

# Instantiation Reduction in Iterative Parameterised Three-Valued Model Checking

Nils Timm<sup>(✉)</sup> and Stefan Gruner

Department of Computer Science, University of Pretoria, Pretoria, South Africa  
{ntimm,sgruner}@cs.up.ac.za

**Abstract.** We introduce an enhanced approach to parameterised three-valued model checking (PMC) based on *iterative* parameterisation. The model is parameterised until it is precise enough for a definite verification result. Results from past iterations are *reused* to reduce the number of parameter instances in future iterations. Our approach is based on a SAT encoding. In the initial iteration we construct an over-approximation of all possible instances in later iterations. For this over-approximation we compute the set of all satisfying interpretations. All subsequent iterations are then accomplished by *validating* whether for each instance one of the precomputed interpretations is satisfying as well, which is less costly than solving each SAT instance from scratch. Our iterative parameterisation approach leads to a substantial speed-up of PMC.

## 1 Introduction

*Three-valued abstraction* [11] is an established technique in software verification. It proceeds by generating a state space model of the system to be analysed over the values *true*, *false* and *unknown*, where the latter value represents the loss of information due to abstraction. The evaluation of temporal logic properties on such models is *three-valued model checking* (3MC) [3]. In case of an *unknown* result, the abstraction is typically refined by adding more predicates over the systems variables until a level of abstraction is reached where the property can be definitely proved or refuted. With each additional predicate the state space grows *exponentially*. Thus, such a classical refinement does not guarantee that eventually a model can be constructed that is both precise enough for a definite outcome and small enough to be manageable with the available resources.

At SBMF '14 we introduced *parameterised three-valued model checking* (PMC) [12] as an extension of 3MC. Predicates and transitions in PMC models can be associated with the values *true*, *false*, *unknown*, or with expressions over *Boolean parameters*. Parameterisation is an alternative way to state that the value of certain predicates or transitions is not known and that the checked property has to yield the same result under each possible parameter instantiation. PMC is thus conducted via evaluating a property on *all* instantiations, whose number is *exponential* in the number of parameters. Parameterisation particularly allows to establish *logical connections* between unknowns in the model: It enables to represent facts like ‘a certain pair of transitions has unknown

but *complementary* truth values'. Such facts can be automatically derived from the system to be verified, and covering these facts in a model can significantly enhance the precision in verification. In many cases it is possible to replace a large number of classical refinement steps by a single parameterisation step in order to obtain the necessary precision for a definite result. In [13] we introduced a propositional logic encoding of PMC problems that allows to perform PMC via SAT solving. The solving performance of our SAT-based approach profits from conflict clause sharing between the SAT instances. However, the number of instances associated with an encoding is still *exponential* in the number of parameters.

Here we introduce an enhanced approach to PMC that substantially overcomes the exponential overhead of parameterisation that existed in [12, 13]. Parameterisation is typically applied *iteratively* until the model is precise enough for a definite result. We show that in an iterative approach partial results from past iterations can be used to narrow the number of relevant instantiations in future iterations. Technically, a parameterisation step splits each instantiation of the current iteration into two instantiations of the subsequent iteration. This allows us establish a successor relation between the instantiations of different parameter iterations. We prove that if a property holds for a certain instantiation of the current iteration then it also holds for all its successor instantiations in future iterations. Hence, in each iteration only those instantiations have to be considered for which no predecessor exists that fulfils the property. Such an *instantiation reduction* works for *disproving* temporal logic properties as well.

Moreover, our iterative approach allows to significantly reduce the computational costs for checking *individual* instantiations. The initially non-parameterised model can be straightforwardly transferred into an over-approximation (wrt. the validity of temporal logic properties) of all parameter instantiations in later iterations. We exploit this fact as follows: In the first iteration we compute the set of all paths witnessing the property of interest in the over-approximation. In each subsequent iteration, instead of conducting model checking from scratch for each instantiation, we only have to track the paths from the over-approximating set in the instantiations in order to determine whether any of them witnesses the property here too. The savings of this approach become evident in our SAT-based PMC scenario: The computation of all witness paths in the over-approximation reduces to the computation of all satisfying interpretations of the initial encoding, which can be efficiently performed via AllSAT techniques [8, 14, 15]. Now all subsequent iterations can be accomplished by just *validating* whether for each SAT instantiation one of the precomputed interpretations is satisfying as well.

Our new concepts *instantiation reduction* and *interpretation validation* thus enhance PMC on the one hand by reducing the *number* of instantiations that have to be processed per iteration, and on the other hand by reducing the *effort* for processing individual instantiations. This enables us to profit from the extra precision of parameterisation without suffering from the previous drawbacks. We implemented a bounded model checker for PMC problems that employs iterative

parameterisation and applies our optimisations. Preliminary experiments show that our new approach leads to a substantial speed-up of PMC.

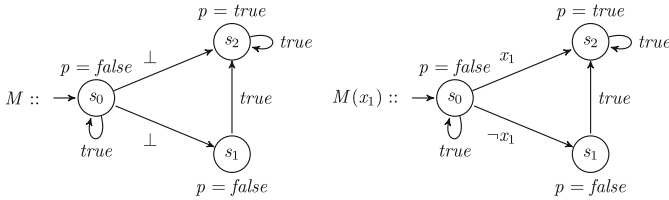
## 2 Background: Parameterised 3-Valued Model Checking

We briefly review *pure* three-valued model checking (3MC) [3] and *parameterised* three-valued model checking (PMC) [12]. A more extensive introduction can be found in [12, 13]. The key feature of 3MC is a third truth value  $\perp$  (i.e. *unknown*) for transitions and labels, which is used to model uncertainty. PMC additionally allows *Boolean parameter expressions*, which enable to establish logical connections between unknown parts. As models we use Kripke structures:

**Definition 1 (Parameterised Three-Valued Kripke Structure).** A parameterised three-valued Kripke structure *over a set of atomic predicates AP and an ordered set of Boolean parameters  $X_n = (x_1, \dots, x_n)$*  is a parameterised tuple  $M(X_n) = (S, s_0, R(X_n), L(X_n))$  where

- $S$  is a finite set of states and  $s_0 \in S$  is the initial state,
- $R(X_n) : S \times S \rightarrow \{\text{true}, \perp, \text{false}\} \cup BE(X_n)$  is a transition function with  $\forall s \in S : \exists s' \in S : R(X_n)(s, s') \in \{\text{true}, \perp\} \cup BE(X_n)$  where  $BE(X_n)$  denotes the set of Boolean expressions over  $X_n$ ,
- $L(X_n) : S \times AP \rightarrow \{\text{true}, \perp, \text{false}\} \cup BE(X_n)$  a label function that associates a truth value or parameter expression with each predicate in each state.

An *instantiation* of a parameterised three-valued Kripke structure  $M(X_n)$  is a *pure* three-valued Kripke structure  $M(B_n)$  where  $B_n \in \{\text{true}, \text{false}\}^n$ . A structure is also *pure* if  $X_n = \emptyset$ . An example for a pure three-valued Kripke structure  $M$  and a parameterised Kripke structure  $M(x_1)$  is depicted below.



A parameterised three-valued Kripke structure can be obtained by parameterising a pure three-valued Kripke structure according to the rules from [12]. This enhances the precision but also leads to an exponential increase of the number of parameter instantiations that have to be considered. For instance, if the transitions  $(s_0, s_1)$  and  $(s_0, s_2)$  of our example structure  $M$  correspond to a complementary branch (e.g. *if-then-else*) in the modelled system, then  $M(x_1)$  is a sound parameterisation of  $M$ .

