Approximation Refinement for Interpolation-Based Model Checking*

Vijay D'Silva[1], Mitra Purandare[1], and Daniel Kroening[2]

[1] Computer Systems Institute, ETH Zurich, Switzerland
{firstname.lastname}@inf.ethz.ch
[2] Computing Laboratory, Oxford University, UK
kroening@comlab.ox.ac.uk

Abstract. Model checking using Craig interpolants provides an effective method for computing an over-approximation of the set of reachable states using a SAT solver. This method requires proofs of unsatisfiability from the SAT solver to progress. If an over-approximation leads to a satisfiable formula, the computation restarts using more constraints and the previously computed approximation is not reused. Though the new formula eliminates spurious counterexamples of a certain length, there is no guarantee that the subsequent approximation is better than the one previously computed. We take an abstract, approximation-oriented view of interpolation based model checking. We study counterexample-free approximations, which are neither over- nor under-approximations of the set of reachable states but still contain enough information to conclude if counterexamples exist. Using such approximations, we devise a model checking algorithm for approximation refinement and discuss a preliminary implementation of this technique on some hardware benchmarks.

1 Introduction

Model Checking is an algorithmic technique for establishing that a transition system satisfies certain mathematically specified correctness requirements [1]. Symbolic model checking techniques employ implicit representations of set of states such as Binary Decision Diagrams (BDDs) or propositional logic formulae. Formulating stages in model checking as a Boolean satisfiability (SAT) problem allows model checking tools to harness the capabilities of propositional SAT solvers, thereby greatly enhancing their scalability. However, image computation and fixed point detection, two essential steps in model checking, both involve the expensive operation of quantifier elimination.

A Craig interpolant is a formula, which can be extracted from a resolution proof generated by a SAT-solver in linear time [2,3]. For a suitably constructed Boolean formula, the interpolant provides a conservative approximation of the image, obviating the need for precise image computation [4]. Interpolants also

* This research is supported by SRC contract 2006-TJ-1539 and a DAC graduate fellowship.

provide a means for detecting fixed points detection without quantifier elimination. On the other hand, the approximate image may contain spurious counterexamples, necessitating quality guarantees for the approximation.

The progress and efficiency of an interpolation-based model checking algorithm is contingent on its ability to (a) avoid and eliminate spurious counterexamples, and (b) to rapidly reach a fixed point. If the original algorithm encounters a spurious counterexample, the computation is begun afresh using a more constrained formula. Though techniques have been suggested for reusing interpolants [5], there is a significant loss of information between computations using different constraints. In particular, the approximation is guaranteed to exclude spurious counterexamples shorter than a specified bound. Increasing the bound from k to k' only guarantees that spurious counterexamples shorter than k' are eliminated. The new approximation may contain spurious counterexamples longer than k', which were previously absent, and may omit states that were previously known to be safe. Thus, both the spurious counterexamples and the number of iterations may differ vastly between subsequent runs of the interpolation-based algorithm.

Contribution. The motivation for our work is to devise a method to reuse and refine approximations after each restart. To this end, our contributions are:

1. Modeling interpolation-based model checking with approximate image and pre-image operators that provide *counterexample guarantees*. This abstract view allows us to employ standard tools from fixed point approximation.
2. *Counterexample-free approximations*. These neither over- nor under-approximate the set of reachable states but contain sufficient information to conclude if counterexamples exist. Such approximations do not result in satisfiability and can be reused.
3. A new algorithm for interpolation-based model checking. We combine approximate forward and backward analysis to successively refine counterexample-free approximations until a result is obtained.

Related Work. Model checking using interpolants was first proposed in [4]. Marques-Silva [6,5] identifies optimizations and conditions for interpolant reuse, but no guarantee about the approximation is provided. In fact, Jhala and McMillan [7] report that the interpolants are "*often unnecessarily weak,*" and introduce an interpolant strengthening method. Their observation concurs with our experience that the interpolant is often too weak to admit reusable approximations.

Approximate techniques combining forward and backward reachability are well known in symbolic model checking with BDDs [8,9,10] and abstract interpretation [11,12], and have been analyzed theoretically [13,14]. The work of Cabodi et al. [15] is closest to ours, combining forward and backward analyzes with interpolants. Their focus is eliminating redundancy in the interpolants, but reuse or refinement are not considered. Refinement techniques using interpolation focus on *abstractions* of the transition relation [16] rather than approximations, as we do. Our algorithm for refining counterexample-free approximations is inspired by that of [12] for refining abstract domains. The differences are that our approximation operators are not monotone and our algorithm is based on our fixed point characterization of counterexample-free approximations.

2 Background

We begin with a review of finite state model checking, followed by a description of symbolic model checking using Craig interpolants.

