Counterexample-Driven Model Checking*

(Extended Abstract)

Natarajan Shankar and Maria Sorea**

SRI International
Computer Science Laboratory
333 Ravenswood Avenue
Menlo Park, CA 94025, USA
{shankar, sorea}@csl.sri.com
http://www.csl.sri.com/

Abstract. The generation of counterexamples is frequently touted as one of the primary advantages of model checking as a verification technique. However, the generation of trace-like counterexamples is limited to a small fragment of branching-time temporal logic. When model checking does succeed in verifying a property, there is typically no independently checkable witness that can be used as evidence for the verified property. We present a definition of witnesses, and, dually, counterexamples, for computation-tree logic (CTL), and describe a model checking algorithm that is based on the generation of evidence. Our model checking algorithm is local in the sense that it explores only the reachable states. It partitions the given initial set of states into those that do, and those that do not satisfy the given property, with a corresponding witness and counterexample that is independently verifiable. We have built a model checker based on these ideas that works quite efficiently despite the overhead of generating evidence.

1 Introduction

Model checking verifies a temporal property of a system by exploring its computation graph in explicit or symbolic terms. As a verification technique, state-space exploration has several advantages. Properties can be verified by visiting only the reachable states in the system. Counterexamples in the form of computation traces can be generated when a property fails to hold. There is no need to strengthen the properties being verified as is often the case with deductive verification. Various reduction and approximation techniques can be used to reduce the size of the explored state space. However, the above advantages are not always realizable in practice. Only a limited class of properties have trace

* Funded by SRI International, by NSF Grant CCR-0082560, DARPA/AFRL Contract F33615-00-C-3043, and NASA Contract NAS1-20334. A more complete draft technical report and the source and binary code for the model checker based on this paper can be obtained from the authors on request.

** Also affiliated with University of Ulm, Germany.

counterexamples. In practice, few temporal properties are checked solely by exploring the reachable state space. When model checking succeeds, there is often no witness that independently justifies the relationship between the model and the property.

We present an approach to symbolic model checking CTL (Computation-Tree Logic) formulas that overcomes these limitations. The algorithm explores the reachable state space and generates a witness for those states where the property holds and a counterexample where the property fails. These witnesses and counterexamples can be independently verified. The key to our approach is the use of *symbolic* witnesses and counterexamples that are constructed as trees over sets of states. Our model checking algorithm works by induction on the structure of the given CTL formula. Each temporal connective is evaluated by means of a forward unfolding of the state space followed by a fixpoint computation. The witness or counterexample can then be computed using the combination of the fixpoint set and the unfolded sequence of computation states. The correctness of the witness or counterexample can be independently verified. We have implemented a model checker based on witness/counterexample construction and have compared it with the standard fixpoint approach that does not yield witnesses or counterexamples. The efficiency of this model checker compares favorably with the more conventional methods despite the overhead of collecting and generating evidence.

Our model checking approach generates a directed unfolding of the symbolic state graph according to the temporal property being verified. The actual form of this unfolding for each temporal operator can be systematically derived from its fixpoint definition. The fixpoints can then be computed relative to the state sets generated in the unfolding phase. We do not actually compute the set of reachable states but only a subset of it that is relevant to evaluating the temporal property on the set of initial states.

The remainder of this paper is organized as follows. The syntax and semantics of CTL are covered in Section 2 along with an overview of our approach to generating counterexamples and witnesses during model checking. The details of the model checking algorithm are given in Section 3. A characterization of the witnesses and counterexamples is given in Section 4 along with the correctness argument for the model checking algorithm. We close with some observations in Section 5, a comparison to the related work in Section 6, and some concluding remarks in Section 7.

2 Preliminaries

Let $\mathbf{AP}$ be a set of atomic propositions, and $p, \bar{p} \in \mathbf{AP}$ ($\bar{p}$ stands for the negation of p). The formulas of the CTL logic in negation normal form are inductively defined as follows:

$$\begin{aligned}\varphi := p \mid \bar{p} \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{EX}\varphi \mid \mathbf{EF}\varphi \mid \mathbf{EG}\varphi \mid \mathbf{E}[\varphi_1 \,\mathbf{U}\, \varphi_2] \mid \mathbf{E}[\varphi_1 \,\mathbf{R}\, \varphi_2] \mid \\ \mathbf{AX}\varphi \mid \mathbf{AF}\varphi \mid \mathbf{AG}\varphi \mid \mathbf{A}[\varphi_1 \,\mathbf{U}\, \varphi_2] \mid \mathbf{A}[\varphi_1 \,\mathbf{R}\, \varphi_2]\end{aligned}$$

The semantics of a CTL formula is defined with respect to a *Kripke structure.* A Kripke structure is a tuple $\mathbf{M} = \langle \mathbf{AP}, \mathbf{S}, \mathbf{N} \rangle$ where $\mathbf{AP}$ is a finite set of atomic propositions, $\mathbf{S}$ is a finite set of states which we take as[1] $2^{\mathbf{AP}}$, the power-set of $\mathbf{AP}$, and $\mathbf{N} \in 2^{\mathbf{S} \times \mathbf{S}}$ is the transition relation. For $(s, s') \in \mathbf{N}$, we also write $s' \in \mathbf{N}(s)$, and if $S \subseteq \mathbf{S}$, then $\mathbf{N}(S)$ is $\bigcup_{s \in S} \mathbf{N}(s)$. The converse transition relation $\widetilde{\mathbf{N}}$ is defined by $\widetilde{\mathbf{N}}(s, s') \iff \mathbf{N}(s', s)$. We assume that the transition relation $\mathbf{N}$ is total, that is, every state has a successor. A *path* π is an infinite sequence of states $\pi = (s_0, s_1, \ldots)$ such that $s_{i+1} \in \mathbf{N}(s_i)$ for all $i \geq 0$.

Intuitively, the formula $\mathbf{EX}\varphi$ holds in a state s of $\mathbf{M}$ iff there exists a successor s' of s such that φ is true in s'. The formula $\mathbf{EG}\varphi$ will be true at a state s if there is a path starting at s such that φ holds at each state on the path. An *existential until* formula $\mathbf{E}[\varphi_1 \, \mathbf{U} \, \varphi_2]$ holds in some state s iff on some path starting from s, φ_1 holds until φ_2 holds. Similarly, a *universal until* formula $\mathbf{A}[\varphi_1 \, \mathbf{U} \, \varphi_2]$ holds in s if this conditions holds for all paths from s. The release operator $\mathbf{R}$ is the logical dual of $\mathbf{U}$. It requires that φ_2 holds along the path up to and including the first state where φ_1 holds. However, φ_1 is not required to hold eventually. The full semantics of CTL is described in Appendix A.

Given a set $S \subseteq \mathbf{S}$ and the transition relation $\mathbf{N}$, we define three *predicate transformers* $\mathit{post}(\mathbf{N})(S)$, $\mathit{pre}(\mathbf{N})(S)$, and $\widetilde{\mathit{pre}}(\mathbf{N})(S)$ from $2^{\mathbf{S}}$ to $2^{\mathbf{S}}$ as follows.

$$\begin{aligned} \mathit{post}(\mathbf{N})(S) &:= \mathbf{N}(S) \\ \mathit{pre}(\mathbf{N})(S) &:= \widetilde{\mathbf{N}}(S) \\ \widetilde{\mathit{pre}}(\mathbf{N})(S) &:= \{s \in \mathbf{S} \mid N(s) \subseteq S\} \end{aligned}$$

The *postcondition* function $\mathit{post}(\mathbf{N})(S)$ computes for a given set S of states, the set of states that can be reached in one step from some state in S. The *preimage* function $\mathit{pre}(\mathbf{N})(S)$ returns the set of states that can reach S in a single step. The *precondition* function $\widetilde{\mathit{pre}}(\mathbf{N})(S)$ returns the set of those states that have no successors outside of S.

