsubseteq_S w_2$) if $w_1(l_m) < w_2(l_m)$, or $w_1(l_m) = w_2(l_m)$ and $w_1 \sqsubseteq_{S_{m-1}} w_2$. For an empty preference sequence $S = \langle\ \rangle$, $w_1 \sqsubseteq_S w_2$ for any pair of substitutions w_1 and w_2.*

Note that $\sqsubseteq_S$ defines a lexicographic order on substitutions, and hence is a partial order. Moreover, for any pair of substitutions w_1, w_2 that agree at all literals not in S, we have $w_1 \sqsubseteq_S w_2 \wedge w_2 \sqsubseteq_S w_1 \Rightarrow w_1 = w_2$. This means that for every set of substitutions that agree on all literals not in S, there is a unique minimum element w.r.t. $\sqsubseteq_S$. We denote this element by $\min_{\sqsubseteq_S}$.

Definition 4.5 (Preferred Stable Models). *The preferred stable model of a normal logic program P w.r.t. to a substitution w and a preference sequence S, denoted by $PSM_S(P)(w)$, is defined inductively on the size of P as follows:*

$$PSM_S(\{\})(w) = w$$
$$PSM_S(\{p_i \leftarrow \beta_i\} \cup P')(w) = \min_{\sqsubseteq_S}(PSM_S(P')(w) \cap SM(\{p_i \leftarrow \beta_i\} \cup P')(w))$$

By $PSM_S(P)$ we denote the set of all preferred stable models w.r.t. arbitrary substitutions.

It is easy to show that the above definition is well-defined in the sense that the value of PSM_S is independent of the clause $\{p_i \leftarrow \beta_i\}$ selected for use in the recursive case.

A preference sequence S is said to be *complete* w.r.t. program P if every proposition in P appears (positively or negatively) in S. Hence every program that has at least one stable model has exactly one preferred stable model w.r.t. a complete preference sequence. Formally,

Proposition 4.6 (Uniqueness of Preferred Stable Models). *Let P be a normal logic program, and S be a preference sequence that is complete w.r.t. P. Then $|PSM_S(P)| \leq 1$. Moreover $|PSM_S(P)| = 0$ iff $PSM_S(P) = \{\}$.*

5 Mapping Boolean Equation Systems to Propositional Logic Programs

In order to map BESs to logic programs, we first consider the mapping between the variables in a given BES ϕ and propositions in the corresponding logic program P. The logic program P we derive is over propositions $\{p_1, p_2, \ldots, q_1, q_2, \ldots\}$ such that each variable X_i in ϕ corresponds to literal p_i if $\sigma(i) = \mu$ and to $\neg q_i$ if $\sigma(i) = \nu$. The idea behind the mapping is to translate the equations into clauses in a normal logic program, considering greatest fixed points in terms of their dual least fixed points. The salient aspect of the mapping we define is that negation is used only where absolutely necessary: negation will be used only when variables

of differing (fixed-point) signs are related. The following function $\mathcal{M}$ translates variables of a BES to propositions in the translated logic program:

$$\mathcal{M}(X_i) = \begin{cases} p_i & \text{if} \quad \sigma(i) = \mu \\ \neg q_i & \text{if} \quad \sigma(i) = \nu \end{cases}$$

We lift $\mathcal{M}$ to boolean expressions by replacing the variables in a given expression using the above definition. In order to translate greatest fixed-point equations, we need to find the dual of the equation. This is done by constructing $\overline{\alpha}$, which is $\neg\alpha$ to negation normal form. When we apply $\mathcal{M}$ to boolean expressions with negation, we also reduce expressions of the form $\neg\neg p$ to p.

Definition 5.1. *The translation function $\mathcal{P}$ maps boolean equation systems to normal logic programs, such that $\mathcal{P}(\phi)$ for a boolean equation system ϕ is given by*

$$\begin{aligned} \mathcal{P}(\phi_0) &= \{\} \\ \mathcal{P}(\phi_{i+1}) &= \begin{cases} \{p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1})\} \cup \mathcal{P}(\phi_i) & \textit{if} \quad \sigma(i+1) = \mu \\ \{q_{i+1} \leftarrow \mathcal{M}(\overline{\alpha_{i+1}})\} \cup \mathcal{P}(\phi_i) & \textit{if} \quad \sigma(i+1) = \nu \end{cases} \qquad \textit{for} \quad i \geq 0 \end{aligned}$$

Note that for each X_i in ϕ there is exactly one proposition in $\mathcal{P}(\phi)$. Observe that if X_k appears in α_i then the literal corresponding to X_k appears in negated form in the program clause corresponding to α_i if and only if $\sigma(i) \neq \sigma(k)$. Thus, negative dependencies are introduced in $\mathcal{P}(\phi)$ only if the corresponding variables in ϕ differ in sign. In an alternation-free BES, the dependency between opposite-signed variables is cycle-free. Hence we have the following proposition:

Proposition 5.2. *If ϕ is an alternation-free boolean equation system then $\mathcal{P}(\phi)$ is a stratified logic program.*

Stratified logic programs have unique stable models which can be evaluated in polynomial time. Thus, the logic programs generated from alternation-free BESs can be efficiently evaluated.