In the following we first introduce 3MC and then generalise it to PMC. A path  $\pi$  of a structure  $M$  is a sequence  $s_0 s_1 s_2 \dots$  with  $R(s_i, s_{i+1}) \in \{\text{true}, \perp\}$ .  $\pi(i)$  denotes the  $i$ -th state of  $\pi$ , whereas  $\pi^i$  denotes the  $i$ -th suffix  $\pi(i) \pi(i+1) \dots$  of  $\pi$ . By  $\Pi_M$  we denote the set of all paths of  $M$  starting in the initial state. We use the temporal logic LTL for specifying properties with regard to paths.

**Definition 2 (Syntax of LTL).** Let  $AP$  be a set of atomic predicates and  $p \in AP$ . The syntax of LTL formulae  $\psi$  is given by

$$\psi ::= p \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi \mid \mathbf{X}\psi \mid \mathbf{G}\psi \mid \mathbf{F}\psi \mid \psi \mathbf{U} \psi.$$

In 3MC we operate under the three-valued Kleene logic  $\mathcal{L}_3$  [7]. Based on  $\mathcal{L}_3$ , LTL formulae can be evaluated on paths Kripke structures:

**Definition 3 (Three-Valued Evaluation of LTL).** Let  $M = (S, s_0, R, L)$  over  $AP$  be a pure three-valued Kripke structure. Then the evaluation of an LTL formula  $\psi$  on a path  $\pi \in \Pi_M$ , written  $[\pi \models \psi]$ , is defined as follows

$$\begin{aligned} [\pi \models p] &:= L(\pi(0), p) \\ [\pi \models \neg\psi] &:= \neg[\pi \models \psi] \\ [\pi \models \psi \vee \psi'] &:= [\pi \models \psi] \vee [\pi \models \psi'] \\ [\pi \models \mathbf{G}\psi] &:= \bigwedge_{i \in \mathbb{N}} (R(\pi(i), \pi(i+1)) \wedge [\pi^i \models \psi]) \\ [\pi \models \mathbf{F}\psi] &:= \bigvee_{i \in \mathbb{N}} ([\pi^i \models \psi] \wedge \bigwedge_{j=0}^{i-1} R(\pi(j), \pi(j+1))) \end{aligned}$$

Complete LTL definitions can be found in [12]. 3MC [3] is now defined as follows:

**Definition 4 (Three-Valued LTL Model Checking).** Let  $M = (S, s_0, R, L)$  over  $AP$  be a three-valued Kripke structure. Moreover, let  $\psi$  be an LTL formula over  $AP$ . The universal value of  $\psi$  in  $M$  is  $[M \models_U \psi] := \bigwedge_{\pi \in \Pi_M} [\pi \models \psi]$ . The existential value of  $\psi$  in  $M$  is  $[M \models_E \psi] := \bigvee_{\pi \in \Pi_M} [\pi \models \psi]$ .

Universal model checking can always be transferred into existential model checking based on the equation  $[M \models_U \psi] = \neg[M \models_E \neg\psi]$ . In 3MC there exist three possible outcomes: *true*, *false* and  $\perp$ . For our example structure  $M$  we get  $[M \models_U \mathbf{G}\neg p] = \neg[M \models_E \mathbf{F}p] = \perp$ . Hence,  $M$  is not precise enough for a definite result. From now on, we only consider the existential case, since it is the basis for SAT-based bounded model checking [2] which we later apply.

3MC can always be reduced to two instances of two-valued model checking (2MC) if the LTL formula is restricted to  $\text{LTL}^+$ , i.e. negation-free LTL.  $\text{LTL}^+$  formulae are evaluated on *complement-closed* structures. In these structures each  $p \in AP$  has a complementary  $\bar{p} \in AP$  such that  $L(s, p) = \neg L(s, \bar{p})$ . Every structure can be extended to a complement-closed one, without increasing the number of states. For the evaluation on complement-closed structures, each LTL formula can be transferred into an equivalent  $\text{LTL}^+$  formula. Thus, the restriction to  $\text{LTL}^+$  does not limit the expressiveness. The reduction of 3MC to 2MC is based on *completions* of complement-closed structures.

**Definition 5 (Completion).** Let  $M = (S, s_0, R, L)$  over  $AP$  be a three-valued Kripke structure. Then  $M^p = (S, s_0, R^p, L^p)$  is the pessimistic completion and  $M^o = (S, s_0, R^o, L^o)$  is the optimistic completion with  $\forall s, s' \in S$  and  $\forall p \in AP$ :

$$L^p(s, p) := \begin{cases} \text{false} & \text{if } L(s, p) = \perp \\ L(s, p) & \text{else} \end{cases} \quad R^p(s, s') := \begin{cases} \text{false} & \text{if } R(s, s') = \perp \\ R(s, s') & \text{else} \end{cases}$$

$$L^o(s, p) := \begin{cases} \text{true} & \text{if } L(s, p) = \perp \\ L(s, p) & \text{else} \end{cases} \quad R^o(s, s') := \begin{cases} \text{true} & \text{if } R(s, s') = \perp \\ R(s, s') & \text{else} \end{cases}$$

The following lemma from [3] establishes the reduction from 3MC to 2MC:

**Lemma 1.** *Let  $M = (S, s_0, R, L)$  be a complement-closed three-valued Kripke structure and  $\psi$  an  $LTL^+$  formula. Then the following holds:*

$$[M \models_E \psi] = \begin{cases} true & \text{iff } [M^p \models_E \psi] = true \\ false & \text{iff } [M^o \models_E \psi] = false \\ \perp & \text{else} \end{cases}$$

Hence, if a formula holds for the pessimistic completion then it also holds for the three-valued Kripke structure. The same applies to *false* results obtained for the optimistic completion. All definitions wrt. pure 3MC can be straightforwardly generalised to PMC [12], since PMC reduces to multiple instances of 3MC.

**Definition 6 (Parameterised Three-Valued  $LTL^+$  Model Checking).**

*Let  $M(X_n) = (S, s_0, R(X_n), L(X_n))$  be a parameterised three-valued Kripke structure over  $AP$  and  $X = (x_1, \dots, x_n)$ . Moreover, let  $\psi$  be an  $LTL^+$  formula over  $AP$ . The existential value of  $\psi$  in  $M(X_n)$ , written  $[M(X_n) \models_E \psi]$ , is defined as*

$$[M(X_n) \models_E \psi] := \begin{cases} true & \text{if } \forall B_n \in \{t, f\}^n ([M(B_n)^p \models_E \psi] = true) \\ false & \text{if } \forall B_n \in \{t, f\}^n ([M(B_n)^o \models_E \psi] = false) \\ \perp & \text{else} \end{cases}$$

We call  $M(B_n)^p$  a *pessimistic instantiation* where the parameters are instantiated wrt.  $B_n$  and the  $\perp$ 's are set to *false*. In an *optimistic instantiation*  $M(B_n)^o$  the  $\perp$ 's are set to *true* instead. If checking a property yields *true* for all pessimistic instantiations, this result is transferred to the parameterised structure. The same holds for *false* results for all optimistic instantiations. In all other cases PMC returns  $\perp$ . For our example  $M(x_1)$ , we get  $[M(x_1) \models_E \mathbf{F}p] = true$  since  $\mathbf{F}p$  holds existentially for both  $M(true)^p$  and  $M(false)^p$ . In contrast to the pure three-valued  $M$ , the parameterised  $M(x_1)$  captures the fact that the transition values of  $(s_0, s_1)$  and  $(s_0, s_2)$  are unknown but also *complementary*, which gives us the necessary precision for a definite result. The gain of precision comes at the cost of an increase of the number of instantiations *exponential* in the number of parameters. Each instantiation corresponds to a distinct model checking problem that has to be solved. Subsequently, we will show that in *iterative* parameterisation, both the *effort for solving each instance* and the *number of instances to be considered* can be significantly reduced.