The propositional μ-calculus [Koz83] provides a least fixpoint operator (μ) and a greatest fixpoint operator (ν), which make it possible to characterize the temporal operators of CTL. Clarke and Emerson [CE81] give the following fixpoint characterizations of the nontrivial CTL operators.

$$\begin{aligned} \mathbf{AF}\varphi &= \mu Z.\, \varphi \vee \mathbf{AX} Z \\ \mathbf{EF}\varphi &= \mu Z.\, \varphi \vee \mathbf{EX} Z \\ \mathbf{AG}\varphi &= \nu Z.\, \varphi \wedge \mathbf{AX} Z \\ \mathbf{EG}\varphi &= \nu Z.\, \varphi \wedge \mathbf{EX} Z \\ \mathbf{A}[\varphi_1 \, \mathbf{U} \, \varphi_2] &= \mu Z.\, \varphi_2 \vee (\varphi_1 \wedge \mathbf{AX} Z) \\ \mathbf{E}[\varphi_1 \, \mathbf{U} \, \varphi_2] &= \mu Z.\, \varphi_2 \vee (\varphi_1 \wedge \mathbf{EX} Z) \\ \mathbf{A}[\varphi_1 \, \mathbf{R} \, \varphi_2] &= \nu Z.\, \varphi_2 \wedge (\varphi_1 \vee \mathbf{AX} Z) \\ \mathbf{E}[\varphi_1 \, \mathbf{R} \, \varphi_2] &= \nu Z.\, \varphi_2 \wedge (\varphi_1 \vee \mathbf{EX} Z) \end{aligned}$$

[1] We are assuming that the states are represented using a binary encoding whose Boolean variables are in $\mathbf{AP}$.

Symbolic model checking [BCM+92,McM93,CGP99] establishes the satisfaction of a temporal property φ over a Kripke model $\mathbf{M}$ with respect to a set of initial states I. Given a set of initial states I, the notation $\mathbf{M}, I \models \varphi$ is interpreted as $\forall s \in I.\, \mathbf{M}, s \models \varphi$. For example, to verify $\mathbf{M}, I \models \mathbf{AF}p$, we represent the set of initial states I as a binary decision diagram (BDD) $[\![I]\!]$ over the Boolean variables in $\mathbf{AP}$. The next state relation $\mathbf{N}$ can also be represented as a BDD $[\![\mathbf{N}]\!]$ over primed and unprimed versions of the Boolean variables in $\mathbf{AP}$. The states in $\mathbf{S}$ satisfying the formula $\mathbf{AF}p$, written as $[\![\mathbf{AF}p]\!]$, can then be computed as a BDD given by $[\![\mathbf{AF}p]\!]_k$ for the least k such that $[\![\mathbf{AF}p]\!]_k = [\![\mathbf{AF}p]\!]_{k+1}$, where $[\![\mathbf{AF}p]\!]_0 = [\![p]\!]$ and $[\![\mathbf{AF}p]\!]_{i+1} = \widetilde{pre}(\mathbf{N})([\![\mathbf{AF}p]\!]_i)$. We can then check whether the BDD $[\![I]\!] \wedge \neg[\![\mathbf{AF}p]\!]$ is satisfiable. If so, we have a counterexample, and if not, we have $\mathbf{M}, I \models \mathbf{AF}p$. Similar fixpoint model-checking algorithms can be given for the other temporal operators based on the fixpoint definitions with the predicate transformers $pre(\mathbf{N})(S)$ or $\widetilde{pre}(\mathbf{N})(S)$ above.

Our model-checking algorithm also employs fixpoints over predicate transformers while computing information from which witnesses or counterexamples can be easily extracted. It can be seen as a refinement of the above naïve computation of the fixpoint. Taking the example of $\mathbf{AF}p$, the *bad* states $\neg[\![\mathbf{AF}p]\!]$ can be computed as the greatest fixpoint $\nu Z.\, [\![\neg p]\!] \wedge pre(\mathbf{N})(Z)$. Let R be given by $\mu Z.\, [\![I]\!] \vee post(\mathbf{N})(Z)$ which is the set of states *reachable* from initial state set I. It can be shown that $[\![I]\!] \wedge \nu Z.\, [\![\neg p]\!] \wedge pre(\mathbf{N})(Z)$ is satisfiable iff $[\![I]\!] \wedge \nu Z.\, [\![\neg p]\!] \wedge R \wedge pre(\mathbf{N})(Z)$ is satisfiable. We have thus narrowed the starting point of the fixpoint iteration from $[\![\neg p]\!]$ to $[\![\neg p]\!] \wedge R$, but we can do even better. Let W represent the set of states given by $\nu Z.\, [\![\neg p]\!] \wedge R \wedge pre(\mathbf{N})(Z)$. The set $I \wedge W$ is equivalent to $I \wedge \nu Z.\, R^- \wedge pre(\mathbf{N})(Z)$, where R^- is a subset of $[\![\neg p]\!] \wedge R$ given by $\mu Z.\, ([\![\neg p]\!] \wedge [\![I]\!]) \vee ([\![\neg p]\!] \wedge post(\mathbf{N})(Z))$.

The model checking of $\mathbf{AF}p$ thus consists of a phase where R^- is computed in a forward iteration using $post(\mathbf{N})$, followed by a backward iteration phase where W is computed relative to R^- using $pre(\mathbf{N})$. The iterative computation of R^- can be carried out as $V_0 = I$, $V_{i+1} = post(\mathbf{N})(R_i^- - V_{<i})$, and $R_i^- = V_i \wedge [\![\neg p]\!]$, where $V_{<i} = \bigcup_{j<i} V_j$ and $X - Y$ is defined as $X \wedge \neg Y$. The iterative computation of W is given by W_m for the least m such that $W_{m+1} = W_m$, where $W_0 = R^-$ and $W_{i+1} = pre(\mathbf{N})(W_i) \cap W_i$. The resulting algorithm has two advantages. The greatest fixpoint computation of W takes place over a set of states that is smaller than all states or even all reachable states. Computing the greatest fixpoint from a smaller initial set may not guarantee smaller BDDs, but this does indeed appear to be the case on nontrivial examples. If $I \cap W$ is nonempty, a counterexample trace can be constructed by picking a computation sequence $s_0, \ldots, s_k$ from $R_0^- \wedge W, \ldots, R_k^- \wedge W$. If $R_0^- \wedge W$ is empty, then the property is verified. The witness for this is given by the sequence of state sets $G_0, \ldots, G_m$, where $G_i = V - W_i$, for $i \leq m$. It can be shown that if $R_0^- \wedge W$ is empty, then W must be empty, so $V - W_m = V$. It is clear that this is a valid witness for $\mathbf{AF}p$ since it can be shown that $G_0 \subseteq [\![p]\!]$, $[\![I]\!] \subseteq G_m$, and $G_{i+1} \subseteq \widetilde{pre}(\mathbf{N})(G_i) \cup G_i$.