We complete the translation by defining a mapping between sign maps of a given BES and preference sequences for the corresponding logic program.

Definition 5.3. *The translation function $\mathcal{P}$ maps the sign map σ of a BES of size n to the preference sequence $\langle l_1, l_2, \ldots, l_n \rangle$ such that for all $1 \leq i \leq n$:*

$$l_i = \begin{cases} p_i & \textit{if} \quad \sigma(i) = \mu \\ \neg q_i & \textit{if} \quad \sigma(i) = \nu \end{cases}$$

We have overloaded the symbol $\mathcal{P}$ to denote the translation functions that map different aspects of the BES to logic programs with preferences. Note that $\mathcal{P}(\sigma)$ is a complete preference sequence whenever σ is a sign map of a closed BES.

6 Solutions to Boolean Equation Systems are Preferred Stable Models

We now show that given a closed BES ϕ of size n, its solution can be obtained from the preferred stable model of $\mathcal{P}(\phi)$. This is done by showing that

(i) the preference order among fixed points in a BES corresponds to the order imposed by preference sequences,
(ii) every stable model of $\mathcal{P}(\phi)$ is a fixed point of ϕ, and
(iii) the preferred fixed point of ϕ is a stable model of $\mathcal{P}(\phi)$.

We first formalize the relationship between the valuations of a BES and the models of the corresponding logic program. Given a valuation v over $\chi = \{X_1, X_2, \dots\}$ let $\mathcal{P}(v)$ be a substitution over $\mathcal{A} = \{p_1, p_2, \dots, q_1, q_2, \dots\}$ such that for all $i \geq 0$, $\mathcal{P}(v)(p_i) = v(X_i)$ and $\mathcal{P}(v)(q_i) = \neg v(X_i)$. Similarly, for any substitution w over $\mathcal{A}$ satisfying $w(p_i) = \neg w(q_i)$ for every i, we write $\mathcal{P}^{-1}(w)$ to denote the valuation over χ such that $\mathcal{P}^{-1}(w)(X_k) = w(p_k)$.

The correspondence between valuations and substitutions, and between sign maps and preference sequences, is formalized in the following theorem:

Theorem 6.1. *Let ϕ be a closed BES of size n with sign map σ and let $S = \mathcal{P}(\sigma)$ be a preference sequence. For any pair of valuations v_1, v_2 over $\{X_1, X_2, \dots, X_n\}$, $v_1 \sqsubseteq_n v_2$ iff $\mathcal{P}(v_1) \sqsubseteq_S \mathcal{P}(v_2)$.*

The following theorem formally states the second step needed for establishing the correspondence between preferred fixed points and preferred stable models:

Theorem 6.2. *Let ϕ be a BES of size n and v be a valuation. If u is a stable model of $\mathcal{P}(\phi_n)[\mathcal{P}(v)]$ then $\mathcal{P}^{-1}(u)$ is a fixed point of ϕ w.r.t. the valuation v.*

The proof follows from the definition of stable models and the translation mapping $\mathcal{P}$.

The third step would be trivial if the converse of Theorem 6.2 were true. However, it turns out that not all fixed points of a BES correspond to stable models of the translated program. This mismatch arises because the definition of fixed points completely ignores the signs of the equations as well as the order of nesting, while the translation from BES to logic programs ignores only the order of nesting. Thus even for non-nested BESs the set of fixed points may be larger than the set of stable models of the corresponding program. For example, consider the BES ϕ with equations $\langle X_1 = X_2, X_2 = X_1\rangle$ and sign map $\langle \mu, \mu\rangle$. The BES has two fixed points: $v_1 = (X_1 = 0, X_2 = 0)$ and $v_2 = (X_1 = 1, X_2 = 1)$, with v_1 as the solution. The program $\mathcal{P}(\phi)$ is $\{p_1 \leftarrow p_2, p_2 \leftarrow p_1\}$, and has only one stable model $\{\}$. Similarly the system $\mathcal{E}_3$ in Example 2.3 (Section 2) has three fixed points but of these only two ($(X_1 = 0, X_2 = 0)$ and $(X_1 = 1, X_2 = 1)$) correspond to stable models ($\{q_1\}$ and $\{p_1, p_2\}$ respectively) of the translated program.