### 3 PMC via Path Tracking

Rule-based parameterisation [12] is an alternative to classical refinement: Instead of adding predicates to an abstract model, the value of selected transitions and predicates is changed from  $\perp$  to logical expressions over a parameter set  $X_n$ . Such a parameterisation is typically applied iteratively until a definite result in model checking can be obtained, or until classical refinement is inevitable for

a definite outcome. In contrast to classical refinement, parameterisation does not affect the state space. If  $\pi$  is a path of a three-valued Kripke structure  $M$ , then  $\pi$  is also a path or at least corresponds to a sequence of states in any parameterisation  $M(X_n)$  and its possible instantiations. For instance, the path  $\pi = s_0 s_2 s_2 \dots$  from the Kripke structure  $M$  in our running example from Sect. 2 also exists in the parameterised Kripke structure  $M(x_1)$ . Moreover, this  $\pi$  is a path in the instantiation  $M(true)$  and it corresponds to a sequence of states in  $M(false)$ . This allows us consider the same  $\pi$  in the context of different Kripke structures. In terms of LTL properties this means that we can evaluate a formula  $\psi$  on a particular path resp. sequence  $\pi$  in a Kripke structure  $M$  or any parameterisation  $M(X_n)$  and its instantiations  $M(B_n)$ . In order to make clear *which* Kripke structure is currently in use, we refer to the evaluation of  $\psi$  on  $\pi$  in  $M$  by  $[\pi \models \psi]_M$ . The semantics of this evaluation follows from Definition 3.

Now we show that in case of an *unknown* result for a 3MC problem  $[M \models_E \psi]$ , paths discovered in  $M$  can be exploited for solving any corresponding parameterised problem  $[M(X_n) \models_E \psi]$  more efficiently. According to Lemma 1, the outcome of 3MC is *unknown* if the following holds:  $[M^p \models_E \psi] = false$  and  $[M^o \models_E \psi] = true$ . Hence, there exist paths in the optimistic completion  $M^o$  that witness the property  $\psi$ , but no such path exists in the pessimistic completion  $M^p$ . The optimistic completion is an over-approximation of  $M$  in terms of the existential validity of  $LTL^+$  properties. This over-approximation relation also holds between  $M^o$  and any instantiation  $M(B_n)^y$ ,  $y \in \{o, p\}$ , of a parameterisation  $M(X_n)$  of  $M$ . With regard to paths we get the following lemma:

**Lemma 2.** *Let  $M(X_n)$  be an arbitrary parameterisation of a three-valued Kripke structure  $M$  over  $AP$ , and let  $\psi$  be an  $LTL^+$  formula over  $AP$ . Then for any  $B_n \in \{true, false\}^n$  and any  $y \in \{o, p\}$  the following holds:*

$$\forall \pi \in \Pi_{M(B_n)^y} [\pi \models \psi]_{M(B_n)^y} = true \Rightarrow [\pi \models \psi]_{M^o} = true$$

*Proof.* See <http://www.cs.up.ac.za/cs/ntimm/proofLemma2.pdf>

Hence, each path  $\pi$  that witnesses a property  $\psi$  in any instantiation of a parameterisation of  $M$  is also a witness for  $\psi$  in the optimistic completion  $M^o$ . The set  $\Pi_{M^o}^\psi = \{\pi \in \Pi_{M^o} \mid [\pi \models \psi]_{M^o} = true\}$  is thus a superset of the set of all paths  $\pi$  with  $[\pi \models \psi]_{M(B_n)^y} = true$  for any  $M(B_n)^y$ ,  $y \in \{o, p\}$ . This fact can be exploited to reduce the effort for performing PMC with iterative parameterisation. Solving a parameterised model checking problem  $[M(X_n) \models_E \psi]$  requires to determine whether for each instantiation  $M(B_n)$  there exists a path  $\pi$  with  $[\pi \models \psi]_{M(B_n)^p} = true$ , or whether for all paths  $[\pi \models \psi]_{M(B_n)^o} = false$  holds. In general, this involves the exhaustive exploration of the state space of each instantiation. Provided that we already have computed the set  $\Pi_{M^o}^\psi$  we now can solve  $[M(X_n) \models_E \psi]$  (and any other parameterisation of the original problem) by just iterating over the paths in  $\Pi_{M^o}^\psi$  due to of the aforementioned superset relation. Lemma 2 together with Definition 6 gives us the following theorem:

**Theorem 1.** *Let  $M(X_n)$  be the parameterisation of a three-valued Kripke structure  $M$  over  $AP$ , and let  $\psi$  be an  $LTL^+$  formula over  $AP$ . Furthermore, let  $\Pi_{M^o}^\psi = \{\pi \in \Pi_{M^o} \mid [\pi \models \psi]_{M^o} = \text{true}\}$ . Then the following holds:*

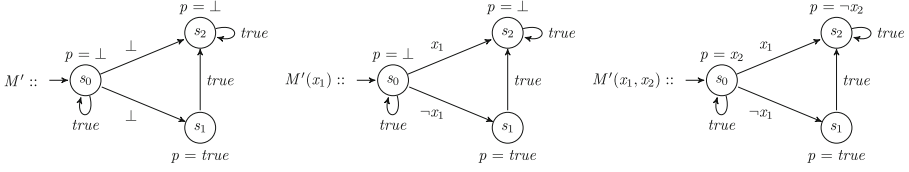
$$[M(X_n) \models_E \psi] = \begin{cases} \text{true} & \text{iff } \forall B_n \in \{t, f\}^n \exists \pi \in \Pi_{M^o}^\psi [\pi \models \psi]_{M(B_n)^p} = \text{true} \\ \text{false} & \text{iff } \forall B_n \in \{t, f\}^n \forall \pi \in \Pi_{M^o}^\psi [\pi \models \psi]_{M(B_n)^o} = \text{false} \\ \perp & \text{else} \end{cases}$$

Thus, instead of conducting model checking from scratch for each possible instantiation of  $M(X_n)$ , we only need to track the paths from  $\Pi_{M^o}^\psi$  on each instantiation. For our running example we have already seen that model checking the initially non-parameterised problem  $[M \models_E \mathbf{F}p]$  yields *unknown*. We now compute the set of witness paths for  $\mathbf{F}p$  in the optimistic completion  $M^o$ , which is  $\Pi_{M^o}^{\mathbf{F}p} = \{\pi_1 = (s_0 s_1 s_2 \dots), \pi_2 = (s_0 s_2 \dots)\}$ . Due to Lemma 2 we have that  $\Pi_{M^o}^{\mathbf{F}p}$  is a superset of all paths  $\pi$  with  $[\pi \models \mathbf{F}p]$  for each (optimistic or pessimistic) instantiation of any parameterisation of  $M$ . The parameterised model  $M(x_1)$  from our running example has the two possible pessimistic instantiations  $M(\text{true})^p$  and  $M(\text{false})^p$ . By tracking the paths from  $\Pi_{M^o}^{\mathbf{F}p}$  we can show that in both instantiations there exists a path that witnesses  $\mathbf{F}p$ :  $[\pi_1 \models \mathbf{F}p]_{M(\text{false})^p} = \text{true}$  and  $[\pi_2 \models \mathbf{F}p]_{M(\text{true})^p} = \text{true}$ . Now Theorem 1 allows us to conclude that  $[M(x_1) \models_E \mathbf{F}p] = \text{true}$ . We have solved a *parameterised* three-valued model checking problem by just tracking witness paths previously discovered in the corresponding *pure* three-valued model.