$$
\begin{aligned}
\mathbf{WMC}(\varphi, I, \mathbf{N}) = \text{case } \varphi \text{ of}\\
p &: \mathbf{PMC}(p, I, \mathbf{N})\\
\bar{p} &: \overline{\mathbf{PMC}}(\bar{p}, I, \mathbf{N})\\
\mathbf{EX}\varphi &: \mathbf{EXMC}(\varphi, I, \mathbf{N}, \emptyset, \emptyset)\\
\mathbf{EF}\varphi &: \mathbf{EFMC}(\varphi, I, \mathbf{N}, \emptyset, \emptyset)\\
\mathbf{EG}\varphi &: \mathbf{EGMC}(\varphi, I, \mathbf{N}, \emptyset, \emptyset)\\
\mathbf{E}[\varphi_1 \,\mathbf{U}\, \varphi_2] &: \mathbf{EUMC}(\varphi_1, \varphi_2, I, \mathbf{N}, \emptyset, \emptyset, \emptyset)\\
\mathbf{E}[\varphi_1 \,\mathbf{R}\, \varphi_2] &: \mathbf{ERMC}(\varphi_1, \varphi_2, I, \mathbf{N}, \emptyset, \emptyset, \emptyset)\\
\mathbf{AX}\varphi &: \mathbf{AXMC}(\varphi, I, \mathbf{N}, \emptyset, \emptyset)\\
\mathbf{AF}\varphi &: \mathbf{AFMC}(\varphi, I, \mathbf{N}, \emptyset, \emptyset)\\
\mathbf{AG}\varphi &: \mathbf{AGMC}(\varphi, I, \mathbf{N}, \emptyset, \emptyset)\\
\mathbf{A}[\varphi_1 \,\mathbf{U}\, \varphi_2] &: \mathbf{AUMC}(\varphi_1, \varphi_2, I, \mathbf{N}, \emptyset, \emptyset, \emptyset)\\
\mathbf{A}[\varphi_1 \,\mathbf{R}\, \varphi_2] &: \mathbf{ARMC}(\varphi_1, \varphi_2, I, \mathbf{N}, \emptyset, \emptyset, \emptyset)
\end{aligned}
$$

Fig. 1. Model Checking CTL.

3 A Model-Checking Algorithm

As above, let $\mathbf{M} = \langle \mathbf{AP}, \mathbf{S}, \mathbf{N} \rangle$ be a Kripke structure, where $\mathbf{AP}$ is a finite set of atomic propositions, $\mathbf{S}$ a finite set of states, and $\mathbf{N}$ the (total) transition relation. The top level of the algorithm **WMC** is shown in Figure 1. It invokes individual operations corresponding to the top-level connective. The cases corresponding to the propositional connectives have been elided. We discuss the special cases of the CTL operators **EG** and **AG**. The model-checking algorithm for the entire CTL logic can be found in Appendix B. The model checker takes as input a CTL formula φ, and a set of initial states I. The output O is a pair of lists $\langle [U_0, \ldots, U_k], [W_0, \ldots, W_m] \rangle$. Each U_i is of the form $\langle S_i, B_i, O'_i, O''_i \rangle$, where $S_0 = I$, B_i is the subset of S_i violating φ, and O'_i and O''_i are the outputs corresponding to the immediate subformulas of φ with respect to S_i. In the case of unary CTL connectives, the field O''_i will be empty, which is represented as $\_$. We will refer to the fields of U_i as $U_i.S$, $U_i.B$, $U_i.O'$, and $U_i.O''$, respectively. Each W_i is a set of states representing the stages in a fixpoint computation.

The **EGMC** algorithm for $\mathbf{EG}\varphi$ is displayed in Figure 2. When we write $\vec{W}$, we mean a list of elements of the form $[W_0, \ldots]$, and $\vec{W}^m$ implies that the list $\vec{W}$ is of length $m + 1$, that is, of the form $[W_0, \ldots, W_m]$. The notation $[X; \vec{W}]$ represents the list obtained by adding element X to the front of the list of $\vec{W}$.

In addition to the formula φ and the set of initial states I, **EGMC** takes as input, two sets of states, V and V^+. The set V collects the visited states, while $V^+ \subseteq V$ contains those states that satisfy the subformula φ. Initially, both V and V^+ are empty. First, the subformula φ is processed over the given set of initial states I (Figure 2, line (2)). This yields the output O' whose components

$$\mathbf{EGMC}(\varphi, I, \mathbf{N}, V, V^{+}) = \tag{1}$$
$$\text{let } O' = \mathbf{WMC}(\varphi, I, \mathbf{N}); \tag{2}$$
$$\langle \vec{U'}, \vec{W'} \rangle = O'; \tag{3}$$
$$I' = I - (V \cup U'_0.B) \tag{4}$$
$$\text{in (if } I' = \emptyset \tag{5}$$
$$\text{then} \tag{6}$$
$$\text{let } \vec{W}^m = pre(\mathbf{N})^\wedge(V^{+}) \tag{7}$$
$$\text{in } S = \langle [\langle I, I - W_m, O', \_ \rangle], \vec{W} \rangle \tag{8}$$
$$\text{else} \tag{9}$$
$$\text{let } \langle \vec{U}, \vec{W}^m \rangle = \mathbf{EGMC}(\varphi, post(\mathbf{N})(I'), \mathbf{N}, V \cup I, V^{+} \cup I', \mathbf{N}) \tag{10}$$
$$\text{in } \langle [\langle I, I - W_m, O', \_ \rangle; \vec{U}], \vec{W} \rangle \tag{11}$$
$$\text{endif)} \tag{12}$$

Fig. 2. Model Checking $\mathbf{EG}\varphi$.

are the lists $\vec{U'}$ and $\vec{W'}$ (line (3)). Since $U'_0.B$ consists of the states in I where φ fails to hold, the set $I - (V \cup U'_0.B)$ are the new states where φ holds. This set is labelled I' (line (4)). The set I' could also be computed as I' is $I - U'_0.B$, where O' of the form $\langle \vec{U'}, \vec{W'} \rangle$ is $\mathbf{WMC}(\varphi, I - V, \mathbf{N})$. If I' is empty, the algorithm terminates (lines (5) to (8)). The output returned in line (8) consists of the list $\vec{W}$ of length $m+1$ recording the stages in the computation of $\nu Z.\, V^{+} \wedge pre(\mathbf{N})(Z)$ as generated by $pre(\mathbf{N})^\wedge(V^{+})$ (in line (7)) which is defined to return the list $\vec{W}^m$, where $W_0 = V^{+}$, $W_{i+1} = W_i \cap pre(\mathbf{N})(W_i)$, and $W_m = pre(\mathbf{N})(W_m)$. The U part of the output in this base case is the singleton list $[\langle I, I - W_m, O', \_ \rangle]$. Otherwise **EGMC** is invoked recursively on a new set of initial states $post(\mathbf{N})(I')$, and updated values for V and V^{+} (line (10)). The resulting output $\langle \vec{U}, \vec{W}^m \rangle$ is used to construct the output in line (11) where the tuple $\langle I, I - W_m, O', \_ \rangle$ has been added to the front of $\vec{U}$. The algorithm always terminates since we never reexamine a previously visited state, and the state space is finite.

Example 1. Given the Kripke model in Figure 3, it is easy to see that the formula $\mathbf{EG}p$ is not satisfied in the initial state 0. The recursive invocations of **EGMC** yield the entries in the following table.

Step	I	V	V^{+}	I'
0	$\{0\}$	$\emptyset$	$\emptyset$	$\{0\}$
1	$\{1, 2\}$	$\{0\}$	$\{0\}$	$\{1\}$
2	$\{2\}$	$\{0, 1, 2\}$	$\{0, 1\}$	$\emptyset$

The sequences O'_i ($i = 0, 1, 2$) contain information about the satisfiability of the subformula p during the three computation steps.

$$O'_0 = \langle [\langle \{0\}, \emptyset, \_, \_ \rangle], [\emptyset] \rangle$$

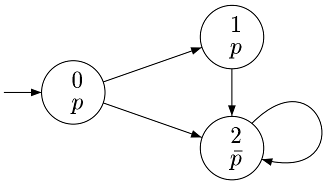

Fig. 3. Model Checking $\mathbf{EG}p$ – Example.

$$O'_1 = \langle [\langle \{1,2\}, \{2\}, \_, \_ \rangle], [\{2\}] \rangle$$
$$O'_2 = \langle [\langle \{2\}, \{2\}, \_, \_ \rangle], [\{2\}] \rangle$$

The final value of V^+ is $\{0,1\}$ and $\vec{W}$ is a list of subsets of V^+ computed by $pre(\mathbf{N})^\wedge(V^+)$ to be $[\{0,1\}, \{0\}, \emptyset]$. Here W, given by W_2, is empty since each of the three states eventually reaches state 2, where p does not hold, along every path. Finally, the algorithm returns the pair $\langle \vec{U}_0, \vec{W} \rangle$, where $\vec{U}_0$ is

$$[\langle \{0\}, \boxed{\{0\}}, O'_0, \_ \rangle,\ \langle \{1,2\}, \{1,2\}, O'_1, \_ \rangle,\ \langle \{2\}, \{2\}, O'_2, \_ \rangle].$$

Since $0 \in U_0.B$, the formula $\mathbf{EG}p$ does not hold of the state 0 in the given Kripke model. A counterexample is produced as $[C_0, C_1, C_2]$, where $C_i = V - W_i$, which yields $[\{2\}, \{1,2\}, \{0,1,2\}]$. Note that $C_{i+1} \subseteq C_i \cup \widetilde{pre}(\mathbf{N})(C_i)$ for $i < 2$, and $[\![I]\!] \subseteq C_2$.