Thus, we need to show that we have not "lost" the solution to a BES in the translation, as formally stated by the following theorem.

Theorem 6.3. *Let ϕ be a closed BES of size n and v be a valuation. Then, for all n, $\mathcal{P}([\![\phi_n]\!](v))$ is a stable model of $\mathcal{P}(\phi_n)[\mathcal{P}(v)]$.*

Proof. See Appendix B.

Thus, the stable models of $\mathcal{P}(\phi)$ correspond (via the translation function $\mathcal{P}$ over valuations) to a subset of the set of fixed points of ϕ that contains the preferred fixed point of ϕ. Since preference orders over valuations and substitutions coincide (from Theorem 6.1) it is easy to establish from the definition of preferred stable models that the preferred stable model of $\mathcal{P}(\phi)$ corresponds to the preferred fixed point of ϕ.

Corollary 6.4. *Let ϕ be a closed BES with sign map σ and let $S = \mathcal{P}(\sigma)$. Then $PFP(\phi) = \{v\}$ and $PSM_S(\mathcal{P}(\phi)) = \{w\}$ such that $v = \mathcal{P}^{-1}(w)$.*

7 Conclusions

We have shown how to compute alternating fixed points of boolean equation systems by translating a given equation system ϕ into a propositional normal logic program $\mathcal{P}(\phi)$, and computing a particular stable model of $\mathcal{P}(\phi)$.

This result provides the basis for extending the XMC model checker, which currently handles only the alternation-free fragment of the modal mu-calculus, to the full mu-calculus, complete with alternating fixed points. The extended model checker works roughly as follows. Translate the given modal mu-calculus formula (expressed equationally) into a propositional normal logic program P, using the translation procedure of Section 5. Feed P into the XSB logic programming system [15], XMC's underlying logic-programming engine. XSB computes the well-founded semantics of normal logic programs. Since, in general, P is non-stratified, this results in a certain "residual program" $r(P)$ of P. Now, feed $r(P)$ into the stable-models generator of [9]—with its search procedure suitably modified—to compute $r(P)$'s preferred stable model. The answer to the original model-checking question can be directly obtained from this model. Experimental results have shown that the residual programs so derived are typically quite small and the preferred stable model can be calculated quite efficiently [8].

We have also shown that for alternation-free boolean equation systems, the logic programs we derive are stratified. Consequently, our mapping of boolean equation systems to logic programs preserves the linear-time complexity of evaluating solutions of such equation systems established in [4]. We moreover conjecture that for boolean equation systems with alternating fixed points, time complexity exponential in the "alternation depth" of the equation system can be attained, again matching the best upper bound known to date. This result would depend critically on the use of the Gelfound-Lifschitz transformation to steer the computation of the preferred stable models of the non-stratified logic programs that our translation produces in the case of alternating fixed points.

References

1. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In D. Kozen, editor, *Proceedings of the Workshop on Logic of Programs,* Yorktown Heights, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer Verlag, 1981.
2. E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM TOPLAS*, 8(2), 1986.
3. E. M. Clarke and J. M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4), December 1996.
4. R. Cleaveland and B. U. Steffen. A linear-time model checking algorithm for the alternation-free modal mu-calculus. *Formal Methods in System Design*, 2:121–147, 1993.
5. G. Delzanno and A. Podelski. Model checking in CLP. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 223–239, 1999.
6. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *International Conference on Logic Programming*, pages 1070–1080, 1988.
7. D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
8. Xinxin Liu, C. R. Ramakrishnan, and Scott A. Smolka. Fully local and efficient model checking of alternating fixed points. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1384 of *Lecture Notes in Computer Science*, pages 56–70. Springer Verlag, 1998.
9. I. Niemela and P. Simons. Efficient implementation of the well-founded and stable model semantics. In *Joint International Conference and Symposium on Logic Programming*, pages 289–303, 1996.
10. V. R. Pratt. A decidable mu-calculus. In *Proceedings of the 22nd IEEE Ann. Symp. on Foundations of Computer Science,* Nashville, Tennessee, pages 421–427, 1981.
11. J. P. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *Proceedings of the International Symposium in Programming*, volume 137 of *Lecture Notes in Computer Science*, Berlin, 1982. Springer-Verlag.
12. Y. S. Ramakrishna, C. R. Ramakrishnan, I. V. Ramakrishnan, S. A. Smolka, T. L. Swift, and D. S. Warren. Efficient model checking using tabled resolution. In *Proceedings of the 9th International Conference on Computer-Aided Verification (CAV '97)*, Haifa, Israel, July 1997. Springer-Verlag.
13. C.R. Ramakrishnan, I.V. Ramakrishnan, S.A. Smolka, Y. Dong, X. Du, A. Roychoudhury, and V.N. Venkatakrishnan. XMC: A logic-programming-based verification toolset. In *Computer Aided Verification (CAV)*, 2000.
14. B. Vergauwen and J. Lewi. Efficient local correctness checking for single and alternating boolean equation systems. In *Proceedings of ICALP'94*, pages 304–315. LNCS 820, 1994.
15. XSB. The XSB tabled logic programming system. Available from `http://xsb.sourceforge.net`.

