Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 702165, 13 pages
http://dx.doi.org/10.1155/2014/702165

Research Article

Counterexample-Preserving Reduction for Symbolic Model Checking

Wanwei Liu, Rui Wang, Xianjin Fu, Ji Wang, Wei Dong, and Xiaoguang Mao

School of Computer Science, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Wanwei Liu; wwliu@nudt.edu.cn

Received 12 February 2014; Accepted 14 April 2014; Published 14 May 2014

Academic Editor: Xiaoyu Song

The cost of LTL model checking is highly sensitive to the length of the formula under verification. We observe that, under some specific conditions, the input LTL formula can be reduced to an easier-to-handle one before model checking. In such reduction, these two formulae need not to be logically equivalent, but they share the same counterexample set w.r.t the model. In the case that the model is symbolically represented, the condition enabling such reduction can be detected with a lightweight effort (e.g., with SAT-solving). In this paper, we tentatively name such technique "counterexample-preserving reduction" (CePRe, for short), and the proposed technique is evaluated by conducting comparative experiments of BDD-based model checking, bounded model checking, and property directed reachability-(IC3) based model checking.

1. Introduction

LTL [1] is one of the most frequently used specification languages in model checking (cf. [2]). It designates properties over a linear structure, which can be viewed as an execution of the program. The task of LTL model checking is to search the state space (explicitly or implicitly), with the goal of detecting the existence of feasible traces violating the specification. If such traces exist, the model checker will report one of them as a "counterexample"; otherwise, the model checker will give an affirmative report.

It can be shown that the complexity of LTL model checking for $M \models \varphi$ is in $\mathcal{O}(|M| \times 2^{|\varphi|})$; meanwhile, the nesting depth of temporal operators might be the major factor affecting the cost in compiling LTL formulae. Hence, it is reasonable to simplify the specification before conducting model checking. For example, in [3], Somenzi and Bloem provided a set of rewriting schemas for simplifying LTL specifications, and these rewriting schemas preserve logical equivalence.

One may argue that "a majority of LTL formulae used in real applications are simple, succinct rather than complicated." Nevertheless, we might need to observe the following facts.

(i) Some LTL formula, for example $\mathbf{F}(p\mathbf{U}q)$, is usually considered to be a "simple" one. Nevertheless, it can be further simplified to $\mathbf{F}q$, and this fact tends to be ignored (on one hand, $p\mathbf{U}q$ implies $\mathbf{F}q$, and hence $\mathbf{F}(p\mathbf{U}q)$ implies $\mathbf{FF}q$ (i.e., $\mathbf{F}q$); on the other hand, q implies $p\mathbf{U}q$, and hence $\mathbf{F}q$ implies $\mathbf{F}(p\mathbf{U}q)$).

(ii) Indeed, people do use complicated specifications in the real industrial field, as well as in some standard benchmark (cf. [4]).

(iii) Last but not least, not all specifications are designated manually. Actually, some formulae are generated by specification-generaton-tools (e.g., ProSpec). Indeed, one may find that lots of these machine-generated specifications can be simplified.

Symbolic model checking [5] is one of the most significant breakthroughs in model checking, and two major fashions of symbolic model checking are widely used: one is the BDD-based manner [6] and the other is SAT-based manner, such as BMC [7] or PDR [8–11] algorithms.

Instead of using an explicit representation, the symbolic approach represents state space with a series of Boolean formulae. This enables implicit manipulation of the verification process and it usually leads to an efficient

implementation [12]. Meanwhile, the symbolic encoding of transitions and invariants of the model provides heuristic information to simplify the specification. For example,

(i) the formulae $p\mathbf{U}q$ and $(r\mathbf{U}p)\mathbf{U}q$ can be, respectively, reduced as q and $(r\mathbf{U}p) \vee q$, if we know that $p \rightarrow q$ holds everywhere in the model;

(ii) each occurrence of $\mathbf{G}\theta$ in the specification can be replaced with $\top$ (i.e., logically true), if we can inductively infer that the Boolean formula θ holds at each reachable state in the model.