Figure 4 illustrates the **AGMC** algorithm for the case $\mathbf{AG}\varphi$. This algorithm is similar to its existential counterpart **EGMC**. The main difference is that the set of good states is computed using the precondition transformer $\widetilde{pre}$ instead of pre, as is the case for the **EG** operator. These basic algorithms can be optimized in small ways. For example, when $V = V^+$ in line (7), the fixpoint computation of $\vec{W}^m$ is redundant. Such specialized optimizations are not covered in this paper.

4 Witnesses and Counterexamples

We address the problem of characterizing the witnesses and counterexamples generated by the model-checking algorithm, restricting our discussion to the CTL operators **EG** and **AG**.

Given a Kripke structure $\mathbf{M} = \langle \mathbf{AP}, \mathbf{S}, \mathbf{N} \rangle$, a CTL formula φ, and a set of initial states I, the model-checking algorithm presented in the previous chapter returns output O as a pair of sequences $\langle \vec{U}^k, \vec{W}^m \rangle$, where each U_i has the form $\langle S, B, O', O'' \rangle$. The values corresponding to these fields of U_i are abbreviated as S_i, B_i, O'_i, and O''_i, respectively. The set S_0 is the set of initial states I, and the set B_i consists of those states in S_i that violate the property φ.

Informally, a witness w for a formula $\mathbf{EG}\varphi$ with respect to a state s_0 is a sequence of the form $[\langle s_0, w_0 \rangle, \ldots, \langle s_n, w_n \rangle]$, where $s_n = s_i$ for some $i < n$,

$$
\begin{array}{l}
\mathbf{AGMC}(\varphi, I, \mathbf{N}, V, V^{+}) = \\
\quad \text{let } O' = \mathbf{WMC}(\varphi, I, \mathbf{N}); \\
\qquad \langle \vec{U}', \vec{W}' \rangle = O'; \\
\qquad I' = I - (V \cup U'_0.B) \\
\qquad \text{in (if } I' = \emptyset \\
\qquad\quad \text{then} \\
\qquad\qquad \text{let } \vec{W}^m = \widetilde{pre}^{\wedge}(\mathbf{N})(V^{+}) \\
\qquad\qquad \text{in } S = \langle [\langle I, I - W_m, O', \_ \rangle], \vec{W} \rangle \\
\qquad\quad \text{else} \\
\qquad\qquad \text{let } \langle \vec{U}, \vec{W}^m \rangle = \mathbf{AGMC}(\varphi, post(\mathbf{N})(I'), \mathbf{N}, V \cup I, V^{+} \cup I') \\
\qquad\qquad \text{in } \langle [\langle I, I - W_m, O', \_ \rangle; \vec{U}], \vec{W} \rangle \\
\qquad\quad \text{endif)}
\end{array}
$$

Fig. 4. Model Checking $\mathbf{AG}\varphi$.

$\mathbf{N}(s_i, s_{i+1})$ holds for $i < n$, and w_i is a witness for φ with respect to s_i for $i \leq n$. However, this kind of witness, for a specific state s_0 is not really useful for our purpose even though it can be easily constructed from the output O returned by $\mathbf{EGMC}(\varphi, I, \mathbf{N}, \emptyset, \emptyset)$. We need a witness that carries the justification that $\mathbf{M}, I \models \mathbf{EG}\varphi$. More generally, each execution of the model checker partitions I into states I^{+}, where the property holds, and states I^{-}, where the property fails. We need a witness w for $\mathbf{M}, I^{+} \models \mathbf{EG}\varphi$, and a counterexample c for $\mathbf{M}, I^{-} \not\models \mathbf{EG}\varphi$.

We first describe the form and characterization of a witness for a formula $\mathbf{EG}\varphi$ independent of its construction. We write $w \vdash \mathbf{M}, G \models \mathbf{EG}\varphi$ to indicate that w is a witness for the judgement that the states in G satisfy $\mathbf{EG}\varphi$. The witness w is a pair $\langle \vec{X}^m, \vec{w}^k \rangle$ of lists of lengths m and k, respectively, where

1. There exist $G_0, \ldots, G_k$ such that $w_i \vdash \mathbf{M}, G_i \models \varphi$, for $i \leq k$
2. $X_0 \subseteq \bigcup_{i=0}^{k} G_i$
3. $X_{i+1} \subseteq X_i$, for $i < m$
4. $X_m \subseteq pre(\mathbf{N})(X_m)$
5. $G \subseteq X_m$

It should be intuitively clear that this is a valid witness for $\mathbf{M}, G \vdash \mathbf{EG}\varphi$ since $G \subseteq X_m$ and for any state $s \in X_m$, we have that $s \in [\![\varphi]\!]$ and $\mathbf{N}(s) \cap G \neq \emptyset$, i.e., there is a state $s' \in \mathbf{N}(s)$ such that $s' \in G$. In other words, from any state in X_m, we can find an infinite path consisting of states from within X_m. The above conditions can be expressed in CTL itself so that it is easy to extract a CTL proof structure from the witness. Clauses 2 and 3 in the above definition can be eliminated in favor of the condition that $X_m \subseteq \bigcup_{i=0}^{k} G_i$, but we retain the intermediate states X_i for the sake of uniformity with the other CTL operators.

We now have to show that the output O returned by $\mathbf{EGMC}(\varphi, I, \mathbf{N}, \emptyset, \emptyset)$ can be used to construct a witness meeting the above conditions. The top-level operation for constructing a witness for a formula φ and output O from $\mathbf{WMC}(\varphi, I, \mathbf{N})$ has the form $\mathit{witness}(\varphi, O)$. The operation $\mathbf{WEG}(\varphi, \mathbf{N}, O)$ constructs a generic witness w for $\mathbf{EG}\varphi$ as $\langle \vec{W}^m, \vec{w}^k \rangle$, where $w_i = \mathit{witness}(\varphi, O'_i)$ for $i \leq k$.

The notation $c \vdash \mathbf{M}, C \not\models \mathbf{EG}\varphi$ indicates that c is a counterexample demonstrating that for every $s \in C$, $\mathbf{M}, s \not\models \mathbf{EG}\varphi$. A counterexample c justifying $\mathbf{M}, C \not\models \mathbf{EG}\varphi$ has the form $\langle \vec{X}^m, \vec{c}^k \rangle$, where

1. There exist state sets $C_0, \ldots, C_k$ such that $c_i \vdash \mathbf{M}, C_i \not\models \varphi$, for $i \leq k$
2. $X_0 = \bigcup_{i=0}^k C_i$
3. $X_{i+1} \subseteq X_i \cup \widetilde{pre}(\mathbf{N})(X_i)$, for $i < m$
4. $C \subseteq X_m$

Note that a counterexample for $\mathbf{EG}\varphi$ must be a witness for $\mathbf{AF}\neg\varphi$. The conditions on c ensure that every computation originating from a state s in X_i must eventually reach a state in X_0, where $X_0 \subseteq [\![\neg\varphi]\!]$.