A Modal Mu-Calculus and Boolean Equation Systems

Formulas in the modal mu-calculus are constructed from existential (denoted by $\langle\cdot\rangle$) and universal (denoted by $[\cdot]$) modalities; explicit greatest and least fixed-point operators (denoted by ν and μ, respectively); formula variables that index the fixed points; and the traditional conjunction/disjunction operators and constants *true* and *false* from classical logic. Models of mu-calclus formulas are given in terms of sets of vertices (called states) of a (edge-) labeled graph (called labeled transition system). For instance, the formula

$$\nu X.\langle - \rangle \mathit{true} \wedge [-]X$$

characterizes deadlock freedom: a state s that models X is such that it is possible to make a transition from s (i.e., the meaning of $\langle - \rangle \mathit{true}$) and every destination state reached models X (the meaning of $[-]X$).

Fixed points in a mu-calculus formula may be nested: i.e., a fixed-point formula ϕ_1 may occur in the scope of another fixed point formula ϕ_2. We then say that the outer formula ϕ_2 depends on the inner formula ϕ_1. Moreover, the inner fixed-point formula ϕ_1 may refer to the variable indexing the outer fixed point ϕ_2, thereby making ϕ_1 and ϕ_2 mutually dependent. Forumlas where the mutually dependent fixed points have different fixed-point operators (i.e., μ and ν) are called *alternating* fixed-point formulas. An example of an alternating fixed-point formula is:

$$\nu X.\mu X'.[a]X \wedge [-a]X'$$

which expresses the property that transitions labeled a occur infinitely often in every infinite path of the system.

The problem of determining whether a labeled transition system constitutes a model of a modal mu-calculus formula can be directly translated into the problem of solving a corresponding boolean equation system. For example, the model for the deadlock-freedom formula with respect to the labeled transition system depicted in Figure 1(a) can be derived from the solution to the boolean equation system given in Figure 1(c). Each variable X_i in the boolean equation system can be seen as describing whether or not state s_i in the LTS is in the model of the modal mu-calculus formula X.

B Proof of Theorem 6.3

Theorem 6.3. Let ϕ be a closed BES of size n and v be a valuation. Then, for all n, $\mathcal{P}([\![\phi_n]\!](v))$ is a stable model of $\mathcal{P}(\phi_n)[\mathcal{P}(v)]$.

Proof. We shall establish the proof of this theorem by induction on n, the size of the formula ϕ.

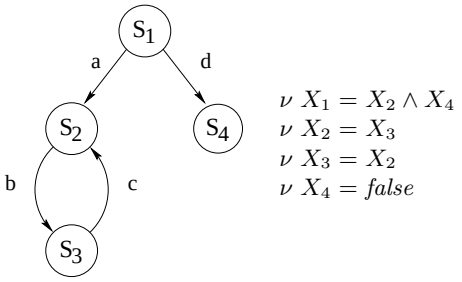

Fig. 1. Example Labeled Transition System (a), mu-calculus formula for deadlock freedom (b), corresponding Boolean Equation System (c), and its solution (d).