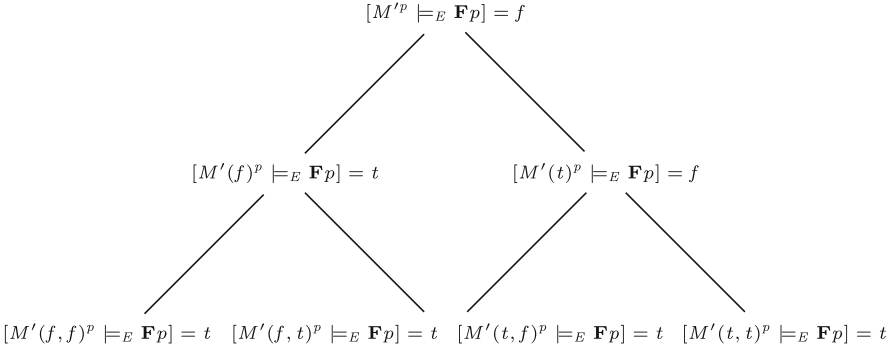
*Iterative* parameterisation of an initially non-parameterised problem thus allows to exploit information obtained in past iterations in order to reduce the search space for witness paths in future iterations. The computational savings of conducting PMC via *path tracking* will become evident when PMC is reduced to propositional logic satisfiability, which we discuss from Sect. 5 on.

## 4 Instantiation Reduction in Iterative Parameterisation

The basic concept of PMC is to verify *each* possible instantiation of a parameterised model and to check whether the results are consistent, i.e. *true* for all pessimistic instances or *false* for all optimistic ones. Inconsistency, i.e. some *true* and some *false* results, may be ruled out via further parameterisation, which however leads to an exponential increase in the number of instantiations. Here we show that in *iterative* parameterisation not necessarily all of these instantiations have to be considered. Partial results from past iterations can be used to reduce the number of relevant instantiations in the current iteration. We illustrate such a reduction based on another example: the three-valued Kripke structure  $M'$  and its parameterisations  $M'(x_1)$  and  $M'(x_1, x_2)$  depicted below.



Checking the property  $\mathbf{F}p$  yields the following results:  $[M' \models_E \mathbf{F}p] = \perp$ ,  $[M'(x_1) \models_E \mathbf{F}p] = \perp$  and  $[M'(x_1, x_2) \models_E \mathbf{F}p] = \text{true}$ . Hence, the iteration that introduced the second parameter<sup>1</sup>  $x_2$  has brought the necessary precision for a definite result in verification. The following tree characterises the iterations and the pessimistic instantiations<sup>2</sup> that have to be considered in order to prove that  $\mathbf{F}p$  holds for our example problem with iterative parameterisation:



In the third iteration we get consistently *true* results for all possible instantiations of  $M'(x_1, x_2)^p$ , which allows us to conclude that  $\mathbf{F}p$  holds. However, checking *all* instances is actually not necessary under iterative parameterisation. Note that we obtained the partial result  $[M'(\text{false})^p \models_E \mathbf{F}p] = \text{true}$  in the previous iteration. It is easy to show that any instantiation of a *further* parameterisation of  $M'(\text{false})$  satisfies equally many or more  $LTL^+$  properties than the pessimistic instantiation  $M'(\text{false})^p$ . The argument here is that parameterisation always involves a replacement of selected  $\perp$ 's by parameter expressions. While these  $\perp$ 's would be set to *false* under a pessimistic instantiation, the replacing parameter expressions might also take the value *true* in an instantiation of a further parameterised model, which generally leads to the satisfaction of *more*  $LTL^+$  properties. Thus, for our running example we can immediately conclude

$$[M'(\text{false})^p \models_E \mathbf{F}p] = \text{true} \Rightarrow \wedge \begin{array}{l} [M'(\text{false}, \text{false})^p \models_E \mathbf{F}p] = \text{true} \\ [M'(\text{false}, \text{true})^p \models_E \mathbf{F}p] = \text{true} \end{array}$$

<sup>1</sup> This parameterisation step indicates that the value of the unknown predicate  $p$  is *inverted* by taking the transition from  $s_0$  to  $s_2$ , which might be a fact automatically derived from the modelled system via our parameterisation rules defined in [12].

<sup>2</sup> The *optimistic* instantiations that are considered for attempting to *disprove* the property yield a similar tree.



which saves us the explicit consideration of the instantiations  $M'(false, false)^p$  and  $M'(false, true)^p$  and thus significantly reduces the computational effort for solving our example PMC problem. Lemma 3 generalises the capabilities for *instantiation reduction* in iterative parameterisation:

**Lemma 3.** *Let  $M(B_n)$  be an instantiation of a parameterised three-valued Kripke structure  $M(X_n)$  and let  $M(B_n)(X_m)$  be a further parameterisation of  $M(B_n)$ . Then for any  $\psi \in LTL^+$  the following holds:*

1.  $[M(B_n)^p \models_E \psi] = true \Rightarrow \forall B_m \in \{t, f\}^m [M(B_n)(B_m)^p \models_E \psi] = true$
2.  $[M(B_n)^o \models_E \psi] = false \Rightarrow \forall B_m \in \{t, f\}^m [M(B_n)(B_m)^o \models_E \psi] = false$

*Proof.* See <http://www.cs.up.ac.za/cs/ntimm/proofLemma3.pdf>

This lemma establishes the useful relation between results obtained for an instantiation  $M(B_n)^y$ ,  $y \in \{o, p\}$ , and for any ‘successor’ instantiation  $M(B_n)(B_m)^y$ , that allows us to cut down the number of instantiations to be considered in future iterations. In our iterative approach to PMC we thus memorise cases where a *true* result for a *pessimistic* instantiation or a *false* result for an *optimistic* instantiation was obtained. In subsequent iterations the consideration of instances that originate from the further parameterisation of these cases is no longer necessary. Lemma 3 together with Definition 6 gives us the following theorem:

**Theorem 2.** *Let  $M(X_n) = (S, s_0, R(X_n), L(X_n))$  be a parameterised three-valued Kripke structure over AP and  $X = (x_1, \dots, x_n)$ . Moreover, let  $\psi$  be an  $LTL^+$  formula over AP. Then the following holds:*

$$[M(X_n) \models_E \psi] = \begin{cases} true & \text{iff } \forall (B_n) \in \{t, f\}^n \left[ \boxed{\text{with } \neg \exists m \leq n : [M(B_{n-m})^p \models_E \psi] = t} : \right. \\ & \quad \left. ([M(B_n)^p \models_E \psi] = true) \right] \\ false & \text{iff } \forall (B_n) \in \{t, f\}^n \left[ \boxed{\text{with } \neg \exists m \leq n : [M(B_{n-m})^o \models_E \psi] = f} : \right. \\ & \quad \left. ([M(B_n)^o \models_E \psi] = false) \right] \\ \perp & \text{else} \end{cases}$$

The (boxed) constraints on the number of parameter instantiations that need to be considered in order to solve a PMC problem allow to perform PMC with iterative parameterisation with significantly less computational effort. For our running example we were e.g. able to reduce the number of relevant instantiations in the third iteration *by half*. Theorem 2 can be straightforwardly combined with Theorem 1 which enables us to profit from *instantiation reduction* and *path tracking* at the same time. In the following sections we will show that our enhancements of PMC can be transferred to *SAT-based* PMC. We start with a brief review of the basics of SAT-based parameterised three-valued model checking.