2.1 Finite State Model Checking

A transition system $M = (S, T)$ consists of a finite set of states S, and a transition relation $T \subseteq S \times S$. We fix M as the transition system for the rest of the paper. A path is a sequence of states $s_0 \to \cdots \to s_n$ such that for $0 \leq i < n$, the pair $(s_i, s_{i+1}) \in T$. The image of a set of states $Q \subseteq S$ is the set of successor states with respect to T, given by the operator $\mathit{post}(Q) = \{s' \in S | \exists s \in Q \text{ and } (s, s') \in T\}$. Let $\mathit{post}^0(Q) = Q$ and $\mathit{post}^{i+1}(Q) = \mathit{post}(\mathit{post}^i(Q))$. Given a set $I \subseteq S$ of initial states, the set of states reachable from I and a fixed point characterization thereof are given by the equation:

$$R_I = \bigcup_{i \geq 0} \mathit{post}^i(I) = \mu Q.(I \cup \mathit{post}(Q))$$

The pre-image of a set $Q \subseteq S$ is the set of states with a successor in Q, described by the operator $\mathit{pre}(Q) = \{s \in S | \exists s' \in Q \text{ and } (s, s') \in T\}$. The set of backward reachable states from a set $F \subseteq S$ of *failure states* and a fixed point characterization thereof are given by the equation:

$$B_F = \bigcup_{i \geq 0} \mathit{pre}^i(F) = \mu Q.(F \cup \mathit{pre}(Q)) \ .$$

A set of states P is *inductive* if for any $Q \subseteq P$, it holds that $\mathit{post}(Q) \subseteq P$. The set P is an *inductive invariant* of a system M with initial states I if P is inductive and $I \subseteq P$. Observe that R_I is the smallest inductive invariant of M, and that $\overline{B_F}$, the complement of B_F, is inductive.

Given sets I of initial and F of failure states, with $I \cap F = \emptyset$, let $\langle I, F\rangle$ denote the assertion that the states in F are unreachable from those in I. The *verification problem* requires deciding if $\langle I, F\rangle$ holds for M. We write $M \models \langle I, F\rangle$ if the assertion holds. A *counterexample* to $\langle I, F\rangle$ is a path $s_0 \to \cdots s_n$ with $s_0 \in I$ and $s_n \in F$. A *possible counterexample* is a path with $s_0 \in I$ or $s_n \in F$. A *spurious counterexample* is a path with $s_n \in F$ and $s_0 \notin R_I$ or $s_0 \in I$ and $s_n \notin B_F$. If $R_I \cap B_F = \emptyset$, we can conclude that $M \models \langle I, F\rangle$.

The length of a path is the number of transitions on it. Consider the set of shortest paths between pairs of states in S. The *diameter* of M is the length of the longest path in this set. The *reachability diameter* of $I \subseteq S$, denoted $rd(I)$, is the length of the longest path in this set emanating from a state in I. The *backward diameter* of $F \subseteq S$, denoted $bd(F)$, is the length of the longest path in this set terminating in a state in F.

2.2 Symbolic Model Checking Using Interpolants

In SAT-based symbolic model checking, sets and relations are represented using propositional logic formulae. In the sequel, we use sets and relations, and their propositional encoding by their characteristic Boolean functions interchangeably.

In 1957, Craig showed that for each inconsistent pair of formulae A, B in first order logic, there exists a formula φ – the *Craig interpolant* [17] – such that

- A implies φ,
- φ is inconsistent with B, and
- φ refers only to non-logical symbols common to A and B.

Intuitively, the interpolant φ can be understood as an abstraction of A. Computing precise images and pre-images using SAT is an expensive operation. McMillan proposed using Craig interpolation for effective, over-approximate image computation [4]. Given a depth k, one starts with an *unwinding* of the transition relation as in Bounded Model Checking (BMC):

$$I(x_0) \wedge T(x_0, x_1) \wedge \ldots \wedge T(x_{k-1}, x_k) \wedge (F(x_1) \vee \cdots \vee F(x_k)) \tag{1}$$

A SAT solver is used to determine if the formula above is satisfiable. If so, there exists a counterexample to $\langle I, F \rangle$ of length at most k and the procedure terminates reporting an error. Otherwise, the formula is partitioned into two parts denoted by A and B as below.

$$\begin{aligned} A &\equiv I(x_0) \wedge T(x_0, x_1) \\ B &\equiv T(x_1, x_2) \wedge \ldots \wedge T(x_{k-1}, x_k) \wedge (F(x_1) \vee \cdots \vee F(x_k)) \end{aligned} \tag{2}$$

The formulae A and B are inconsistent, so there exists an interpolant, say φ. The interpolant represents a set that contains *post*(I), i.e., over-approximates the image of I. In addition, no failure state can be reached from φ in up to $k-1$ steps because φ and B are inconsistent. The algorithm proceeds to the next iteration by checking $\langle \varphi, F \rangle$, which may yield a new interpolant φ'. If an interpolant implies the disjunction of all previously computed interpolants, a fixed point is reached, and one may conclude that F is unreachable from I.

However, as φ over-approximates the image, it may represent states that are not reachable, and thus, Eq. 1 with $\varphi(x_0)$ in place of $I(x_0)$ may become satisfiable even though no counterexample exists. In this case, McMillan's technique restarts the approximation with a higher value of k. The previously computed interpolants are abandoned, and the only information retained after a restart is the new value of k.

3 Approximate Analysis Using Counterexample Guarantees

3.1 Approximations in Model Checking

Interpolation-based model checking relies on computing approximations. In general, model checking methods that use approximations face two challenges:

1. **Counterexamples.** Let $s_0 \rightarrow \cdots \rightarrow s_n$ with $s_n \in F$ be a possible counterexample. Deciding if s_0 is reachable is as hard as model checking, hence other approaches are required to deal with counterexamples.
2. **Progress.** If the approximation computed does not lead to a conclusive answer, a better approximation has to be found, either by refining the existing one or repeating the computation using a better approximation operator.