Once again, given $O = \mathbf{EGMC}(\varphi, I, \mathbf{N}, \emptyset, \emptyset)$, we can construct a counterexample c meeting the above conditions. The top-level operation for constructing counterexamples for a formula φ from $O = \mathbf{WMC}(\varphi, I, \mathbf{N})$ is $\mathit{counter}(\varphi, O)$. The operation $\mathbf{CEG}(\varphi, \mathbf{N}, O)$ constructs the counterexample c as $\langle [V - W_0, \ldots, V - W_m], \vec{c}^k \rangle$, where $V = \bigcup_{i=0}^k S_i$, and $c_i = \mathit{counter}(\varphi, O'_i)$ for $i \leq k$.

A witness w such that $w \vdash \mathbf{M}, G \models \mathbf{AG}\varphi$ has the form $\langle \vec{X}^m, \vec{w}^k \rangle$, where

1. There exist $G_0, \ldots, G_k$ such that $w_i \vdash \mathbf{M}, G_i \models \varphi$, for $i \leq k$
2. $X_0 \subseteq \bigcup_{i=0}^k G_i$
3. $X_{i+1} \subseteq X_i$, for $i < m$
4. $X_m \subseteq \widetilde{pre}(\mathbf{N})(X_m)$
5. $G \subseteq X_m$

These conditions ensure that $G \subseteq [\![\varphi]\!]$ and for any states s, s' such that $s \in G$ and $s' \in \mathbf{N}(s)$, we have $s' \in G$. In other words, G is a stable property of $\mathbf{M}$ that entails φ.

As in the case of $\mathbf{EG}\varphi$, a suitable witness w is constructed from the output $O = \mathbf{AGMC}(\varphi, I, \mathbf{N}, \emptyset, \emptyset)$ by $\mathbf{WAG}(\varphi, O)$ as $\langle \vec{W}^m, \vec{w}^k \rangle$, where $w_i = \mathit{witness}(\varphi, O'_i)$ for $i \leq k$.

A counterexample c, where $c \vdash \mathbf{M}, C \not\models \mathbf{AG}\varphi$, has the form $\langle \vec{X}^m, \vec{c}^k \rangle$, where

1. There exist state sets $C_0, \ldots, C_k$ such that $c_i \vdash \mathbf{M}, C_i \not\models \varphi$, for $i \leq k$
2. $X_0 = \bigcup_{i=0}^k C_i$
3. $X_{i+1} \subseteq X_i \cup \mathit{pre}(\mathbf{N})(X_i)$, for $i < m$
4. $C \subseteq X_m$

These conditions ensure that any state $s \in X_i$ must have a computation path leading to a state $s' \in X_0$ and $X_0 \subseteq [\![\neg\varphi]\!]$.

A counterexample c meeting the above conditions is constructed from $O = \mathbf{AGMC}(\varphi, I, \mathbf{N}, \emptyset, \emptyset)$ by $\mathbf{CAG}(\varphi, O)$ as $\langle [V - W_0, \ldots, V - W_m], \vec{c^k} \rangle$, where $V = \bigcup_{i=0}^{k} S_i$, and $c_i = \mathit{counter}(\varphi, O_i')$ for $i \leq k$.

We can verify the following theorem from the full characterization of CTL witnesses and counterexamples. The proof is by induction on the structure of the formula. The informal argument for the case of $\mathbf{EG}\varphi$ and $\mathbf{AG}\varphi$ has already been sketched. The remaining cases are similar.

Theorem 1 (Witness Validity). *If* $w \vdash \mathbf{M}, G \models \varphi$ *then* $\mathbf{M}, G \models \varphi$, *and if* $c \vdash \mathbf{M}, C \not\models \varphi$, *then* $\mathbf{M}, C \not\models \varphi$.

We can also establish that our model-checking algorithm yields valid witnesses and counterexamples. The following theorem is also proved by induction on the structure of φ by showing for each case of the definition of **WMC** that w is a valid witness and c is a valid counterexample. In each case, the sets G and C are constructed from output O so as to partition V and hence I.

Theorem 2 (Correctness). *Let* O *be* $\mathbf{WMC}(\varphi, I, \mathbf{N})$, *then there exist disjoint sets* G *and* C *such that* $I \subseteq G \cup C$, *a witness* $w = \mathit{Witness}(\varphi, \mathbf{N}, O)$, *and a counterexample* $c = \mathit{Counter}(\varphi, \mathbf{N}, O)$, *such that* $w \vdash \mathbf{M}, G \models \varphi$ *and* $c \vdash \mathbf{M}, C \not\models \varphi$.

5 Experiments

We have implemented our algorithm in Common Lisp using the Eindhoven BDD package [Jan93] as a C library. We have conducted preliminary experiments that suggest that our local model-checking algorithm constructs BDDs of smaller size and thus performs more efficiently in time and space than a model checker using the same BDD package but based on the standard fixpoint definitions of the CTL operators [BCM+92]. The performance comparison is given in Figures 5 for the mutual exclusion property of a buggy version of the synchronous arbiter from the SMV example suite. Figure 6 gives the comparison on the mutual exclusion property for a correct version of the arbiter circuit. The timings (given in milliseconds) do not include the time spent constructing the BDD representations for the set of initial states and the transition relation. They do include the time spent constructing the output O for the case of our model checker **WMC**, whereas the timings for the baseline model checker **MC** are simply those for checking the property. Note that when an **AG** property holds, the **WMC** model checker requires only one greatest fixpoint iteration since the set of states in $V^+ = V$ is already a fixpoint. Further timings are not given due to the slowness in constructing the BDD for the transition relation.

6 Related Work

Prior related work in this area includes results on counterexample generation, methods for certifying model checkers, and model-checking algorithms based on forward iteration.

	No. of cells	5	10	15	20	25	30	35	40	45	47
WMC	Max. BDD size	40	92	142	199	251	301	347	407	455	101
	Time (msecs)	0	130	80	229	290	470	530	660	880	150
	No. of iterations	3	3	3	3	3	3	2	3	3	2
MC	Max. BDD size	381	1041	1317	2295	4838	5533	5657	9976	6801	3338
	Time (msecs)	10	260	440	1210	3510	2500	1710	11360	4804	1330
	No. of iterations	3	4	3	5	4	3	3	4	3	3

Fig. 5. Performance comparison on mutual exclusion for buggy arbiter

	No. of cells	5	10	15	20	25	30	35	40	45	47
WMC	BDD size	25	38	62	86	102	121	137	162	181	96
	time	0	10	20	210	40	90	140	120	380	80
	No. of iterations	1	1	1	1	1	1	1	1	1	1
MC	BDD size	186	673	892	1265	2934	3086	3406	8164	3737	1647
	time (msecs)	20	140	320	250	890	1170	1260	4040	2890	540
	No. of iterations	3	3	3	4	4	4	4	4	4	3

Fig. 6. Performance comparison on mutual exclusion for correct arbiter

Clarke, Grumberg, McMillan, and Zhao [CGMZ95] present techniques for the efficient generation of counterexamples for fragments of ACTL and Fair ACTL (CTL with fairness constraints). Counterexample construction for the three CTL basic operators: **AX**, **AG**, and **AU**, are given, and the problem of finding fair counterexamples is classified as NP-complete. The algorithms for generating counterexamples have been implemented in the SMV model checker [McM93]. Kick [Kic96] shows that it is possible to construct tree-like counterexamples for the entire μ-calculus, but the resulting trees are large and quite complicated. Clarke, Jha, Lu, and Veith [CJLV02] investigate tree-like counterexamples for ACTL based on a backward and then forward exploration of the state space. Our model-checking algorithm, in contrast, avoids the computation of the entire reachable state space through a forward unfolding of the state space followed by a local fixpoint computation. It partitions the set of initial states into (possibly empty) good states with a corresponding symbolic witness, and (possibly empty) bad states with a corresponding symbolic counterexample.