Basis: $n = 1$ Let us first consider the case $\sigma(1) = \mu$. Thus,

$$\mathcal{P}(\phi_1) = \{p_1 \leftarrow \mathcal{M}(\alpha_1)\}$$

Further, since p_1 do not appear in negative literals in $\mathcal{M}(\alpha_1)$

$$\mathcal{P}(\phi_1)[\mathcal{P}(v)] \downharpoonleft \mathcal{P}([\![\phi_1]\!](v)) = \{(p_1 \leftarrow \mathcal{M}(\alpha_1))\}[\mathcal{P}(v)]$$

Thus,

$$M(\mathcal{P}(\phi_1)[\mathcal{P}(v)] \downharpoonleft \mathcal{P}([\![\phi_1]\!](v))) = M(\{(p_1 \leftarrow \mathcal{M}(\alpha_1))\}[\mathcal{P}(v)])$$

Thus, from the definition of semantics of definite logic programs, we have

$$\begin{aligned} M(\mathcal{P}(\phi_1)[\mathcal{P}(v)] \downharpoonleft \mathcal{P}([\![\phi_1]\!](v)))(p_1) &= \mathtt{lfp}[\lambda x.\mathcal{M}(\alpha_1)[\mathcal{P}(v)[x/p_1]]\ ] \\ &= \mathtt{lfp}[\lambda x.\alpha_1[x/X_1]] \\ &= [\![\phi_1]\!](v)(X_1) \\ &= \mathcal{P}([\![\phi_1]\!](v))(p_1) \end{aligned}$$

Next we consider the case when $\sigma(1) = \nu$. Thus,

$$\mathcal{P}(\phi_1) = \{q_1 \leftarrow \mathcal{M}(\overline{\alpha_1})\}$$

Further, as q_1 does not appear in any negative literal in $\mathcal{M}(\overline{\alpha_1})$,

$$\mathcal{P}(\phi_1)[\mathcal{P}(v)] \downharpoonleft \mathcal{P}([\![\phi_1]\!](v)) = \{(q_1 \leftarrow \mathcal{M}(\overline{\alpha_1}))[\mathcal{P}(v)]\}$$

Now,

$$\begin{aligned}
&M(\mathcal{P}(\phi_1)[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_1]\!](v)))(q_1)\\
&= \mathtt{lfp}([\lambda x.\mathcal{M}(\overline{\alpha_1})[\mathcal{P}(v)[x/q_1]] \,])\\
&= \neg\mathtt{gfp}([\lambda x.\mathcal{M}(\alpha_1)[\mathcal{P}(v)[x/q_1]] \,])\\
&= \neg\mathtt{gfp}([\lambda x.\alpha_1[v[x/X_1]] \,])\\
&= \neg[\![\phi_1]\!](v)(X_1)\\
&= \mathcal{P}([\![\phi_1]\!](v))(q_1)
\end{aligned}$$

Induction step: Let us consider the case when $\sigma(i+1) = \mu$ (The proof with $\sigma(i+1) = \nu$ is similar and is omitted.) Then,

$$\mathcal{P}(\phi_{i+1}) = \left\{ \begin{array}{l} p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1}) \\ \mathcal{P}(\phi_i) \end{array} \right\}$$

Thus,

$$\mathcal{P}(\phi_{i+1})[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)) = \left\{ \begin{array}{l} (p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1}))[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)) \\ \mathcal{P}(\phi_i)[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)) \end{array} \right\}$$

Thus,

$$\begin{aligned}
&M(\mathcal{P}(\phi_{i+1})[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)))\\
&= M(\left\{ \begin{array}{l} (p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1}))[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)) \\ \mathcal{P}(\phi_i)[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)]) \end{array} \right\})
\end{aligned}$$

The following fact about propositional definite logic programs is well known and rather easy to establish. As always, we use characteristic functions to denote models of propositional logic programs.

Proposition B.1. *Let $P = \{p_1 \leftarrow \alpha_1, \ldots, p_n \leftarrow \alpha_n\}$ be a propositional definite logic program. Let $P_j^1 = \{p_2 \leftarrow \alpha_2', \ldots, p_n \leftarrow \alpha_n'\}$ for $j \in \{0,1\}$, where α_i' is obtained by substituting j for p_1 in α_i. Then, $M(P) = M(P_0^1)[0/p_1]$ iff $M(p_1 \leftarrow \alpha_1[M(P_0^1)])(p_1) = 0$ and $M(P) = M(P_1^1)[1/p_1]$ otherwise.*

Let us now consider the case when $[\![\phi_{i+1}]\!](v)(X_{i+1}) = 0$. Then, $\mathcal{P}([\![\phi_{i+1}]\!](v))(p_{i+1}) = 0$.