Interpolation-based methods provide a *counterexample guarantee*, i.e., a formal statement about the counterexamples in the approximation. If a possible counterexample is encountered, the fixed point computation is repeated using a stronger guarantee. Though a stronger guarantee eliminates spurious counterexamples of a given length, the approximation computed need not be comparable to the previous one. In particular, the new approximation may contain spurious counterexamples that were previously absent and omit states previously known not to lead to counterexamples of a certain length.

We model the image computation step in interpolation-based model checking using approximate image operators and formalize the notion of counterexample guarantees. The goal of this formalization is to obtain an abstract, approximation-oriented view of interpolation-based model checking. This view allows us to utilize ideas from approximate model checking [8] and abstract interpretation [18] to derive a new algorithm incorporating approximation reuse and refinement.

3.2 Approximation Operators

We briefly recall standard results from fixed point approximation about combining forward and backward analyzes (See [19] for the related background).

An approximate image operator $\hat{post}$ satisfies $post(Q) \subseteq \hat{post}(Q)$ for all Q. An approximate pre-image operator $\hat{pre}$ satisfies $pre(Q) \subseteq \hat{pre}(Q)$ for all Q. The approximate sets of forward- and backward-reachable states are

$$\hat{R}_I = \bigcup_{i \geq 0} \hat{post}^i(I) \qquad\qquad \hat{B}_F = \bigcup_{i \geq 0} \hat{pre}^i(F)\ .$$

It is a standard fixed point approximation result that $R_I \subseteq \hat{R}_I$ and $B_F \subseteq \hat{B}_F$. We say *approximate operator* to refer to an approximate post- or pre-image operator. An operator $F : S \rightarrow S$ is *additive* if $F(Q \cup Q') = F(Q) \cup F(Q')$. F is *monotone* if for $Q \subseteq Q'$, it holds that $F(Q) \subseteq F(Q')$. An additive operator on a lattice is necessarily monotone. The operators *post* and *pre* are additive, hence monotone. We consider operators $\hat{post}$ and $\hat{pre}$ that are *not* additive or monotone. The approximate image obtained depends on a resolution proof generated by a SAT solver, which in turn depends on the SAT solver's heuristics, thus we cannot make conclusions about monotonicity or additivity. This is a significant difference from several approximate operators in the literature. Widening operators [18] may also not be additive or monotone.

Recall that $\langle I, F\rangle$ is the assertion that F is unreachable from I. Our aim is to determine if $\langle I, F\rangle$ holds by combining forward and backward analysis to successively refine approximations of R_I and B_F.

Definition 1. *Consider the assertions $\langle I, F\rangle$ and $\langle I', F'\rangle$. The assertion $\langle I', F'\rangle$* refines *$\langle I, F\rangle$ if $I' \subseteq I$ and $F' \subseteq F$. The assertion $\langle I', F'\rangle$* is sufficient for *$\langle I, F\rangle$ if it holds that if $M \models \langle I', F'\rangle$ then $M \models \langle I, F\rangle$.*

If $\langle I', F'\rangle$ refines $\langle I, F\rangle$, then $\langle I, F\rangle$ is sufficient for $\langle I', F'\rangle$. This is the core idea behind conservative approximation and abstraction techniques. Another approach, which we adopt, is to use conservative methods to refine a verification problem to a sufficient one. Lemma 1 illustrates one such well known refinement.

Lemma 1. *$M \models \langle I, F\rangle$ if and only if $M \models \langle I \cap B_F, F \cap R_I\rangle$.*

If approximate sets are used, $\langle I \cap \hat{B}_F, F \cap \hat{R}_I\rangle$ is sufficient for $\langle I, F\rangle$, but if $M \not\models \langle I \cap \hat{B}_F, F \cap \hat{R}_I\rangle$, the analysis is inconclusive. In this situation, we refine the approximations $\hat{R}_I$ and $\hat{B}_F$, which in turn may lead to a refinement of $\langle I \cap \hat{B}_F, F \cap \hat{R}_I\rangle$ sufficient for $\langle I, F\rangle$. Lemma 2 provides a fixed point characterization of this iterative refinement process (See [14] for a detailed discussion). The fixed point characterization does not affect the precise result computed. In contrast, if approximate operators are used, fixed point iteration may lead to a more precise result than just $\langle I \cap \hat{B}_F, F \cap \hat{R}_I\rangle$.

Lemma 2. *Let R_I and B_F be the forward- and backward-reachable states for the verification problem $\langle I, F\rangle$. Let $\hat{R}_I$ and $\hat{B}_F$ be corresponding approximate sets computed with $\hat{post}$ and $\hat{pre}$.*

1. *Let $G(\langle X, Y\rangle) = \langle I \cap X \cap B_Y, F \cap Y \cap R_X\rangle$ be a mapping between verification problems. Then, $\langle I \cap B_F, F \cap R_I\rangle$ is the greatest fixed point of G.*
2. *Let $\hat{post}$ and $\hat{pre}$ be monotone. Define $\hat{G}(\langle X, Y\rangle) = \langle I \cap X \cap \hat{B}_Y, F \cap Y \cap \hat{R}_X\rangle$. Let $\langle I_G, F_G\rangle$ be the greatest fixed point of G and $\langle I_{\hat{G}}, F_{\hat{G}}\rangle$ be the greatest fixed point of $\hat{G}$. Then, $I_G \subseteq I_{\hat{G}}$ and $F_G \subseteq F_{\hat{G}}$.*

Such a characterization forms the basis of our algorithm. Though the approximate operators we consider are not monotone, we can define an iterative computation to obtain a similar result. The main obstacle to realizing such an iteration is that $\hat{R}_I$ or $\hat{B}_F$ cannot be computed if the approximation introduces possible counterexamples. Interpolants are computed from proofs of unsatisfiability and counterexamples result in a satisfiable formula. Therefore, we need to study counterexamples in the approximation and design methods to avoid them.