## 5 Background: SAT-Based Bounded PMC

In [13] we showed that a PMC problem can be encoded as a parameterised propositional logic formula and then solved by applying SAT solving to each

formula instance, which we briefly review here. A prerequisite is to *bound* the length of the considered paths by some  $k \in \mathbb{N}$ . By  $\Pi_{M_k}$  we denote the set of all  $k$ -bounded paths in a Kripke structure  $M$ . Such finite prefixes  $\pi(0) \dots \pi(k)$  can still represent infinite paths if the prefix has a  $k$ -loop, i.e. the last state  $\pi(k)$  has a successor state that is also part of the prefix. For the bounded evaluation of  $LTL^+$  properties we have to distinguish between paths *with* and *without* a  $k$ -loop.

**Definition 7 (Three-Valued Bounded Evaluation of  $LTL^+$ ).** *Let  $M = (S, s_0, R, L)$  over  $AP$  be a complement-closed three-valued Kripke structure, let  $k \in \mathbb{N}$  and let  $\pi$  be a path of  $M$  without a  $k$ -loop. Then the  $k$ -bounded evaluation of an  $LTL^+$  formula  $\psi$  on  $\pi$ , written  $[\pi \models_k^i \psi]$  where  $i \in \mathbb{N}$  with  $i \leq k$  denotes the current position along the path, is inductively defined as follows*

$$\begin{aligned} [\pi \models_k^i p] &:= L(\pi(i), p) \\ [\pi \models_k^i \psi \vee \psi'] &:= [\pi \models_k^i \psi] \vee [\pi \models_k^i \psi'] \\ [\pi \models_k^i \mathbf{G}\psi] &:= false \\ [\pi \models_k^i \mathbf{F}\psi] &:= \bigvee_{j=i}^k ([\pi \models_k^j \psi] \wedge \bigwedge_{l=i}^{j-1} R(\pi(l), \pi(l+1))) \end{aligned}$$

If  $\pi$  has a  $k$ -loop, then  $[\pi \models_k^i \psi] := [\pi^i \models \psi]$ . Moreover, the existential value of  $\psi$  in  $M$  under the bounded semantics is  $[M \models_{E,k} \psi] := \bigvee_{\pi \in \Pi_{M_k}} [\pi \models_k^0 \psi]$ .

If all  $k \in \mathbb{N}$  are considered then the bounded semantics is equivalent to the unbounded one:  $[M \models_E \psi] = \bigvee_{k \in \mathbb{N}} [M \models_{E,k} \psi]$ . Typically, the bound is iteratively increased until the property of interest can be either proven or a completeness threshold is reached. The bounded semantics for 3MC can be extended to PMC:

**Definition 8 (Bounded Parameterised Three-Valued Model Checking).** *Let  $M(X_n) = (S, s_0, R(X_n), L(X_n))$  be a parameterised three-valued KS over  $AP$  and  $X_n$ . Let  $\psi$  be an  $LTL^+$  formula over  $AP$  and  $k \in \mathbb{N}$ . The existential value of  $\psi$  in  $M(X_n)$  under the bounded semantics is*

$$[M(X_n) \models_{E,k} \psi] := \begin{cases} true & \text{if } \forall B_n \in \{t, f\}^n ([M(B_n)^p \models_{E,k} \psi] = true) \\ false & \text{if } \forall B_n \in \{t, f\}^n ([M(B_n)^o \models_{E,k} \psi] = false) \\ \perp & \text{else} \end{cases}$$

As shown in [13] bounded PMC can be reduced to the new satisfiability problem  $SAT_{X3}$ . For a parameterised three-valued Kripke structure  $M(X_n)$  over  $AP$  and  $X_n$ , an  $LTL^+$  formula  $\psi$  and a bound  $k \in \mathbb{N}$ , a parameterised propositional logic formula  $F(X_n)_k$  is constructed such that  $[M(X_n) \models_{E,k} \psi] = SAT_{X3}(F(X_n)_k)$ .

Solving  $SAT_{X3}$  reduces to solving classical SAT for each possible parameter instantiation. We briefly review how  $F(X_n)_k$  is constructed for a given PMC problem. The construction of  $F(X_n)_k$  is divided into the translation of  $M(X_n)$  into a formula  $\llbracket M(X_n) \rrbracket_k$  and the translation of  $\psi$  into a formula  $\llbracket \psi \rrbracket_k$ . The encoding of  $M(X_n)$  first requires to encode its states as Boolean formulae. For

this, a set of Boolean atoms  $\{A, B, \dots\}$  is introduced. Let  $PL$  be the set of propositional formulae over  $\{A, B, \dots\}$  and the constants *true* and *false*. Then an encoding of the states of a Kripke structure is defined as follows.

**Definition 9 (State Encoding).** *Let  $M(X_n) = (S, s_0, R(X_n), L(X_n))$  be a parameterised Kripke structure. A Boolean encoding of its states corresponds to an injective mapping  $e : S \rightarrow PL$  where  $\forall s \in S : e(s)$  is a conjunction of literals.*

$M(X_n)$  can now be translated into a formula  $\llbracket M(X_n) \rrbracket_k$  that characterises  $k$ -bounded paths in  $M(X_n)$ . As we consider states as parts of such paths, the state encoding is extended by index values  $i \in \{0, \dots, k\}$  where  $i$  denotes the position along a path. We get an extended set of indexed atoms  $\{A_0, B_0, \dots, A_k, B_k, \dots\}$ .

**Definition 10 (Kripke Structure Encoding).** *Let  $M(X_n) = (S, s_0, R(X_n), L(X_n))$  be a parameterised three-valued KS and  $e$  an encoding of its states. We define  $Init_0$  as the predicate characterising the initial state of  $M(X_n)$  with*

$$Init_0 := e(s_0)_0$$

and  $T_{i,i+1}$  as the predicate that characterises the transitions of  $M(X_n)$  with

$$T_{i,i+1} := \bigvee_{s \in S} \bigvee_{s' \in S} e(s)_i \wedge e(s')_{i+1} \wedge R(X_n)(s, s').$$

Then the entire encoding of  $M(X_n)$  for a bound  $k \in \mathbb{N}$  is defined as

$$\llbracket M(X_n) \rrbracket_k := Init_0 \wedge \bigwedge_{i=0}^{k-1} T_{i,i+1}$$

The second part of the encoding concerns the  $LTL^+$  formula  $\psi$ . This generally requires to distinguish the cases where  $\psi$  is evaluated on a path *with* and *without* a loop. Due to space limitations we only consider the case without a loop here.