Actually, we can make sure of these conditions with the following efforts.

(i) To ensure that "$p \rightarrow q$ holds everywhere in the model," one possible way is to make sure that $p \rightarrow q$ is an *invariant* in the model—that is, just to examine if $\rho \wedge \neg(p \rightarrow q)$ is unsatisfiable (we in the latter denote it as $\rho \vdash p \rightarrow q$), where ρ is the Boolean encoding of the model's transition relation.

(ii) Likely, to justify that θ holds at each reachable state (note that a "dead-end" has no infinite path starting from it, and hence we may safely omit dead-ends in the model when doing this), it suffices to ensure that $\theta_0 \vdash \theta$ and $\rho \vdash \theta \rightarrow \theta'$, where θ_0 is the initial condition of the model.

We could do this because the component ρ should be satisfied at each transition step. Hence, it encloses both "local invariants" and "transitional invariants". For example, if $\rho = p \wedge (q \rightarrow q')$, then we may consider p as a local invariant, whereas $q \rightarrow q'$ is a transitional invariant.

Hence, this provides an opportunity to replace the specification with a simpler one, accompanied with some lightweight extra task of condition detection. Even if such detection fails, the overhead is usually negligible. More importantly, such reductions can be performed before starting model checking. At the same time, such kind of simplification is not always feasible to conduct manually; the reason is that the scale of boolean facilities is too large to be effectively handled by humans. Therefore, it is reasonable of leveraging the SAT-solvers to accomplish this task.

In this paper, we systematically investigate the above idea and tentatively name this technique *counter example-preserving reduction* (CePRe, for short). To justify it, we have extended NuSMV and implemented CePRe as an upfront option for LTL model checking. Subsequently, we conduct experiments over both industrial benchmarks and randomly generated cases. Experimental results show that CePRe can improve the efficiency significantly.

This paper is organized as follows. Section 2 revisits some basic notions. Section 3 introduces the details of the CePRe technique and Section 4 gives a formal treatment of performance analysis. In Section 5, the experimental results over industrial benchmarks and over random generated cases are given. We summarize the whole paper with Section 6.

2. Preliminaries

We presuppose a countable set $\mathscr{P}$ of *atomic propositions*, ranging over p, q, and so forth, and for each proposition $p \in \mathscr{P}$, we create a *primed version* p' (not belonging to $\mathscr{P}$) for it. For each set $\mathscr{V} \subseteq \mathscr{P}$, we define $\mathscr{V}' \triangleq \{p' \mid p \in \mathscr{V}\}$. We use $\mathbb{B}(\mathscr{V})$ to denote the set of Boolean formulae over $\mathscr{V}$. Similarly, we denote by $\mathbb{B}(\mathscr{V} \cup \mathscr{V}')$ the set of Boolean formulae built up from $\mathscr{V} \cup \mathscr{V}'$. The scope of the *prime* operator can be naturally lifted to Boolean formulae over $\mathbb{B}(\mathscr{V})$, by defining

$$\begin{aligned} &\top' = \top, \qquad \bot' = \bot, \qquad (\neg\theta)' \triangleq \neg\theta', \\ &(\theta_1 \longrightarrow \theta_2)' \triangleq \theta_1' \longrightarrow \theta_2'. \end{aligned} \tag{1}$$

An *assignment* is a subset $\mathscr{V}$ of $\mathscr{P}$. Intuitively, it assigns 1 (or true) to the propositions belonging to $\mathscr{V}$ and assigns 0 (or false) to the other propositions. For each $\mathscr{V} \subseteq \mathscr{U} \subseteq \mathscr{P}$ and $\theta \in \mathbb{B}(\mathscr{U})$, we denote by $\mathscr{V} \Vdash \theta$ if θ is evaluated to 1 under the assignment $\mathscr{V}$.