3.3 Counterexample Guarantees

Approximate images computed by interpolation do not contain spurious counterexamples shorter than the specified bound. We formalize this notion as a *counterexample guarantee* and study its properties.

Definition 2. *A counterexample guarantee (P, k) is a set of states $P \subseteq S$ and a natural number k. An approximate image operator $\hat{post}$ provides the counterexample guarantee (P, k) if for all sets $Q \subseteq S$, states $s \in \hat{post}(Q)$ and paths $s = s_0 \rightarrow \cdots \rightarrow s_j$ with $j < k$, if $s_j \in P$, then $s \in post(Q)$. A counterexample guarantee for an approximate pre-image operator $\hat{pre}$ is similarly defined.*

We can be certain that a state in the approximate image, leading to a counterexample shorter than k is not introduced by the approximation. Similarly, a state in the approximate pre-image, reachable from a state in P by a path shorter than k is also contained in the precise pre-image. For example, the counterexample guarantee $(F, 0)$ provides no information about the spurious states in an approximate image. If $(F, 1)$ is the guarantee, the approximation does not introduce any states in F, but may introduce states on a path to F. Thus, if the approximate image of Q contains a state in F, we know that a state in Q leads to F.

Let $\textsc{Int}(A, B)$ be a procedure that returns the interpolant for an unsatisfiable pair A and B. An approximate image providing the counterexample guarantee (F, k) can be derived by computing $\textsc{Int}(A, B)$, where A and B are as follows [4]:

$$\begin{aligned} A &\equiv Q(x_0) \wedge T(x_0, x_1) \\ B &\equiv T(x_1, x_2) \wedge \ldots \wedge T(x_{k-1}, x_k) \wedge (F(x_1) \vee \cdots \vee F(x_k)) \end{aligned} \tag{3}$$

Similarly, we obtain an approximate pre-image operator providing the counterexample guarantee (I, k) by computing $\textsc{Int}(A, B)$, where A and B are:

$$\begin{aligned} A &\equiv T(x_{k-1}, x_k) \wedge Q(x_k) \\ B &\equiv (I(x_0) \vee \cdots \vee I(x_{k-1})) \wedge T(x_0, x_1) \wedge \ldots \wedge T(x_{k-2}, x_{k-1}) \end{aligned} \tag{4}$$

We refer to Eq. 3 and 4 as *interpolation formulae*. If the counterexample guarantee an operator provides is insufficient to reach a conclusion about a possible counterexample, we can generate an operator providing a stronger guarantee.

Definition 3. *A counterexample guarantee (P', k') is stronger than (P, k) if $P \subseteq P'$ and $k \leq k'$.*

McMillan's original algorithm increases k if a possible counterexample is discovered [4], and Marques-Silva [6] suggests heuristics for choosing the increment. This processes can be viewed as iterative strengthening of a counterexample guarantee until it is, intuitively speaking, strong enough. Another possibility is to add states to P. Let $\hat{R}_I$ and $\hat{B}_F$ be computed with $\hat{post}$ and $\hat{pre}$. A counterexample guarantee (P, k) is *image-adequate* if $\hat{R}_I$ contains no spurious counterexamples. (P, k) is *pre-image-adequate* if $\hat{B}_F$ contains no spurious counterexamples.

Theorem 1. *Let $\langle I, F \rangle$ be a verification problem. The counterexample guarantee $(F, bd(F) + 1)$ is image-adequate and $(I, rd(I) + 1)$ is pre-image-adequate.*

Adequate guarantees are worst-case requirements. In practice, properties can be proved using approximate operators that provide weaker guarantees [4]. Corollary 1 indicates the bounds for the special cases of inductive sets and Corollary 2 recalls different adequate guarantees.

Corollary 1. *The counterexample guarantee* $(F, 1)$ *is image-adequate if* $\overline{F}$ *is inductive.* $(I, 1)$ *is pre-image-adequate if* I *is inductive.*

Corollary 2. *For* $k > 1$*, if* (F, k) *is image-adequate, then so is* $(F \cup pre(F), k - 1)$*. If* (I, k) *is pre-image-adequate, then so is* $(I \cup post(I), k - 1)$*.*

To see Corollary 2, observe that if I and F are not inductive, then $rd(I) = rd(I \cup post(I)) + 1$ and $bd(F) = bd(F \cup pre(F)) + 1$. Thus, if, when feasible, the counterexample guarantee is strengthened by taking the union with an image or pre-image in addition to increasing k, fewer operators may have to be tried before the adequacy bound is reached.

3.4 Counterexample-Free Approximations

Strengthening a counterexample guarantee eliminates spurious counterexamples shorter than the specified bound, but need not result in a better approximation. We want to design an approximate image operator that also guarantees an improvement in the approximation. One possibility is to constrain subsequent approximations using previous ones.

A sequence of approximations cannot directly be reused after a restart because they may lead to satisfiability. The approximation cannot be used to constrain the next one because it may not include the reachable states. Marques-Silva [5] identifies conditions for interpolant reuse, but there is no guarantee that the new approximation computed is an improvement.