**Definition 11 ( $LTL^+$  Encoding without Loop).** *Let  $p$  be an atomic predicate,  $\psi$  and  $\psi'$   $LTL^+$  formulae, and  $k, i \in \mathbb{N}$  with  $i \leq k$ .*

$$\begin{aligned} \llbracket p \rrbracket_k^i &:= \bigvee_{s \in S} e(s)_i \wedge L(s, p) & \llbracket \mathbf{G}\psi \rrbracket_k^i &:= false \\ \llbracket \psi \vee \psi' \rrbracket_k^i &:= \llbracket \psi \rrbracket_k^i \vee \llbracket \psi' \rrbracket_k^i & \llbracket \mathbf{F}\psi \rrbracket_k^i &:= \bigvee_{j=i}^k \llbracket \psi \rrbracket_k^j \end{aligned}$$

The overall encoding of  $[M(X_n) \models_{E,k} \psi]$  is now  $F(X_n)_k := \llbracket M(X_n) \rrbracket_k \wedge \llbracket \psi \rrbracket_k^0$ .  $F(X_n)_k$  is defined over atoms but also over  $X_n$  and  $\perp$ . Thus,  $SAT_{X3}$  is not a standard satisfiability problem. It reduces to multiple instances of classical SAT.

$$SAT_{X3}(F(X_n)_k) := \begin{cases} true & \text{if } \forall B_n \in \{t, f\}^n (SAT(F(B_n)_k^p) = true) \\ false & \text{if } SAT(F(X_n)_k^o) = false \\ \perp & \text{else} \end{cases}$$

where  $F(B_n)_k^p = F(X_n)_k[X_n/B_n][\perp/false]$  and  $F(X_n)_k^o = F(X_n)_k[\perp/true]$ . Since  $SAT(F((X_n)_k^o) = f)$  is equivalent to  $\forall B_n \in \{t, f\}^n (SAT(F((B_n)_k^o) = f))$ , checking whether  $SAT_{X3}$  yields *false* requires a single SAT test only. The result of the  $SAT_{X3}$  test is equivalent to the result of the encoded PMC problem [13]:

**Lemma 4.** *Let  $M(X_n)$  be a parameterised three-valued Kripke structure over a set of predicates  $AP$ , let  $\psi$  be an  $LTL^+$  formula over  $AP$ , and  $k \in \mathbb{N}$ . Then*

$$[M(X_n) \models_{E,k} \psi] = SAT_{X3}(F(X_n)_k)$$

Thus, bounded PMC can be reduced to multiple instances of SAT. Subsequently, we show that in iterative parameterisation, solving these instances can be efficiently performed via *interpretation validation* in place of SAT solving.

## 6 Bounded PMC via Interpretation Validation

As we have seen, bounded PMC reduces to multiple instances of classical SAT solving. Given a propositional logic formula  $F$  over a set of Boolean atoms  $\mathcal{A}$ , SAT is the (NP-complete) problem of determining whether there exists an interpretation  $I : \mathcal{A} \rightarrow \{true, false\}$  that satisfies  $F$ . The previously introduced encoding  $F(X_n)_k$  of a bounded PMC problem  $[M(X_n) \models_{E,k} \psi]$  has the nice feature that each interpretation that satisfies an instantiation of the encoding exactly characterises a path in the corresponding model that witnesses the property  $\psi$  within  $k$  steps [12]. Now we will show that, based on this feature, the solving performance of SAT-based PMC with iterative parameterisation can be considerably improved. De facto, we transfer our concept *path tracking* in explicit (non-encoded) PMC (Sect. 3) to the SAT scenario. Here the satisfiability tests associated with an encoded problem can be replaced by the significantly less expensive *validation* of satisfying interpretations discovered in earlier iterations.

For illustration, we consider again the pure three-valued Kripke structure  $M$  and its parameterisation  $M(x_1)$  from Sect. 2. For the states of  $M$  we choose the encoding  $e(s_0)_i = \neg A_i \wedge \neg B_i$ ,  $e(s_1)_i = \neg A_i \wedge B_i$  and  $e(s_2)_i = A_i \wedge \neg B_i$  over the set of atoms  $\mathcal{A} = \{A_0, B_0, \dots, A_k, B_k\}$ . The bounded model checking problem  $[M \models_{E,k} \mathbf{F}p]$  can now be logically encoded as follows

$$\begin{aligned} F_k &= Init_0 \wedge \bigwedge_{i=0}^{k-1} T_{i,i+1} \wedge [\mathbf{F}p]_k^0 \\ &= (\neg A_0 \wedge \neg B_0) \wedge \bigwedge_{i=0}^{k-1} ((\neg A_i \wedge \neg B_i \wedge \neg A_{i+1} \wedge \neg B_{i+1} \wedge true) \\ &\quad \vee (\neg A_i \wedge \neg B_i \wedge \neg A_{i+1} \wedge B_{i+1} \wedge \perp_{(s_0,s_1)}) \vee (\neg A_i \wedge \neg B_i \wedge A_{i+1} \wedge \neg B_{i+1} \wedge \perp_{(s_0,s_2)}) \\ &\quad \vee (\neg A_i \wedge B_i \wedge A_{i+1} \wedge \neg B_{i+1} \wedge true) \vee (A_i \wedge \neg B_i \wedge A_{i+1} \wedge \neg B_{i+1} \wedge true)) \\ &\quad \wedge \bigvee_{i=0}^k ((\neg A_i \wedge \neg B_i \wedge false) \vee (\neg A_i \wedge B_i \wedge false) \vee (A_i \wedge \neg B_i \wedge true)) \end{aligned}$$

Note that we annotated the  $\perp$ 's in  $F_k$  with the transitions in  $M$  they are associated with, which helps to distinguish individual  $\perp$ 's in the later parameterisation. Since  $M$  is *pure* three-valued,  $SAT_{X3}$  reduces to two satisfiability instances  $F_k^p = F_k[\perp/false]$  and  $F_k^o = F_k[\perp/true]$ .  $F_k^p$  is the propositional logic equivalent of the explicit PMC problem  $[M^p \models_{E,k} \mathbf{F}p]$ , whereas  $F_k^o$  is the equivalent of  $[M^o \models_{E,k} \mathbf{F}p]$ . We get  $SAT(F_k^p) = false$  and  $SAT(F_k^o) = true$ , which gives us *unknown* as the overall result. Hence, there exist interpretations that satisfy  $F_k^o$  but there is no interpretation that satisfies  $F_k^p$ . Several approaches for the efficient computation of *all* satisfying interpretations of a propositional logic formula have been introduced [8, 14, 15], which we will discuss in detail in the related work section. These approaches for solving the so-called AllSAT problem

allow us to compute the set  $\mathcal{I}_{F_k^o} = \{I \mid I(F_k^o) = \text{true}\}$  of all interpretations that make  $F_k^o$  true. As stated before, each  $I \in \mathcal{I}_{F_k^o}$  exactly characterises a  $k$ -bounded path  $\pi$  with  $[\pi \models_k^0 \mathbf{F}p]_{M^o} = \text{true}$ . For instance, the interpretation  $I : A_0 \mapsto \text{false}, B_0 \mapsto \text{false}, A_1 \mapsto \text{false}, B_1 \mapsto \text{true}, A_2 \mapsto \text{true}, B_2 \mapsto \text{false}$  for a 2-bounded encoding  $F_2$  describes the path  $\pi = s_0 s_1 s_2$  in  $M$ . Since  $\pi$  contains the *unknown* transition  $(s_0, s_1)$  we call it an *unconfirmed witness* for the property of interest. Based on such unconfirmed witnesses the parameterisation rules we defined in [12] can be automatically and iteratively applied, which, in this particular case, gives us the parameterised Kripke structure  $M(x_1)$  from our running example. This parameterisation step can be straightforwardly lifted to the level of the propositional encoding. Based on an analysis of the interpretations in  $\mathcal{I}_{F_2^o}$  the current encoding can be parameterised to  $F(x_1)_2 := F_2[\perp_{(s_0, s_1)}/\neg x_1][\perp_{(s_0, s_2)}/x_1]$ .