Namjoshi [Nam01] introduced the notion of a *certifying model checker* that can generate independently checkable witnesses for properties verified by a model checker. He defined witnesses for properties of labelled transition systems expressed in the modal μ-calculus based on parity games over alternating tree automata. Peled, Pnueli, and Zuck [PPZ01] produce deductive proofs for successfully model checked LTL formulas based on identifying the strongly connected components in the model checking tableau and generating a proof for the absence of feasible paths. Gurfinkel and Chechik [GC03] present an approach for annotating model checker witnesses with proof steps and generating proof obligations that can be independently verified with a theorem prover. All of the above methods generate explanations only when the model checker achieves a verification or a refutation, whereas our approach produces simple and direct witnesses and counterexamples that partition the initial states into good and bad states, respectively. Such a partition is needed for the recursive invocation of the model checker on subformulas of the given formula which do not involve either verification or refutation.

Iwashita, Nakata, and Hirose [INH96] present a CTL model-checking algorithm based on forward state traversal for a fragment of CTL. They show that in many situations backward state traversal is more expensive than forward traversal. When combined with BDD-based state traversal techniques using partitioned transition relations, their method could be successfully applied for verifying large finite-state systems. Henzinger, Kupferman, and Qadeer [HKQ98] investigate the class of specifications that can be checked by symbolic forward state traversal and show that all ω-regular (linear-time) specifications can be verified using forward traversal. Biere, Clarke, and Zhu [BCZ99] give a local tableau construction for LTL model checking based on sets of states and forward image computation with explicit detection of strongly connected components in the tableau. We present a two-phase symbolic model checking algorithm for CTL consisting of a forward traversal that identifies relevant reachable states followed by a backward traversal that partitions the initial set of states into good and bad states.

7 Conclusions

We have given a simple local model-checking algorithm for CTL that constructs witnesses and counterexamples. The algorithm is based on a constrained, forward unfolding of the state space starting from the initial states, followed by a fixpoint computation. Symbolic witnesses and counterexamples are constructed from the results produced by the model checker. The witnesses and counterexamples have been independently characterized. The model-checking algorithm has been proved to produce valid witnesses and counterexamples. Preliminary experiments indicate that our method is significantly more efficient than the standard method used for CTL model checking despite the overhead of collecting evidence.

Our approach of generating both positive and negative evidence during model checking can be extended and refined in a number of directions. We need to construct an independent evidence checker for the witnesses and counterexamples produced by our model checker. If we are interested only in the absence of counterexamples, it is possible in some cases, to terminate as soon as a counterexample is found. We can also restrict our search to bounded length evidence by bounding the forward unfolding, but this may return only partial conclusions. Counterexample-driven model checking can be combined with abstraction to compute over and under-approximations of fixpoint properties. We are also investigating the application of our two-phase method to controller synthesis, where the backward fixpoint computation employs a *controlled* precondition operation. We also need to examine the effectiveness, both in terms of efficiency and the size and clarity of the evidence, when this approach is adapted to Fair CTL, LTL, and CTL*.

References

[BCM+92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.

[BCZ99] Armin Biere, Edmund M. Clarke, and Yunshan Zhu. Multiple state and single state tableaux for combining local and global model checking. In E. R. Olderog and B. Steffen, editors, *Correct System Design*, number 1710 in Lecture Notes in Computer Science, pages 163–179. Springer-Verlag, 1999.

[CE81] E.M. Clarke and E.A. Emerson. Characterizing Properties of Parallel Programs as Fixpoints. In *7th International Colloquium on Automata, Languages and Programming*, volume 85 of *Lecture Notes in Computer Science*. Springer-Verlag, 1981.

[CGMZ95] E.M. Clarke, O. Grumberg, K.L. McMillan, and X. Zhao. Efficient Generation of Counterexamples and Witnesses in Symbolic Model Checking. In *32nd Design Automation Conference (DAC 95)*, pages 427–432, San Francisco, CA, USA, June 1995.

[CGP99] E. M. Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press, 1999.

[CJLV02] Edmund M. Clarke, Somesh Jha, Yuan Lu, and Helmut Veith. Tree-like counterexamples in model checking. In *IEEE Symposium on Logic in Computer Science*, Lecture Notes in Computer Science. Springer Verlag, July 2002.

[GC03] Arie Gurfinkel and Marsha Chechik. Proof-like counter-examples. To appear in Proceedings of TACAS'03, May 2003.

[HKQ98] T. A. Henzinger, O. Kupferman, and S. Qadeer. From pre-historic to post-modern symbolic model checking. *Lecture Notes in Computer Science*, 1427:195–206, 1998.

[INH96] H. Iwashita, T. Nakata, and F. Hirose. CTL model checking based on forward state traversal. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 82–87, Washington, November 10–14 1996. IEEE Computer Society Press.

[Jan93] G. Janssen. *ROBDD Software*. Department of Electrical Engineering, Eindhoven University of Technology, October 1993.

[Kic96] Alexander Kick. *Generation of counterexamples and witnesses for the Mu-calculus*. PhD thesis, University of Karlsruhe, Germany, 1996.

[Koz83] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27(3):333–354, December 1983.

[McM93] Kenneth L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Boston, MA, 1993.

[Nam01] Kedar S. Namjoshi. Certifying model checkers. In *13th Conference on Computer Aided Verification (CAV)*, volume 2102 of *Lecture Notes in Computer Science*, pages 2–13. Springer Verlag, 2001.

[PPZ01] Doron Peled, Amir Pnueli, and Lenore Zuck. From falsification to verification. *Lecture Notes in Computer Science*, 2245:292–304, 2001.

Note to the reviewer: the appendix is not part of the official submission.

A Semantics of CTL

Given $\mathbf{M} = \langle \mathbf{AP}, \mathbf{S}, \mathbf{N} \rangle$ a Kripke structure. The notation $\mathbf{M}, s \models \varphi$ means that φ holds at state s in the Kripke structure $\mathbf{M}$. The relation "$\models$" is defined inductive as follows ($[\![p]\!]$ denotes the set of states where p holds):