If the states leading to satisfiability can be eliminated, the formula will again be unsatisfiable and the method can progress. In particular, we want a *counterexample-free approximation*, an approximate set of states, which excludes violations but retains enough information to conclude if violations exist [12]. Figure 1 illustrates the relationship precise and approximate counterexample-free forward approximations. We define the set of counterexample-free reachable states, $S_{I,F}$, to contain all states reachable from I by a path that never visits F. Define $C_{I,F}$ to be the states backward-reachable from F by a path that never visits I.

$$S_{I,F} = \mu Q.[(I \cup post(Q)) \cap \overline{F}] \qquad\qquad C_{I,F} = \mu Q.[(F \cup pre(Q)) \cap \overline{I}]$$

The counterexample-free approximations of $S_{I,F}$ and $C_{I,F}$ in terms of the operators $\hat{post}_F(Q) = [\hat{post}(Q) \cap \overline{F}]$ and $\hat{pre}_I(Q) = [\hat{pre}(Q) \cap \overline{I}]$ and are as below.

$$\hat{S}_{I,F} = \bigcup_{i \geq 0} \hat{post}_F^{\,i}(I) \qquad\qquad \hat{C}_{I,F} = \bigcup_{i \geq 0} \hat{pre}_I^{\,i}(F)$$

The approximation computed using an $\hat{post}_P$, for a set of states P, is contained in $\overline{P}$. Therefore, we obtain a new approximation, which is not worse than the previous one as desired. If there is a counterexample, the sets $post(S_{I,F}) \cap F$ and $pre(C_{I,F}) \cap I$ are not empty. If $M \models \langle I, F \rangle$, then $R_I = S_{I,F}$ and $B_F = C_{I,F}$. The sets $\hat{S}_{I,F}$ and $\hat{C}_{I,F}$ are approximations which do not contain violations but may have states leading to a violation.

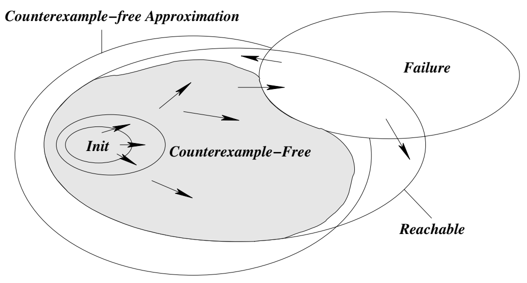

Fig. 1. Counterexample-free forward approximation

Lemma 3. *Let $\langle I, F\rangle$ be an assertion, R_I and B_F be forward- and backward-reachable states, and $S_{I,F}$, $C_{I,F}$, $\hat{S}_{I,F}$ and $\hat{C}_{I,F}$ be as above.*

1. *$R_I \cap F = \emptyset$ if and only if $R_I = S_{I,F}$*
2. *$B_F \cap I = \emptyset$ if and only if $B_F = C_{I,F}$*
3. *$S_{I,F} \subseteq \hat{S}_{I,F}$ and $C_{I,F} \subseteq \hat{C}_{I,F}$*

Assume, for now, that we can compute $\hat{S}_{I,F}$ and $\hat{C}_{I,F}$ using an interpolation-based method. If $post(\hat{S}_{I,F}) \subseteq \hat{S}_{I,F}$, we have an inductive invariant, which contains R_I and can conclude that the property holds. This can be determined using a SAT solver to see if $[\hat{S}_{I,F}(x_0) \wedge T(x_0, x_1) \wedge F(x_1)]$ is unsatisfiable. A similar check applies for $\hat{C}_{I,F}$ and fails if $pre(\hat{C}_{I,F}) \cap I$ is not empty.

Further, if $M \models \langle I, F\rangle$, then $\hat{S}_{I,F}$ and $\hat{C}_{I,F}$ contain R_I and B_F, respectively. If $M \not\models \langle I, F\rangle$, then $\hat{S}_{I,F}$ and $\hat{C}_{I,F}$ must contain a state on a path to and from a violation, respectively. The sets $\hat{S}_{I,F}$ and $\hat{C}_{I,F}$ lose no information about whether $M \models \langle I, F\rangle$ and we can refine the verification problem to $\langle I \cap pre(\hat{C}_{I,F})), F \cap post(\hat{S}_{I,F})\rangle$. Theorem 2 formalizes these intuitive arguments.

Theorem 2. *Let $S_{I,F}$ and $C_{I,F}$ be counterexample-free sets of forward- and backward-reachable states.*

1. *The problem $\langle I \cap pre(C_{I,F}), F \cap post(S_{I,F})\rangle$ is sufficient for $\langle I, F\rangle$.*
2. *Let $G(\langle X, Y\rangle) = \langle I \cap X \cap pre(C_{X,Y}), F \cap Y \cap post(S_{X,Y})\rangle$. Then, $\langle I \cap pre(C_{I,F}), F \cap post(S_{I,F})\rangle$ is the greatest fixed point of G.*

The proof is available in an electronic version of the paper. We can now use approximate operators and define an iterative sequence to compute refinements of $\langle I, F\rangle$, which over-approximate the greatest fixed point of G (the design and soundness of such an iteration follow from fixed point approximation [18]). We still need a method to improve subsequent approximations because $\widehat{post}$ and $\widehat{pre}$ are not monotone. The approximations $\hat{S}_{X,Y}$ and $\hat{C}_{X,Y}$ contain sufficient information to decide $M \models \langle I, F\rangle$ and can thus be used to soundly constrain new approximations.