Solving  $\text{SAT}_{X3}(F(X_n)_k)$  generally requires to solve SAT *from scratch* for each instantiation of  $F(X_n)_k$ . However, since each interpretation satisfying the non-parameterised  $F_k^o$  characterises a path  $\pi$  with  $[\pi \models_k^0 \psi]_{M^o} = \text{true}$  and vice versa, we also have a one-to-one correspondence between the elements of the set of interpretations  $\mathcal{I}_{F_k^o}$  and the set of paths  $\Pi_{M_k^o}^\psi$  (compare Sect. 3). In accordance with the results from Sect. 3,  $\mathcal{I}_{F_k^o}$  is a superset of any  $\mathcal{I}_{F(B_n)_k^y}$  where  $F(B_n)_k = F(X_n)_k[x_1, \dots, x_n/b_1, \dots, b_n]$  is an instantiation of an arbitrary parameterisation  $F(X_n)_k$  of  $F_k$  and  $y \in \{o, p\}$ . In an iterative approach, this allows us to solve encoded PMC problems via *interpretation validation*. Theorem 1 together with Lemma 4 gives us the following theorem:

**Theorem 3.** *Let  $[M(X_n) \models_{E,k} \psi]$  be a parameterisation of a bounded 3MC problem  $[M \models_{E,k} \psi]$ . Moreover, let  $F_k$  and  $F(X_n)_k$  be the corresponding propositional logic encodings and  $\mathcal{I}_{F_k^o} = \{I \mid I(F_k^o) = \text{true}\}$ . Then:*

$$[M(X_n) \models_{E,k} \psi] = \begin{cases} \text{true} & \text{iff } \forall B_n \in \{t, f\}^n \exists I \in \mathcal{I}_{F_k^o} : I(F(B_n)_k^p) = \text{true} \\ \text{false} & \text{iff } \forall I \in \mathcal{I}_{F_k^o} : I(F(X_n)_k^o) = \text{false} \\ \perp & \text{else} \end{cases}$$

Hence, instead of solving one NP-complete SAT problem per instance we just need to validate  $|\mathcal{I}_{F_k^o}|$  interpretations per instance, which requires linear time and space per interpretation. For the optimistic completion  $F_2^o$  of our initially non-parameterised example encoding we can efficiently compute the set  $\mathcal{I}_{F_2^o}$  via AllSAT methods.  $\mathcal{I}_{F_2^o}$  consists of three interpretations  $I_1, I_2$  and  $I_3$  characterising the bounded paths  $\pi_1 = s_0 s_1 s_2$ ,  $\pi_2 = s_0 s_0 s_2$  and  $\pi_3 = s_0 s_2 s_2$ . In the next iteration we consider each instantiation of the parameterised encoding  $F(x_1)_2$  and validate the interpretations:  $I_1$  satisfies  $F(\text{false})_2^p$  and  $I_2$  satisfies  $F(\text{true})_2^p$ . Thus, we can conclude that  $[M(x_1) \models_{E,2} \mathbf{F}p] = \text{true}$ . Bounded PMC via *interpretation validation* is straightforwardly compatible with *instantiation reduction* (compare Sect. 4). In the next section we introduce our model checking framework for PMC problems that implements our proposed enhancements.

## 7 PMC Framework with Iterative Parameterisation

We have developed a prototype (All)SAT-based bounded model checker for PMC problems. It employs the solver library Sat4j [10] and implements the algorithm for efficient AllSAT solving proposed in [15]. The checker iterates over the bound  $k$ , starting with  $k = 0$ , and over possible parameterisations of the initially non-parameterised input model. In each iteration that yields *unknown* it is checked whether any parameterisation rule from [12] is applicable. If further parameterisation is currently not feasible the bound is incremented. Our checker supports two modes: In the *basic* mode in each iteration all instances of the encoded problem are processed via *incremental SAT solving* [6]: For acceleration, learned conflict clauses are reused between the solvers processing different SAT instances, in case the learning happened based on a common subformula. No instantiation reduction is applied in the basic mode. The *enhanced* mode works as follows: After every bound incrementation the AllSAT problem for the propositional logic encoding  $F_k^o$  is solved<sup>3</sup>, which gives us the set  $\mathcal{I}_{F_k^o}$  of all satisfying interpretations. Between the AllSAT problems for different bounds we also employ incremental solving, i.e. we reuse learned clauses. After every parameterisation step we apply *interpretation validation* based on  $\mathcal{I}_{F_k^o}$  for the instantiations of the parameterised encoding  $F(X_n)_k$ . Via *instantiation reduction* we additionally narrow the number of instantiations that actually have to be considered. The following procedure characterises a single iteration and illustrates how our enhancements have been implemented:

```

Data: current  $F(X_n)_k^p$  and  $\mathcal{I}_{F_k^o}$ 
for all  $B_n \in \{true, false\}^n$  do
  | if  $SATResult(F(B_{n-1})_k^p) = true$  then
  |   |  $SATResult(F(B_n)_k^p) := true$ 
  | else
  |   |  $SATResult(F(B_n)_k^p) := validate(\mathcal{I}_{F_k^o}, F(B_n)_k^p)$ 

```

Here *SATResult* is the database where we store satisfiability results. It allows us to look up results from past iterations: If  $SATResult(F(B_{n-1})_k^p) = true$  then we can skip the computation of the SAT result for  $F(B_n)_k^p$  and immediately conclude that it is *true* as well, which implements *instantiation reduction*. The function call  $validate(\mathcal{I}_{F_k^o}, F(B_n)_k^p)$  returns whether any interpretation from  $\mathcal{I}_{F_k^o}$  is satisfying for  $F(B_n)_k^p$ , and thus, implements *interpretation validation*. We use an analogous procedure for processing  $F(X_n)_k^o$  where *false* results from past iterations are exploited in a similar way.

In preliminary experiments we compared the run-time of PMC in the *basic* and the *enhanced* mode. We considered a number of initially non-parameterised problems that we encoded in propositional logic and then iteratively parameterised until a definite result could be obtained. In the enhanced mode we achieved substantial savings in verification time. While a more extensive exper-

<sup>3</sup> The encoding has been transferred into conjunctive normal form via Tseitin transformation. Thus, we only consider interpretations that differ wrt. the valuation of the original problem variables, and not wrt. newly introduced auxiliary variables.

imental evaluation is in preparation, we so far observed that the actual performance gain due to the application of interpretation validation instead of SAT solving results from the trade-off between the extra costs of solving AllSAT for the initial problem and the savings in the later iterations: The more iterations were needed, the more we generally saved in the enhanced mode. And, the smaller the set  $\mathcal{I}_{F_k^o}$ , the more we profited from interpretation validation in PMC. – In terms of a potential reduction of the *size* of  $\mathcal{I}_{F_k^o}$ , we can actually derive another beneficial fact based on Lemma 3 and the encoding definitions: If an  $I \in \mathcal{I}_{F_k^o}$  does not satisfy *any* instantiation in the current iteration then it will also not satisfy any instantiation in a future iteration. This allows to immediately remove such an interpretation from  $\mathcal{I}_{F_k^o}$ , which involves further computational savings.