We now apply Proposition B.1 with $p_1 = p_{i+1}$. Here P_0^1 is

$$\begin{aligned}
&\mathcal{P}(\phi_i)[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)]) \text{ with 0 substituted for } p_{i+1}\\
&= \mathcal{P}(\phi_i)[\mathcal{P}(v[0/X_{i+1}])] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)])
\end{aligned}$$

But,

$$
\begin{aligned}
& M(\mathcal{P}(\phi_i)[\mathcal{P}(v[0/X_{i+1}])] \wr \mathcal{P}([\![\phi_{i+1}]\!](v))) \\
&= M(\mathcal{P}(\phi_i)[\mathcal{P}(v[0/X_{i+1}])] \wr \mathcal{P}([\![\phi_i]\!](v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}]))) \\
& \{ \text{ Since } [\![\phi_{i+1}]\!](v) = [\![\phi_i]\!](v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}]) \ \} \\
&= M(\mathcal{P}(\phi)[\mathcal{P}(v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}])] \wr \mathcal{P}([\![\phi_i]\!](v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}]))) \\
& \{ \text{ Since } [\![\phi_{i+1}]\!](v)(X_{i+1}) = 0 \ \} \\
&= \mathcal{P}([\![\phi_i]\!](v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}])) \\
& \{ \text{ By induction hypothesis } \} \\
&= \mathcal{P}([\![\phi_{i+1}]\!](v))
\end{aligned}
$$

Now,

$$
\begin{aligned}
& M((p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1}))[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v))[\mathcal{P}([\![\phi_{i+1}]\!](v))])(p_{i+1}) \\
&= M((p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1}))[\mathcal{P}([\![\phi_{i+1}]\!](v))] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)))(p_{i+1}) \\
&= M((p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1}))[\mathcal{P}([\![\phi_{i+1}]\!](v))])(p_{i+1}) \\
&= \mathtt{lfp}([\ a \mapsto \mathcal{M}(\alpha_{i+1})[\mathcal{P}([\![\phi_{i+1}]\!](v))[a/p_{i+1}]] \ ] \\
&= \mathtt{lfp}([\ a \mapsto \alpha_{i+1}[[\![\phi_{i+1}]\!](v)[a/X_{i+1}]] \ ]) \\
&= \mathtt{lfp}([\ a \mapsto \alpha_{i+1}[([\![\phi_i]\!](v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}]))[a/X_{i+1}]] \ ] \\
&= \mathtt{lfp}([\ a \mapsto \alpha_{i+1}[([\![\phi_i]\!](v[0/X_{i+1}]))[a/X_{i+1}]] \ ]
\end{aligned}
$$

But this least fixed point is 0 ($= \mathcal{P}([\![\phi_{i+1}]\!](v))(p_{i+1})$) as

$$
\begin{aligned}
& \alpha_{i+1}[([\![\phi_i]\!](v[0/X_{i+1}]))[0/X_{i+1}]] \\
&= \alpha_{i+1}[[\![\phi_i]\!](v[0/X_{i+1}])] \\
&= \alpha_{i+1}[[\![\phi_i]\!](v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}])] \\
&= [\![\phi_{i+1}]\!](v)(X_{i+1}) \\
&= 0
\end{aligned}
$$

Thus, by Proposition B.1,

$$
\begin{aligned}
& M(\mathcal{P}(\phi_{i+1})[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v))) \\
&= \mathcal{P}([\![\phi_{i+1}]\!](v))
\end{aligned}
$$

Next we consider the case when $[\![\phi_{i+1}]\!](v)(X_{i+1}) = 1$. Let

$$I = \{1 \leq j \leq i \ \mid \ \sigma(j) = \mu\}.$$

Then, the program $\mathcal{P}(\phi_{i+1})[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v))$ can be written as follows

$$
\left\{ \begin{array}{l} \left\{ \begin{array}{l} (p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1}))[\mathcal{P}(v)] \\ \{p_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \in I}[\mathcal{P}(v)] \end{array} \right\} \\ \{q_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \notin I}[\mathcal{P}(v)] \end{array} \right\} \ \wr \mathcal{P}([\![\phi_{i+1}]\!](v))
$$

which is the same as

$$
\left\{ \begin{array}{l} \left\{ \begin{array}{l} (p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1}))[\mathcal{P}(v)] \\ \{p_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \in I}[\mathcal{P}(v)] \end{array} \right\} \wr \mathcal{P}([\![\phi_{i+1}]\!](v)) \\ \{q_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \notin I}[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)) \end{array} \right\}
$$

Now, we can use Observation 1 to rewrite this as,

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} (p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1})) \\ \{p_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \in I} \end{array} \right\} [\mathcal{P}([\![\phi_{i+1}]\!](v))] \\ \{q_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \notin I} [\mathcal{P}([\![\phi_{i+1}]\!](v))] \end{array} \right\}$$