4 Approximation Refinement with Interpolation

We present an interpolation-based model checking algorithm. Novel aspects of this algorithm are that:

1. It computes counterexample-free approximations, thus avoiding satisfiability until a fixed point is reached.
2. If an approximation leads to an inconclusive answer, a new approximation is computed, which is guaranteed to refine the previous approximation.
3. The approximation is improved using previously computed approximations, ensuring that computed information is not lost.

An additional feature is that we combine forward and backward analysis. Such combinations are by now standard in the model checking literature and have the benefits of both analyzes such as faster convergence to a result.

4.1 Interpolant-Based Approximation Refinement

Our algorithm is shown in Figure 2. It is provided a verification problem $\langle I, F \rangle$ as input. We assume that $I \cap F$ is the empty set. The pair $\langle I_j, F_j \rangle$ is a sufficient refinement of $\langle I, F \rangle$ obtained after j iterations. $\tilde{S}_j$ and $\tilde{C}_j$ are counterexample-free approximations of the reachable and backward reachable states from I_j and F_j, respectively. On Line 3, if a counterexample of length k_j exists, an error is returned. If not, two counterexample-free approximations are computed.

Recall that $\hat{post}_F$ is an approximate operator returning $[\hat{post}(Q) \cap \overline{F}]$ for any Q. The approximate operator $\hat{pre}_I$ similarly returns $[\hat{pre}(Q) \cap \overline{I}]$ for any Q. We can compute counterexample-free approximations using $\hat{post}_{F_j}$ and $\hat{pre}_{I_j}$, but they are not guaranteed to refine $\tilde{S}_{j-1}$ and $\tilde{C}_{j-1}$ because $\hat{post}$ and $\hat{pre}$ are not monotone. Instead, on Line 8, we use the function $[\hat{post}_{F_j}(Q) \cap \tilde{S}_{j-1}]$, to obtain a counterexample-free approximation which does refine $\tilde{S}_{j-1}$. On Line 9, we compute a similar approximation, $\tilde{C}_j$, which refines $\tilde{C}_{j-1}$.

We then check, on Line 10, if either set leads to counterexample. If not, we know that the approximation is an inductive set with no counterexamples, containing the reachable states, and we return No Counterexample. If this check fails, there are states in $\tilde{S}_j$ and $\tilde{C}_j$, which lead to counterexamples. These states may be introduced by the approximation. We progress by refining the verification problem and incrementing the bound k_j.

We need to implement all computations and checks in the algorithm using a SAT solver as shown on Line 3. To determine if $\tilde{S}_j$ and $\tilde{C}_j$ lead to counterexamples on Line 10, we check if either $[\tilde{S}_j(x_0) \wedge T(x_0, x_1) \wedge F_j(x_1)]$ or $[I_j(x_0) \wedge T(x_0, x_1) \wedge \tilde{C}_j(x_1)]$ is unsatisfiable. The main challenge is to compute the sets $\tilde{S}_j$ and $\tilde{C}_j$. We propose two methods, each making a trade-off between efficiency and accuracy. Given sets $Q, F, \tilde{S}$, we need to compute $\hat{post}(Q) \cap \overline{F} \cap \tilde{S}$. We recall the interpolation formula.

$$\begin{aligned} A &\equiv Q(x_0) \wedge T(x_0, x_1) \\ B &\equiv T(x_1, x_2) \wedge \ldots \wedge T(x_{k-1}, x_k) \wedge (F(x_1) \vee \cdots \vee F(x_k)) \end{aligned}$$

```
VERIFY($M, \langle I, F \rangle$)
Input: Transition system $M$, Verification problem $\langle I, F \rangle$

1:  $I_1 = I, F_1 := F, \tilde{S}_0 := \overline{F}, \tilde{C}_0 := \overline{I}, k_1 = 1$
2:  for $j = 1, 2, 3 \ldots$ do
3:      if SAT($I_j \wedge T(x_0, x_1) \cdots T(x_{k_j - 1}, x_{k_j}) \wedge F_j$) then
4:          return COUNTEREXAMPLE
5:      end if
6:      Let $\hat{post}$ provide the guarantee $(I_j, k_j)$.
7:      Let $\hat{pre}$ provide the guarantee $(F_j, k_j)$.
8:      $\tilde{S}_j := \bigcup_{i \geq 0} [\hat{post}_{F_j}(I_j) \cap \tilde{S}_{j-1}]$
9:      $\tilde{C}_j := \bigcup_{i \geq 0} [\hat{pre}_{I_j}(F_j) \cap \tilde{C}_{j-1}]$
10:     if $post(\tilde{S}_j) \cap F_j = \emptyset$ or $pre(\tilde{C}_j) \cap I_j = \emptyset$ then
11:         return NO COUNTEREXAMPLE
12:     else
13:         $I_{j+1} := I_j \cap pre(\tilde{C}_j)$
14:         $F_{j+1} := F_j \cap post(\tilde{S}_j)$
15:         $k_{j+1} := k_j + 1$
16:     end if
17: end for
```

Fig. 2. Interpolation-based Approximation Refinement

A possible counterexample exists if this formula is satisfiable. One possibility is to compute all satisfying assignments to identify the states $P \subseteq Q$, which lead to failures. If $Q(x_0)$ is replaced by $Q(x_0) \wedge \neg P(x_0)$, the formula becomes unsatisfiable and we can proceed. This process is repeated to obtain a sequence of sets $P_1, P_2, \ldots$ of states leading to counterexamples. This method amounts to computing pre-images of F contained in Q. If the approximation introduced by the interpolant is small, the number of satisfying instances is small and this procedure is feasible. We emphasize that this is *not* the same as computing images using a SAT-solver. We compute states in the approximation leading to satisfiability, rather than states in an image. If the set of reachable states is large, but the spurious counterexamples introduced by the approximation are small, this method is still feasible, whereas computing images this way is not.