## 8 Related Work

*Iterative* verification approaches with *reuse of results* have been considered in several ways. An established approach is *counterexample-guided abstraction refinement* (CEGAR) [4]. The CEGAR-based model checker YASM [9] reuses results between abstraction iterations as follows: Each iteration involves the computation of the set of states from which an error can be definitely reached. Instead of checking the error reachability from scratch in a subsequent iteration, YASM just checks whether any state from the previously computed set is reachable. Another common approach is SAT-based *incremental bounded model checking* [6]. Each iteration corresponds to solving a SAT instance that encodes the model checking problem for a different bound. Results are reused in the sense of sharing conflict clauses that have been learned based on common parts of the SAT instances. While these approaches iterate over *predicate abstractions* and *bounds*, our approach is based on iterative *parameterisation*. Each iteration involves the consideration of all instantiations of a parameterised model. A related approach is *conditional model checking* (CMC) [1]. Here the property is checked under different conditions that restrict *which* part of the model is explored. Multiple checks under complementary conditions allow to obtain an ‘unconditional’ result. CMC has so far not been considered in an iterative context. In our PMC approach we do not check *different parts* of a model but *different approximations* (parameter instantiations) of the system to be verified, that together yield an exact result. Our approach relies on AllSAT, for which several memory and time efficient solving techniques have been proposed. The technique from [15] uses *blocking clauses* to avoid that the same interpretations are found again. It is based on an *incremental* search for interpretations that prevents starting search from scratch after an interpretation was found. Moreover, generated blocking clauses are *merged*, which improves memory efficiency. Other AllSAT solving techniques that follow similar concepts have been introduced in [8, 14].



## 9 Conclusion and Outlook

We introduced an *iterative* approach to SAT-based parameterised three-valued model checking that substantially overcomes the former drawbacks of parameterisation in terms of computational costs. Our new concept *instantiation reduction* allows to exploit results from past iterations in order to narrow the *number* of instantiations that have to be considered in future iterations. With *interpretation validation* we introduced a concept that significantly reduces the *effort* for solving the individual instantiations occurring in PMC. We proved the soundness of our PMC-improving concepts and integrated them into a prototype model checker. Preliminary experiments revealed that our approach leads to a practically relevant speed-up of PMC. It allows to profit from the extra precision of parameterisation without suffering from the previous performance drawbacks.

We are currently working on a concept for instantiation reduction based on results obtained for *different bounds*: Partial results for the instances of an encoding  $F(X_n)_k$  can be also exploited for further reducing the number of relevant instances of all subsequent encodings  $F(X_{n+m})_{k+l}$ . Another direction for future research is the development of a concept for *summarising* interpretations in  $\mathcal{I}_{F_k^o}$  into equivalence classes: For interpretations that characterise paths that are bisimilar or stuttering equivalent wrt. the property of interest it is sufficient to have just one representative in  $\mathcal{I}_{F_k^o}$ , which allows further computational savings due to interpretation validation. We also plan to combine our iterative framework with a classical abstraction refinement procedure: In case parameterisation and bound incrementation do not yield the necessary precision for a definite result, more predicates can be added to the abstract model. *AllSAT-based Predicate abstraction and refinement* [5] can then work hand-in-hand with our AllSAT approach. Finally, the summarisation of all SAT instances occurring per iteration to a single *quantified* Boolean formula (QBF) and the subsequent application of QBF interpretation validation is another direction for future investigation.

## References

1. Beyer, D., Henzinger, T.A., Keremoglu, M.E., Wendler, P.: Conditional model checking: a technique to pass information between verifiers. In: Proceedings of the ACM SIGSOFT FSE 2012, pp. 57:1–57:11. ACM, New York (2012)
2. Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y.: Bounded Model Checking. Handbook of Satisfiability, vol. 185, pp. 457–481. IOS Press, Amsterdam (2009)
3. Bruns, G., Godefroid, P.: Model checking partial state spaces with 3-valued temporal logics. In: Halbwachs, N., Peled, D.A. (eds.) CAV 1999. LNCS, vol. 1633, pp. 274–287. Springer, Heidelberg (1999)
4. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: Emerson, EAllen, Sistla, Aravinda Prasad (eds.) CAV 2000. LNCS, vol. 1855. Springer, Heidelberg (2000)



5. Sharygina, N., Yorav, K., Clarke, E., Kroning, D.: SATABS: SAT-based predicate abstraction for ANSI-C. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 570–574. Springer, Heidelberg (2005)
6. Eén, N., Sörensson, N.: Temporal induction by incremental SAT solving. *Electron. Notes Theor. Comput. Sci.* **89**(4), 543–560 (2003)
7. Fitting, M.: Kleene’s three valued logics and their children. *Fundamenta Informaticae* **20**(1–3), 113–131 (1994)
8. Schuster, A., Grumberg, O., Yadgar, A.: Memory efficient all-solutions SAT solver and its application for reachability analysis. In: Hu, A.J., Martin, A.K. (eds.) FMCAD 2004. LNCS, vol. 3312, pp. 275–289. Springer, Heidelberg (2004)
9. Wei, O., Chechik, M., Gurfinkel, A.: YASM: a software model-checker for verification and refutation. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 170–174. Springer, Heidelberg (2006)
10. Le Berre, D., Parrain, A.: The SAT4J library, release 2.2. *J. Satisfiability Boolean Model. Comput.* **7**, 59–64 (2010)
11. Timm, N.: Three-Valued Abstraction and Heuristic-Guided Refinement for Verifying Concurrent Systems, Ph.D thesis, University of Paderborn (2013)
12. Gruner, S., Timm, N.: Parameterisation of three-valued abstractions. In: Braga, C., Martí-Oliet, N. (eds.) SBMF 2014. LNCS, vol. 8941, pp. 162–178. Springer, Heidelberg (2015)
13. Sibanda, P., Gruner, S., Timm, N.: Parallel SAT-based parameterised three-valued model checking. In: Fischer, B., Geldenhuys, J. (eds.) SPIN 2015. LNCS, vol. 9232, pp. 242–259. Springer, Heidelberg (2015)
14. Yu, Y., Subramanyan, P., Tsiskaridze, N., Malik, S.: All-SAT using minimal blocking clauses. *VLSI Design 2014*, pp. 86–91 (2014)
15. Zhao, W., Wu, W.: Asig: An all-solution sat solver for cnf formulas. In: 11th IEEE International Conference on CAD/Graphics, pp. 508–513, August 2009

Formal Methods: Foundations and Applications

18th Brazilian Symposium, SBMF 2015, Belo Horizonte,  
Brazil, September 21-22, 2015, Proceedings

Cornélio, M.; Roscoe, B. (Eds.)

2016, XVIII, 195 p. 62 illus. in color., Softcover

ISBN: 978-3-319-29472-8