$$
\begin{array}{rcl}
\mathbf{M}, s \models p & \text{iff} & s \in [\![p]\!] \\
\mathbf{M}, s \models \bar{p} & \text{iff} & s \notin [\![p[\![\\
\mathbf{M}, s \models \varphi \wedge \varphi_2 & \text{iff} & \mathbf{M}, s \models \varphi_1 \text{ and } \mathbf{M}, s \models \varphi_2 \\
\mathbf{M}, s \models \varphi \vee \varphi_2 & \text{iff} & \mathbf{M}, s \models \varphi_1 \text{ or } \mathbf{M}, s \models \varphi_2 \\
\mathbf{M}, s \models \mathbf{EX}\varphi & \text{iff} & \mathbf{M}, s' \models \varphi \text{ for some state } s' \in S \text{ such that } s' = \mathbf{N}(s) \\
\mathbf{M}, s \models \mathbf{EF}\varphi & \text{iff} & \text{for some path } \pi = (s_0, s_1, \ldots) \text{ with } s = s_0, \\
& & \text{there exists } i \geq 0 \text{ such that } \mathbf{M}, s_i \models \varphi \\
\mathbf{M}, s \models \mathbf{EG}\varphi & \text{iff} & \text{for some path } \pi = (s_0, s_1, \ldots) \text{ with } s = s_0, \\
& & \mathbf{M}, s_i \models \varphi \text{ for all } i \geq 0 \\
\mathbf{M}, s \models \mathbf{E}[\varphi_1 \, \mathbf{U} \, \varphi_2] & \text{iff} & \text{for some path } \pi = (s_0, s_1, \ldots) \text{ with } s = s_0, \\
& & \text{for some } i \geq 0, \mathbf{M}, s_i \models \varphi_2 \text{ and } \mathbf{M}, s_j \models \varphi_1 \text{ for } 0 \leq j < i \\
\mathbf{M}, s \models \mathbf{E}[\varphi_1 \, \mathbf{R} \, \varphi_2] & \text{iff} & \text{for some path } \pi = (s_0, s_1, \ldots) \text{ with } s = s_0, \\
& & \text{there exists } i \geq 0 \text{ such that } \mathbf{M}, s_j \models \varphi_2 \text{ for all } 0 \leq j \leq i, \\
& & \text{and } (\mathbf{M}, s_i \models \varphi_1 \text{ or } s_i \in \{s_0, \ldots, s_{i-1}\}) \\
\mathbf{M}, s \models \mathbf{AX}\varphi & \text{iff} & \mathbf{M}, s' \models \varphi \text{ for all state } s' \in S \text{ such that } s' = \mathbf{N}(s) \\
\mathbf{M}, s \models \mathbf{AF}\varphi & \text{iff} & \text{for all paths } \pi = (s_0, s_1, \ldots) \text{ with } s = s_0, \\
& & \text{there exists } i \geq 0 \text{ such that } \mathbf{M}, s_i \models \varphi \\
\mathbf{M}, s \models \mathbf{AG}\varphi & \text{iff} & \text{for all paths } \pi = (s_0, s_1, \ldots) \text{ with } s = s_0, \\
& & \mathbf{M}, s_i \models \varphi \text{ for all } i \geq 0 \\
\mathbf{M}, s \models \mathbf{A}[\varphi_1 \, \mathbf{U} \, \varphi_2] & \text{iff} & \text{for all paths } \pi = (s_0, s_1, \ldots) \text{ with } s = s_0, \\
& & \text{for some } i \geq 0, \mathbf{M}, s_i \models \varphi_2 \text{ and } \mathbf{M}, s_j \models \varphi_1 \text{ for } 0 \leq j < i \\
\mathbf{M}, s \models \mathbf{A}[\varphi_1 \, \mathbf{R} \, \varphi_2] & \text{iff} & \text{for all paths } \pi = (s_0, s_1, \ldots) \text{ with } s = s_0, \\
& & \text{there exists } i \geq 0 \text{ such that } \mathbf{M}, s_j \models \varphi_2 \text{ for all } 0 \leq j \leq i, \\
& & \text{and } (\mathbf{M}, s_i \models \varphi_1 \text{ or } s_i \in \{s_0, \ldots, s_{i-1}\})
\end{array}
$$

B Model Checking CTL with Witnesses and Counterexamples Generation

$\mathbf{PMC}(p, I, \mathbf{N}) = \langle [\langle I, I - [\![p]\!], \_, \_\rangle], [I - [\![p]\!]]\rangle$
$\mathbf{PMC}(\bar{p}, I, \mathbf{N}) = \langle [\langle I, I - [\![\bar{p}]\!], \_, \_\rangle], [I - [\![\bar{p}]\!]]\rangle$

$$
\begin{array}{l}
\mathbf{EXMC}(\varphi, I, \mathbf{N}) = \\
\quad \text{let } O' = \mathbf{WMC}(p, post(\mathbf{N})(I), \mathbf{N}); \\
\qquad \langle \vec{U}', \vec{W}' \rangle = O'; \\
\qquad \vec{W} = [\widetilde{pre}(\mathbf{N})(U'_0.B)] \\
\qquad \text{in } \langle [\langle I, I \cap W_0, O', \_\rangle; \vec{U}'], \vec{W}\rangle \\
\mathbf{EFMC}(\varphi, I, \mathbf{N}, V, V^-) = \\
\quad \text{let } O' = \mathbf{WMC}(\varphi, I, \mathbf{N}); \\
\qquad \langle \vec{U}', \vec{W}' \rangle = O'; \\
\qquad I' = U'_0.B - V \\
\qquad \text{in (if } I' = \emptyset \\
\qquad\quad \text{then} \\
\qquad\quad\; \text{let } \vec{W}^m = \widetilde{pre}^\wedge(\mathbf{N})(V^-) \\
\qquad\quad\;\; \text{in } S = \langle [\langle I, U'_0.B \cap W_m, R\rangle], \vec{W}\rangle \\
\qquad\quad \text{else} \\
\qquad\quad\; \text{let } \langle \vec{U}, \vec{W}^m \rangle = \mathbf{EFMC}(\varphi, post(\mathbf{N})(I'), \mathbf{N}, V \cup I, V^- \cup I', \mathbf{N}) \\
\qquad\quad\;\; \text{in } \langle [\langle I, (I \cap W_m), O', \_\rangle; \vec{U}], \vec{W}\rangle \\
\qquad\quad \text{endif)} \\
\mathbf{EGMC}(\varphi, I, \mathbf{N}, V, V^+) = \\
\quad \text{let } O' = \mathbf{WMC}(\varphi, I, \mathbf{N}); \\
\qquad \langle \vec{U}', \vec{W}' \rangle = O'; \\
\qquad I' = I - (V \cup U'_0.B) \\
\qquad \text{in (if } I' = \emptyset \\
\qquad\quad \text{then} \\
\qquad\quad\; \text{let } \vec{W}^m = pre(\mathbf{N})^\wedge(V^+) \\
\qquad\quad\;\; \text{in } S = \langle [\langle I, I - W_m, O'\rangle], \vec{W}\rangle \\
\qquad\quad \text{else} \\
\qquad\quad\; \text{let } \langle \vec{U}, \vec{W}^m \rangle = \mathbf{EGMC}(\varphi, post(\mathbf{N})(I'), \mathbf{N}, V \cup I, V^+ \cup I', \mathbf{N}) \\
\qquad\quad\;\; \text{in } \langle [\langle I, (I - W_m), O', \_\rangle; \vec{U}], \vec{W}\rangle \\
\qquad\quad \text{endif)}
\end{array}
$$

In the definitions below, we assume that $V_2 \subseteq V_1$.

$$
\begin{aligned}
\widetilde{rpre}(\mathbf{N})^\wedge(V_1, V_2) = \vec{W}^m, &\text{ where} \\
& W_0 = V_1 \\
& W_{i+1} = W_i \cap (V_2 \cup \widetilde{pre}(\mathbf{N})(W_i)) \\
& W_m = \widetilde{pre}(\mathbf{N})(W_m)
\end{aligned}
$$

$$
\begin{aligned}
rpre(\mathbf{N})^\wedge(V_1, V_2) = \vec{W}^m, &\text{ where} \\
& W_0 = V_1
\end{aligned}
$$

$$W_{i+1} = W_i \cap (V_2 \cup pre(\mathbf{N})(W_i))$$
$$W_m = pre(\mathbf{N})(W_m)$$

$\mathbf{EUMC}(\varphi_1, \varphi_2, I, \mathbf{N}, V, V_{\overline{\varphi_1\varphi_2}}, V_{\overline{\varphi_2}}) =$
let $O'' = \mathbf{WMC}(\varphi_2, I, \mathbf{N})$
$\langle \vec{U}'', \vec{W}'' \rangle = O''$
$I'' = U_0''.B$
$O' = \mathbf{WMC}(\varphi_1, I'', \mathbf{N})$
$\langle \vec{U}', \vec{W}' \rangle = O'$
$I' = I'' - (V \cup U_0'.B)$
$V_{\overline{\varphi_2}} = V_{\overline{\varphi_2}} \cup U_0''.B;$
$V_{\overline{\varphi_1\varphi_2}} = V_{\overline{\varphi_1\varphi_2}} \cup U_0'.B;$
in (if $I' = \emptyset$
then
let $\vec{W}^m = \widetilde{rpre}(\mathbf{N})^\wedge(V_{\overline{\varphi_2}}, V_{\overline{\varphi_1\varphi_2}})$
in $S = \langle [\langle I, (I' \cap W_m), O', O'' \rangle], \vec{W} \rangle$
else
let $\langle \vec{U}, \vec{W}^m \rangle = \mathbf{EUMC}(\varphi_1, \varphi_2, post(\mathbf{N})(I'), \mathbf{N}, V \cup I,$
$V_{\overline{\varphi_1\varphi_2}}, V_{\overline{\varphi_2}})$
in $\langle [\langle I, (I' \cap W_m), O', O'' \rangle; \vec{U}], \vec{W} \rangle$
endif)