Our second method uses the counterexample guarantee provided with Q. Any spurious counterexample must be of length at least k. Thus, if we constrain the formula above to be $B \wedge \neg F(x_k)$, the formula again becomes unsatisfiable. We can compute $Q'(x_1) = \text{INT}(A, B \wedge \neg F(x_k))$, a set of states satisfying that every path of length $k - 1$ from s either (a) never visits a failure state, or (b) visits a failure state exactly after $k - 1$ steps. The possibilities are mutually exclusive. If in the next iteration, the formula is unsatisfiable, we can compute the interpolant and proceed as before. If the formula is satisfiable, the counterexample may be of length k or $k-1$. We first add the constraint $\neg F(x_{k-1})$ to the formula B above. If it is unsatisfiable, we can proceed. Otherwise, we also add the constraint $\neg F(x_k)$. The formula must now be unsatisfiable and we can compute the interpolant.

With each constraint that is added, we obtain an approximate image which provides us weaker counterexample guarantees.

Forcing unsatisfiability by adding constraints can be adopted every time the formula becomes satisfiable. In the worst case, we may have to add $\neg F(x_i)$ for all $2 \leq i \leq k$. If the formula is still satisfiable, we can add the constraint $\neg F(x_1)$ to the formula A of the pair. In this case, the interpolant is just $\neg F(x_1)$, which is also the largest counterexample-free approximation we can obtain. A formal statement of correctness follows.

Theorem 3. *If the algorithm returns* Counterexample*, then $M \not\models \langle I, F\rangle$. If the algorithm returns* No Counterexample*, then $M \models \langle I, F\rangle$.*

Proof. The proof of the negative case is straightforward, because an unwinding of the transition relation is used to detect the counterexample.

To prove the positive case, consider the sets $\tilde{S}_j$ and $\tilde{C}_j$. We have established in Theorem 2 that $\langle I \cap pre(C_{I,F}), F \cap post(S_{I,F})\rangle$ is sufficient for $\langle I, F\rangle$. It is enough to show that $\langle I_j, F_j\rangle$ in the algorithm is an over-approximation of this pair. Let $\langle X_j, Y_j\rangle$ be the sequence generated by the fixed point iteration in Theorem 2. The sequence is decreasing, so $S_{X_j,Y_j} \subseteq S_{X_{j+1},Y_{j+1}}$. The pair $\langle X_{j+1}, Y_{j+1}\rangle$ is computed using $post_{Y_j}$ and pre_{X_j}. The corresponding sets computed using $\hat{post}_{Y_j}$ and $\hat{pre}_{X_j}$ must therefore be over-approximations. Further, each set $\tilde{S}_j$ is only constrained using $\tilde{S}_{j-1}$, which is an over-approximation of S_{I_j,F_j}, therefore $\tilde{S}_j$ over-approximates S_{I_j,F_j}. The same applies for $\tilde{C}_j$. The pair $\langle I_j, F_j\rangle$ computed from this pair of sets is sufficient for $\langle I \cap pre(C_{I,F}), F \cap post(S_{I,F})\rangle$. Correctness follows. □

To complete the formal analysis, we have a statement about termination.

Theorem 4. *The algorithm always terminates with a positive or negative result.*

Proof. If $M \not\models \langle I, F\rangle$, then, because the sequence $\langle I_j, F_j\rangle$ computed by the algorithm is sufficient for $\langle I, F\rangle$, the pair I_j and F_j never becomes empty. In each iteration, the bound k_j is increased until it reaches the length of the counterexample, when the failure is reported.

If $M \models \langle I, F\rangle$, the bound k_j is eventually increased to either $rd(M) + 1$ or $bd(M) + 1$. Recall from Theorem 1 that such a counterexample guarantee is adequate, meaning that it does not introduce any spurious counterexamples. Thus, in the next iteration, either I_{j+1} or F_{j+1} computed by the algorithm is the empty set and the algorithm terminates. □

Optimizations. The algorithm as presented admits several optimizations. These include standard methods such as frontier set simplification and logic minimization. A standard check to make after Line 3 is to see if the new bound is k-inductive [20]. Recent developments which may enhance our method are dynamic abstraction and redundancy elimination for interpolants [15] and interpolant strengthening [7]. Our current implementation is naïve and is based on the blocking clauses algorithm for AllSAT. Minimal inductive sub-clauses extracted from counterexamples [21] may reduce the effort required to obtain an unsatisfiable formula.

4.2 Experience

We have implemented a preliminary version of this algorithm and experimented with some hardware benchmarks. We have proposed obtaining an unsatisfiable formula by either constraining the satisfiable formula using either blocking clauses or the set of failure states. The second method being symbolic showed more promise, but we are unable to present an evaluation due to technical problems (which we are rectifying). Thus, we can only present results for our algorithm where the approximations are computed using AllSAT.