We now try to apply Proposition B.1 with $p_1 = p_{i+1}$. So, we need to evaluate the least herbrand model of the following program:

$$\left\{ \begin{array}{l} \{p_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \in I} [\mathcal{P}([\![\phi_{i+1}]\!](v))[0/p_{i+1}]] \\ \{q_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \notin I} [\mathcal{P}([\![\phi_{i+1}]\!](v))[0/p_{i+1}]] \end{array} \right\}$$

Note that the two parts of this program are independent of each other, the variable p_{i+1} appears only in the first part and that too only positively. Thus, we need to evaluate the following:

$$u = M(\left\{ \begin{array}{l} \{p_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \in I} [\mathcal{P}([\![\phi_{i+1}]\!](v)[0/X_{i+1}])] \\ \{q_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \notin I} [\mathcal{P}([\![\phi_{i+1}]\!](v))] \end{array} \right\})$$

Let

$$w = M(\left\{ \begin{array}{l} \{p_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \in I} [\mathcal{P}([\![\phi_{i+1}]\!](v)[0/X_{i+1}])] \\ \{q_j \leftarrow \mathcal{M}(\alpha_j)\}_{j \notin I} [\mathcal{P}([\![\phi_{i+1}]\!](v)[0/X_{i+1}])] \end{array} \right\})$$

Since every occurance of p_{i+1} in $\mathcal{M}(\alpha_j)$ is negative for each $j \notin I$, and the two parts of the program are independent of each other, $u(p_j) = w(p_j)$ for all $j \in I$ and $u(q_j) \leq w(q_j)$ for all $j \in I$.

Now let us consider the valuations $[\![\phi_{i+1}]\!](v)[0/X_{i+1}]$ and $[\![\phi_i]\!](v[0/X_{i+1}])$. Note that

$$\begin{array}{rl} [\![\phi_{i+1}]\!](v) &= [\![\phi_i]\!](v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}]) \\ &= [\![\phi_i]\!](v[1/X_{i+1}]) \end{array}$$

Thus, by the monotonicity of the operators in the formula, we have that

$$[\![\phi_{i+1}]\!](v)[0/X_{i+1}](X_j) \geq [\![\phi_i]\!](v[0/X_{i+1}])(X_j)$$

for all $1 \leq j \leq i+1$.

Thus,

$$\mathcal{P}([\![\phi_{i+1}]\!](v)[0/X_{i+1}])(p_j) \geq \mathcal{P}([\![\phi_i]\!](v[0/X_{i+1}]))(p_j)$$

for all $j \in (I \cup \{i+1\})$ and

$$\mathcal{P}([\![\phi_{i+1}]\!](v)[0/X_{i+1}])(q_j) \leq \mathcal{P}([\![\phi_i]\!](v[0/X_{i+1}]))(q_j)$$

for all $j \notin (I \cup \{i+1\})$.

Thus, for all p_j, $j \in I$,

$$\begin{array}{l} M(\{p_j \leftarrow \mathcal{M}(\alpha_j)\}_{j\in I}[\mathcal{P}([\![\phi_{i+1}]\!](v)[0/X_{i+1}])])(p_j) \\ \geq \quad M(\{p_j \leftarrow \mathcal{M}(\alpha_j)\}_{j\in I}[\mathcal{P}([\![\phi_i]\!](v[0/X_{i+1}]))])(p_j) \end{array}$$

and for all $j \notin I$

$$\begin{array}{l} M(\{q_j \leftarrow \mathcal{M}(\alpha_j)\}_{j\notin I}[\mathcal{P}([\![\phi_{i+1}]\!](v)[0/X_{i+1}])])(q_j) \\ \leq \quad M(\{q_j \leftarrow \mathcal{M}(\alpha_j)\}_{j\notin I}[\mathcal{P}([\![\phi_i]\!](v[0/X_{i+1}]))])(q_j) \end{array}$$

Let t be the valuation given by

$$\begin{cases} t(p_j) = M(\{p_j \leftarrow \mathcal{M}(\alpha_j)\}_{j\in I}[\mathcal{P}([\![\phi_i]\!](v[0/X_{i+1}]))])(p_j) & \text{if } j \in I \cup \{p_{i+1}\} \\ t(q_j) = M(\{q_j \leftarrow \mathcal{M}(\alpha_j)\}_{j\notin I}[\mathcal{P}([\![\phi_i]\!](v[0/X_{i+1}]))])(q_j) & \text{if } j \notin I \end{cases}$$