$\mathbf{ERMC}(\varphi_1, \varphi_2, I, \mathbf{N}, V, V_{\varphi_2}, V_{\varphi_2\varphi_1}) =$
let $O'' = \mathbf{WMC}(\varphi_2, I, \mathbf{N});$
$\langle \vec{U}'', \vec{W}'' \rangle = O'';$
$I'' = I - U_0''.B;$
$O' = \mathbf{WMC}(\varphi_1, I'', \mathbf{N});$
$\langle \vec{U}', \vec{W}' \rangle = O';$
$I' = U_0'.B - V;$
$V_{\varphi_2} = V_{\varphi_2} \cup (I - U_0''.B)$
in (if $I' = \emptyset$
then
let $\vec{W}^m = rpre(\mathbf{N})^\wedge(V_{\varphi_2}, V_{\varphi_2\varphi_1})$
in $S = \langle [\langle I, (I' - W_m), O', O'' \rangle], \vec{W} \rangle$
else
let $\langle \vec{U}, \vec{W}^m \rangle = \mathbf{ERMC}(\varphi_1, \varphi_2, post(\mathbf{N})(I'), \mathbf{N},$
$V \cup I, V_{\varphi_2}, V_{\varphi_2\varphi_1} \cup I')$
in $\langle [\langle I, (I' - W_m), O', O'' \rangle; \vec{U}], \vec{W} \rangle$
endif)

$\mathbf{AXMC}(\varphi, I, \mathbf{N}) =$
let $O' = \mathbf{WMC}(p, post(\mathbf{N})(I), \mathbf{N});$
$\langle \vec{U}', \vec{W}' \rangle = O';$
$\vec{W} = [pre(\mathbf{N})(U_0'.B)]$
in $\langle [\langle I, I \cap W_0, O', \_ \rangle; \vec{U}'], \vec{W} \rangle$

$\mathbf{AFMC}(\varphi, I, \mathbf{N}, V, V^-) =$
let $O' = \mathbf{WMC}(\varphi, I, \mathbf{N})$;
$\langle \vec{U}', \vec{W}' \rangle = O'$;
$I' = U'_0.B - V$
in (if $I' = \emptyset$
then
let $\vec{W}^m = pre(\mathbf{N})^\wedge(V^-)$
in $S = \langle [\langle I, U'_0.B \cap W_m, R \rangle], \vec{W} \rangle$
else
let $\langle \vec{U}, \vec{W}^m \rangle = \mathbf{AFMC}(\varphi, post(\mathbf{N})(I'), \mathbf{N}, V \cup I, V^- \cup I', \mathbf{N})$
in $\langle [\langle I, (I \cap W_m), O', \_ \rangle; \vec{U}], \vec{W} \rangle$
endif)

$\mathbf{AGMC}(\varphi, I, \mathbf{N}, V, V^+) =$
let $O' = \mathbf{WMC}(\varphi, I, \mathbf{N})$;
$\langle \vec{U}', \vec{W}' \rangle = O'$;
$I' = I - (V \cup U'_0.B)$
in (if $I' = \emptyset$
then
let $\vec{W}^m = \widetilde{pre}^\wedge(\mathbf{N})(V^+)$
in $S = \langle [\langle I, I - W_m, O' \rangle], \vec{W} \rangle$
else
let $\langle \vec{U}, \vec{W}^m \rangle = \mathbf{AGMC}(\varphi, post(\mathbf{N})(I'), \mathbf{N}, V \cup I, V^+ \cup I', \mathbf{N})$
in $\langle [\langle I, (I - W_m), O', \_ \rangle; \vec{U}], \vec{W} \rangle$
endif)

$\mathbf{AUMC}(\varphi_1, \varphi_2, I, \mathbf{N}, V, V_{\overline{\varphi_1 \varphi_2}}, V_{\overline{\varphi}_2}) =$
let $O'' = \mathbf{WMC}(\varphi_2, I, \mathbf{N})$;
$\langle \vec{U}'', \vec{W}'' \rangle = O''$;
$I'' = U''_0.B$;
$O' = \mathbf{WMC}(\varphi_1, I'', \mathbf{N})$;
$\langle \vec{U}', \vec{W}' \rangle = O'$;
$I' = I'' - (V \cup U'_0.B)$;
$V_{\overline{\varphi_2}} = V_{\overline{\varphi_2}} \cup U''_0.B$;
$V_{\overline{\varphi_1 \varphi_2}} = V_{\overline{\varphi_1 \varphi_2}} \cup U'_0.B$
in (if $I' = \emptyset$
then
let $\vec{W}^m = rpre(\mathbf{N})^\wedge(V_{\overline{\varphi_2}}, V_{\overline{\varphi_1 \varphi_2}})$
in $\langle [\langle I, (I' \cap W_m), O', O'' \rangle], \vec{W} \rangle$
else
let $\langle \vec{U}, \vec{W}^m \rangle = \mathbf{AUMC}(\varphi_1, \varphi_2, post(\mathbf{N})(I'), \mathbf{N}, V \cup I,$
$V_{\overline{\varphi_1 \varphi_2}}, V_{\overline{\varphi_2}})$
in $\langle [\langle I, (I' \cap W_m), O', O'' \rangle; \vec{U}], \vec{W} \rangle$
endif)

$$
\begin{array}{l}
\mathbf{ARMC}(\varphi_1, \varphi_2, I, \mathbf{N}, V, V_{\varphi_2}, V_{\varphi_2\varphi_1}) = \\
\quad \text{let } O'' = \mathbf{WMC}(\varphi_2, I, \mathbf{N}); \\
\qquad \langle \vec{U}'', \vec{W}'' \rangle = O''; \\
\qquad I'' = I - U_0''.B; \\
\qquad O' = \mathbf{WMC}(\varphi_1, I'', \mathbf{N}); \\
\qquad \langle \vec{U}', \vec{W}' \rangle = O'; \\
\qquad I' = U_0'.B - V; \\
\qquad V_{\varphi_2} = V_{\varphi_2} \cup (I - U_0''.B) \\
\qquad \text{in (if } I' = \emptyset \\
\qquad\quad \text{then} \\
\qquad\qquad \text{let } \vec{W}^m = \widetilde{rpre}(\mathbf{N})\hat{}(V_{\varphi_2}, V_{\varphi_2\varphi_1}) \\
\qquad\qquad\; \text{in } S = \langle [\langle I, (I' - W_m), O', O'' \rangle], \vec{W} \rangle \\
\qquad\quad \text{else} \\
\qquad\qquad \text{let } \langle \vec{U}, \vec{W}^m \rangle = \mathbf{ARMC}(\varphi_1, \varphi_2, post(\mathbf{N})(I'), \mathbf{N}, \\
\qquad\qquad\qquad\qquad\qquad V \cup I, V_{\varphi_2}, V_{\varphi_2\varphi_1} \cup (I'' - I')) \\
\qquad\qquad\; \text{in } \langle [\langle I, (I' - W_m), O', O'' \rangle; \vec{U}], \vec{W} \rangle \\
\qquad\quad \text{endif)}
\end{array}
$$