A *united assignment* is a pair $(\mathscr{V}_1, \mathscr{V}_2)$, where both $\mathscr{V}_1$ and $\mathscr{V}_2$ are subsets of $\mathscr{P}$. It assigns 1 to the propositions belonging to $\mathscr{V}_1 \cup \mathscr{V}_2'$ and assigns 0 to the other propositions. Suppose that $\mathscr{V}_1, \mathscr{V}_2 \subseteq \mathscr{U} \subseteq \mathscr{P}$ and $\theta \in \mathbb{B}(\mathscr{U} \cup \mathscr{U}')$; we also write $(\mathscr{V}_1, \mathscr{V}_2) \Vdash \theta$ if θ is evaluated to 1 under the united assignment $(\mathscr{V}_1, \mathscr{V}_2)$.

LTL formulae can be inductively defined as follows.

(i) $\bot$ and $\top$ are LTL formulae.

(ii) Each proposition $p \in \mathscr{P}$ is an LTL formula.

(iii) If both φ_1 and φ_2 are LTL formulae, so does $\varphi_1 \rightarrow \varphi_2$.

(iv) If φ is an LTL formula, then $\mathbf{X}\varphi$ and $\mathbf{Y}\varphi$ are LTL formulae.

(v) If φ_1 and φ_2 are LTL formulae, then both $\varphi_1\mathbf{U}\varphi_2$ and $\varphi_1\mathbf{S}\varphi_2$ are LTL formulae.

Semantics of an LTL formula is defined w.r.t. a *linear structure* $\pi \in (2^{\mathscr{P}})^\omega$ (i.e., π is an infinite word over the alphabet $2^{\mathscr{P}}$) and a position $i < \omega$. Inductively:

(i) $\pi, i \vDash \top$ and $\pi, i \nvDash \bot$;

(ii) $\pi, i \vDash p$ if and only if $\pi(i) \Vdash p$ (where $\pi(i)$ is the ith letter of π, which can be viewed as an assignment);

(iii) $\pi, i \vDash \varphi_1 \rightarrow \varphi_2$ if and only if either $\pi, i \nvDash \varphi_1$ or $\pi, i \vDash \varphi_2$;

(iv) $\pi, i \vDash \mathbf{X}\varphi$ if and only if $\pi, i + 1 \vDash \varphi$;

(v) $\pi, i \vDash \mathbf{Y}\varphi$ if and only if $i > 0$ and $\pi, i - 1 \vDash \varphi$;

(vi) $\pi, i \vDash \varphi_1\mathbf{U}\varphi_2$ if and only if there is some $j \geq i$, s.t. $\pi, j \vDash \varphi_2$, and $\pi, k \vDash \varphi_1$ for each $i \leq k < j$;

(vii) $\pi, i \vDash \varphi_1\mathbf{S}\varphi_2$ if and only if there is some $j \leq i$, s.t. $\pi, j \vDash \varphi_2$ and $\pi, k \vDash \varphi_1$ for each $i \geq k > j$.

For the sake of convenience, we may directly write $\pi, 0 \vDash \varphi$ as $\pi \vDash \varphi$.

As usual, we employ some derived Boolean connectives such as

$$\begin{aligned} &\neg\varphi \triangleq \varphi \longrightarrow \bot, \qquad \varphi \vee \psi \triangleq \neg\varphi \longrightarrow \psi, \\ &\varphi \wedge \psi \triangleq \neg(\neg\varphi \vee \neg\psi) \end{aligned} \tag{2}$$

and derived temporal operators such as

$$\begin{aligned} &\mathbf{F}\varphi \triangleq \top\mathbf{U}\varphi, \qquad \mathbf{Z}\varphi \triangleq \neg\mathbf{Y}\neg\varphi, \qquad \mathbf{O}\varphi \triangleq \top\mathbf{S}\varphi, \\ &\mathbf{G}\varphi \triangleq \neg\mathbf{F}\neg\varphi, \qquad \mathbf{H}\varphi \triangleq \neg\mathbf{O}\neg\varphi, \\ &\varphi\mathbf{R}\psi \triangleq \neg\left(\neg\varphi\mathbf{U}\neg\psi\right), \qquad \varphi\mathbf{T}\psi \triangleq \neg\left(\neg\varphi\mathbf{S}\neg\psi\right). \end{aligned} \tag{3}$$