Thus for all $j \in I \cup \{i+1\}$, $t(p_j) \leq w(p_j)$ and for all $j \notin I$, $t(q_j) \geq w(q_j)$. Further, since all occurances of any p_j in the equations for q_k is in the negated form and vice versa, t is given by the following:

$$M(\left\{ \begin{array}{l} \{p_j \leftarrow \mathcal{M}(\alpha_j)\}_{j\in I} \\ \{q_j \leftarrow \mathcal{M}(\alpha_j)\}_{j\notin I} \end{array} \right\} [\mathcal{P}([\![\phi_i]\!](v[0/X_{i+1}]))] \wr \mathcal{P}([\![\phi_i]\!](v[0/X_{i+1}])))$$

Thus, t is given by:

$$\begin{array}{l} M(\mathcal{P}(\phi_i)[\mathcal{P}([\![\phi_i]\!](v[0/X_{i+1}]))] \wr \mathcal{P}([\![\phi_i]\!](v[0/X_{i+1}]))) \\ = M(\mathcal{P}(\phi_i)[\mathcal{P}(v[0/X_{i+1}])] \wr \mathcal{P}([\phi_i]\!](v[0/X_{i+1}]))) \end{array}$$

Thus, by induction hypothesis, $t = \mathcal{P}([\![\phi_i]\!](v[0/X_{i+1}]))$. But, from the fact that $[\![\phi_{i+1}]\!](v)(X_{i+1}) = 1$ and the definition of $\mathcal{P}$ it is easy to see that

$$M((\{p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1})\}[\mathcal{P}([\![\phi_{i+1}]\!](v))] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)))[t])(p_{i+1}) \;=\; 1.$$

But, for all $j \in I \cup \{i+1\}$, $t(p_j) \leq u(p_j)$ and for all $j \notin I$, $t(q_j) \geq u(q_j)$. Since, p_{i+1} depends positively on p_j it follows that

$$M((\{p_{i+1} \leftarrow \mathcal{M}(\alpha_{i+1})\}[\mathcal{P}([\![\phi_{i+1}]\!](v))] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)))[u])(p_{i+1}) \;=\; 1.$$

Thus, by Proposition B.1,

$$M(\mathcal{P}(\phi_{i+1})[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)))(p_{i+1}) \;=\; 1.$$

Thus,

$$M(\mathcal{P}(\phi_{i+1})[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)))(p_{i+1}) \;=\; \mathcal{P}([\![\phi_{i+1}]\!](v))(p_{i+1}).$$

Let $r_j = p_j$ if $\sigma(j) = \mu$ and q_j otherwise. Now, for any j, $1 \leq j \leq i$,

$$
\begin{array}{ll}
 & M(\mathcal{P}(\phi_{i+1})[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)))(r_j) \\
= & M((\mathcal{P}(\phi_i)[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v))) \\
 & \quad [\mathcal{P}([\![\phi_{i+1}]\!](v))[M(\mathcal{P}(\phi_{i+1})[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)))(p_{i+1})/p_{i+1}]])(r_j) \\
 & \{ \text{ By Proposition B.1 } \} \\
= & M((\mathcal{P}(\phi_i)[\mathcal{P}(v)] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)))[\mathcal{P}([\![\phi_{i+1}]\!](v))])(r_j) \\
= & M((\mathcal{P}(\phi_i)[\mathcal{P}([\![\phi_{i+1}]\!](v))] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)))[\mathcal{P}([\![\phi_{i+1}]\!](v))])(r_j) \\
= & M(\mathcal{P}(\phi_i)[\mathcal{P}([\![\phi_{i+1}]\!](v))] \wr \mathcal{P}([\![\phi_{i+1}]\!](v)))(r_j) \\
= & M(\mathcal{P}(\phi_i)[\mathcal{P}([\![\phi_i]\!](v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}]))] \wr \\
 & \quad \mathcal{P}([\![\phi_i]\!](v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}])))(r_j) \\
= & M(\mathcal{P}(\phi_i)[\mathcal{P}(v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}])] \wr \mathcal{P}([\![\phi_i]\!](v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}])))(r_j) \\
= & \mathcal{P}([\![\phi_i]\!](v[[\![\phi_{i+1}]\!](v)(X_{i+1})/X_{i+1}]))(r_j) \\
 & \{ \text{ By Induction Hypothesis } \} \\
= & \mathcal{P}([\![\phi_{i+1}]\!](v))(r_j)
\end{array}
$$

Thus, we have proved that $\mathcal{P}([\![\phi_{i+1}]\!](v))$ is a stable model of $\mathcal{P}(\phi_i)[\mathcal{P}(v)]$. This completes the proof of the theorem.