Given the preliminary nature of our implementation, our conclusions are, for the moment, primarily qualitative.[1] If the interpolation formula never becomes satisfiable, our method essentially reduces to doing standard interpolation-based model checking. The hardware benchmarks we considered can be divided into three categories:

1. *Small.* Such circuits either have a small depth or result in approximations which preserve unsatisfiability. Examples include the ITC '99 benchmarks [22]. The basic interpolation algorithm is able to prove properties of these circuits using a small unwinding, so our method was never invoked.
2. *Medium.* These circuits compute arithmetic and Boolean functions. The over-approximation introduced does lead to satisfiability and our technique does help to reach a fixed point.
3. *Large.* These include processor benchmarks and satisfiable instances occur often. The enumerative procedure usually exhausts the memory or time limits set. Our experience with such circuits is that the approximation introduced by interpolation is extremely coarse, yielding no useful information.

It appears that our method is superfluous for small circuits, but may yield useful invariants for intermediate circuits, though it is unclear if there will be a performance improvement. With large circuits, the interpolants appear to be too coarse and computing a fixed point provides no benefits. It is an open question if methods for interpolant strengthening will help [7].

5 Conclusion

To summarize, we initiated a study of interpolation-based model checking using fixed point approximation. We introduced counterexample-free approximations to reduce the number of restarts and to enable the reuse of approximations during model checking. Our verification algorithm progresses by iteratively strengthening counterexample guarantees and refining approximations.

[1] The implementation is available at: `http://www.verify.ethz.ch/ebmc/`

The new method yields useful invariants and reduces the restarts required when model checking medium sized circuits but is unnecessary for small circuits. On large circuits, it is inconclusive, as it appears that the interpolants are extremely coarse, so computing a fixed point does not yield much information. This highlights the need for computing tighter interpolants, and other techniques to force unsatisfiability, the focus of our current research.

References

1. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching-time temporal logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, pp. 52–71. Springer, Heidelberg (1982)
2. Krajícek, J.: Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. Journal of Symbolic Logic 62, 457–486 (1997)
3. Pudlák, P.: Lower bounds for resolution and cutting plane proofs and monotone computations. Journal of Symbolic Logic 62, 981–998 (1997)
4. McMillan, K.L.: Interpolation and SAT-Based Model Checking. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 1–13. Springer, Heidelberg (2003)
5. Marques-Silva, J.: Interpolant learning and reuse in SAT-based model checking. Electron. Notes Theor. Comput. Sci. 174, 31–43 (2007)
6. Marques-Silva, J.: Improvements to the Implementation of Interpolant-Based Model Checking. In: Borrione, D., Paul, W. (eds.) CHARME 2005. LNCS, vol. 3725, pp. 367–370. Springer, Heidelberg (2005)
7. Jhala, R., McMillan, K.L.: Interpolant-based transition relation approximation (2007)
8. Govindaraju, S.G., Dill, D.L.: Verification by approximate forward and backward reachability. In: International Conference on Computer-Aided Design (ICCAD), pp. 366–370. ACM Press, New York (1998)
9. Cabodi, G., Nocco, S., Quer, S.: Mixing forward and backward traversals in guided-prioritized BDD-based verification. In: Computer Aided Verification (CAV), Springer, pp. 471–484. Springer, Heidelberg (2002)
10. Stangier, C., Sidle, T.: Invariant Checking Combining Forward and Backward Traversal. In: Hu, A.J., Martin, A.K. (eds.) FMCAD 2004. LNCS, vol. 3312, pp. 414–429. Springer, Heidelberg (2004)
11. Cousot, P., Cousot, R.: Refining model checking by abstract interpretation. Automated Software Engineering 6, 69–95 (1999)
12. Cousot, P., Ganty, P., Raskin, J.F.: Fixpoint-guided abstraction refinements. In: Symposium on Static Analysis (SAS), pp. 333–348. Springer, Heidelberg (2007)
13. Henzinger, T.A., Kupferman, O., Qadeer, S.: From *Pre*-historic to *Post*-modern symbolic model checking. Formal Methods in System Design 23, 303–327 (2003)
14. Massé, D.: Combining Forward and Backward Analyses of Temporal Properties. In: Danvy, O., Filinski, A. (eds.) PADO 2001. LNCS, vol. 2053, pp. 155–172. Springer, Heidelberg (2001)
15. Cabodi, G., et al.: Stepping forward with interpolants in unbounded model checking. In: International conference on Computer-aided design (ICCAD), pp. 772–778. ACM Press, New York (2006)

16. Somenzi, F., Li, B.: Efficient Abstraction Refinement in Interpolation-Based Unbounded Model Checking. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006 and ETAPS 2006. LNCS, vol. 3920, pp. 227–241. Springer, Heidelberg (2006)
17. Craig, W.: Linear reasoning. A new form of the Herbrand-Gentzen theorem 22, 250–268 (1957)
18. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Principles of Programming Languages (POPL), pp. 238–252. ACM Press, New York (1977)
19. Cousot, P.: Semantic foundations of program analysis. In: Program Flow Analysis: Theory and Applications, pp. 303–342. Prentice-Hall, Englewood Cliffs (1981)
20. Sheeran, M., Singh, S., Stalmarck, G.: Checking safety properties using induction and a SAT-solver. In: Formal Methods in Computer-Aided Design (FMCAD), pp. 108–125. Springer, Heidelberg (2000)
21. Bradley, A., Manna, Z.: Checking safety by inductive generalization of counterexamples to induction. In: Formal Methods in Computer-Aided Design (FMCAD), IEEE, Los Alamitos (to appear, 2007)
22. Corno, F., Reorda, M.S., Squillero, G.: RT-level ITC'99 benchmarks and first ATPG results. IEEE Design and Test 17, 44–53 (2000)