Temporal operators like **X**, **U**, **F**, **G**, and **R** are called *future operators*, whereas **Y**, **Z**, **S**, **O**, **H,** and **T** are called *past operators*. An LTL formula is said to be *pure future* (resp., *pure past*) if it involves no past (resp., future) operators.

Theorem 1 (see [13]). *Each LTL formula has an equivalent pure future expression.*

Theorem 1 tells the fact that past operators do not add any expressive power to LTL formulae. Nevertheless, with these, we can give a much more succinct description in defining specifications.

Given an LTL formula φ, we denote by $\mathrm{sub}(\varphi)$ the set constituted with subformulae of φ. Particularly, we, respectively, denote by $\mathrm{sub}_{\mathbf{U}}(\varphi)$ and $\mathrm{sub}_{\mathbf{S}}(\varphi)$ the set consisting of "**U**-subformulae" and "**S**-subfomulae" of φ, where an **U**-formula (resp., **S**-formula) is a formula rooted at **U** (resp., **S**). (Note that $\mathbf{F}\varphi$ is also a **U**-formula whereas $\mathbf{G}\varphi$ is not.)

A *model* is a tuple $M = \langle \mathscr{V}, \rho, \theta_0, \mathscr{F}\rangle$, where

(i) $\mathscr{V} \subseteq \mathscr{P}$ is a finite set of atomic propositions,

(ii) $\rho \in \mathbb{B}(\mathscr{V} \cup \mathscr{V}')$ is the *transition relation*,

(iii) $\theta_0 \in \mathbb{B}(\mathscr{V})$ is the *initial condition*,

(iv) $\mathscr{F} \subseteq \mathbb{B}(\mathscr{V})$ is a set of *fairness constraints*.

A *derived linear structure* of M is an infinite word $\pi \in (2^{\mathscr{V}})^{\omega}$, such that

(1) $\pi(0) \Vdash \theta_0$;

(2) $(\pi(i), \pi(i+1)) \Vdash \rho$ for each $i \prec \omega$;

(3) for each $\varphi \in \mathscr{F}$, there are infinitely many i's having $\pi(i) \Vdash \varphi$.

We denote by $\mathbf{L}(M)$ the set of derived linear structures of M and call it the *language* of M.

For a model M and an LTL formula φ, we denote as $M \vDash \varphi$ if $\pi \vDash \varphi$ for each $\pi \in \mathbf{L}(M)$. Meanwhile, we define

$$\mathbf{CE}(\varphi, M) \triangleq \{\pi \in \mathbf{L}(M) \mid \pi \nvDash \varphi\} \tag{4}$$

and call it the *counterexample set* of φ w.r.t. M.

3. Counterexample-Preserving Reduction

We describe the CePRe technique in this section, but first of all, let us fix the components of the model and just let M be $\langle \mathscr{V}, \rho, \theta_0, \mathscr{F}\rangle$ in the rest of this section. For M, we are particularly concerned about formulae having the same counterexample set—we say that φ and ψ are *interreduceable* w.r.t. M if and only if $\mathbf{CE}(\varphi, M) = \mathbf{CE}(\psi, M)$, denoted as $\varphi \approx_M \psi$. Hence, $\varphi \approx_M \psi$ implies that $M \vDash \varphi \Leftrightarrow M \vDash \psi$.

Table 1: Reduction rules about model components.

$\theta_0 \vdash \theta \rhd \theta \approx \top$ (Init)	$\rho \vdash \theta \rhd \mathbf{G}\theta \approx \top$ (Trans)
$\theta \in \mathscr{F} \rhd \mathbf{GF}\theta \approx \top$ (Fair)	$\theta_0 \vdash \theta; \rho \vdash \theta \to \theta' \rhd \mathbf{G}\theta \approx \top$ (Ind)

Table 2: Reduction rules of (Conj) and (Disj).

$(\mathbf{P}^{w}\varphi \wedge \mathbf{P}^{s}\varphi) \approx \mathbf{P}^{s}\varphi$ (Conj)	$(\mathbf{P}^{w}\varphi \vee \mathbf{P}^{s}\varphi) \approx \mathbf{P}^{w}\varphi$ (Disj)

Table 3: Reduction rules for formulae involving nested pure future operators.

$\mathbf{F}(\varphi\mathbf{U}\psi) \approx \mathbf{F}\psi$ (FU)	$\varphi\mathbf{U}(\mathbf{F}\psi) \approx \mathbf{F}\psi$ ($\mathrm{U_F}$)
$\mathbf{FF}\varphi \approx \mathbf{F}\varphi$ (FF)	$\mathbf{GFG}\varphi \approx \mathbf{FG}\varphi$ (GFG)

The central part of CePRe is a series of *reduction rules* being of the form

$$\mathrm{Cond} \rhd \varphi \approx_M \psi \ (\mathrm{NAME}), \tag{5}$$

where "Cond" is called the *additional condition*.

Though the relation $\approx_M$ is actually symmetric, we always write the reduced formula on the righthand of the "$\approx$" sign in a reduction rule. Since the model M is fixed, in this section, we omit it from the subscript. In addition, if the additional condition trivially holds, we will discard this part and directly write the rule as $\varphi \approx \psi$, and in this case we say that this rule is "*model-independent*"; otherwise, we say that the underlying reduction rule is "*model-dependent.*"

3.1. The Basic Reduction Rules. In this section, we define some basic CePRe rules. First of all, we have some elementary reduction rules relating to model components, as depicted in Table 1. For the rules (Init), (Ind), and (Trans), the notation "$\vdash$" occurring in the condition part stands for the "*inferring*" relation in propositional logic ($\rho \vdash \theta$ if and only if $\rho \wedge \neg\theta$ is unsatisfiable), and we here require that $\theta, \theta_1, \theta_2 \in \mathbb{B}(\mathscr{V})$.

Subsequently, let us define a partial order "$\sqsubseteq$" over unary temporal operators (and their combinations) as follows:

$$\begin{aligned} &\mathbf{F} \sqsubseteq \mathbf{GF} \sqsubseteq \mathbf{FG} \sqsubseteq \mathbf{G} \\ &\mathbf{F} \sqsubseteq \mathbf{X}^{i} \sqsubseteq \mathbf{G} \quad (i \prec \omega) \\ &\mathbf{O} \sqsubseteq \mathbf{HO} \sqsubseteq \mathbf{OH} \sqsubseteq \mathbf{H}, \end{aligned} \tag{6}$$

where $\mathbf{X}^{0}\varphi \triangleq \varphi$ and $\mathbf{X}^{i+1}\varphi \triangleq \mathbf{X}(\mathbf{X}^{i}\varphi)$.

Assume that $\mathbf{P}^{w}, \mathbf{P}^{s} \in \{\mathbf{F}, \mathbf{FG}, \mathbf{GF}, \mathbf{G}, \mathbf{O}, \mathbf{HO}, \mathbf{OH}, \mathbf{H}\} \cup \{\mathbf{X}^{i} \mid i \prec \omega\}$ and $\mathbf{P}^{w} \sqsubseteq \mathbf{P}^{s}$; then we have two model-indenpendent rules, as depicted in Table 2. Though these rules seem to be trivial, they are useful in doing combinational reductions (see the example given in Section 3.3).

Table 3 provides some reduction rules that can be used to simplify nested temporal operators.

Since we always stand at the starting point when doing model checking (i.e., the goal is to check if $\pi, 0 \vDash \varphi$ for each $\pi \in \mathbf{L}(M)$), we can sometimes "erase" the outermost past operators, as depicted in Table 4.

Table 5 introduces a series of rules handling formulae involving adjacent past and future temporal operators.

Table 4: Reduction rules for formulae involving (outermost) past operators.

$\mathbf{Y}\varphi \approx \bot$ (Y)	$\mathbf{O}\varphi \approx \varphi$ (O)	$\varphi\mathbf{S}\psi \approx \varphi$ (S)

Table 5: Reduction rules for formulae involving adjacent past and future operators.

$\mathbf{XY}\varphi \approx \varphi$ (XY)	$\mathbf{FH}\varphi \approx \mathbf{H}\varphi$ (FH)
$\mathbf{FO}\varphi \approx \mathbf{F}\varphi \vee \mathbf{O}\varphi$ (FO)	$\mathbf{F}(\varphi\mathbf{S}\psi) \approx \mathbf{F}\psi \vee \varphi\mathbf{S}\psi$ (FS)

We let $\theta_1, \theta_2, \ldots$ range over $\mathbb{B}(\mathscr{V})$ and let $\varphi_1, \varphi_2, \ldots$ be arbitrary LTL formulae, and then we have some model-dependent rules. The first group of such rules is listed in Table 6. Table 7 provides another set of model-dependent reduction rules, and these rules are mainly concerned with LTL formulae involving adjacent **U**-operators. Lastly, Table 8 provides some reduction rules that can be used to simplify formulae with mixed usage of **U** and **R**.

3.2. Principles for Rule Generation. Actually, we may acquire more CePRe rules by applying some *principles* to the existing rules. In this section, we introduce four major principles in our framework.

3.2.1. The Inversion Principle. We say that the temporal opertors "**U** and **S,**" "**R** and **T,**" "**G** and **H,**" and "**F** and **O**" are pairwise *inverse operators* (The inverse operator of **X** may be **Z** or **Y**, which is judged from the context. Fortunately, in this paper, we need not consider this case). Then, for CePRe rules list in Table 3 and Table 6–Table 8, we may apply the following principle:

$$\frac{\rho \vdash \theta \rhd \varphi \approx \psi}{\rho \vdash \widetilde{\theta} \rhd \mathsf{inv}(\varphi) \approx \mathsf{inv}(\psi)}, \tag{7}$$

namely, *Inversion Principle*. In detail:

(i) if $\theta \in \mathbb{B}(\mathscr{V})$ then $\widetilde{\theta} = \theta$; if $\theta \in \mathbb{B}(\mathscr{V} \cup \mathscr{V}')$, then $\widetilde{\theta}$ is obtained from θ by switching the primed and unprimed propositions;

(ii) $\mathsf{inv}(\varphi)$ and $\mathsf{inv}(\psi)$ are obtained from φ and ψ by swithing the temporal operators with their inverse operators, respectively.

For example, one can apply this principle to (U) and (R), and then the following two rules

$$\begin{aligned} &\rho \vdash \theta_1 \vee \theta_2 \rhd \theta_1 \mathbf{S}\theta_2 \approx \mathbf{O}\theta_2 \quad \text{(S)} \\ &\rho \vdash \theta_2' \longrightarrow \theta_1' \vee \theta_2 \rhd \theta_1 \mathbf{T}\theta_2 \approx \theta_2 \quad \text{(T)} \end{aligned} \tag{8}$$

are immediately obtained.

3.2.2. The Duality Principle. We say that "$\top$ and $\bot$," "$\wedge$ and $\vee$," "**F** and **G,**" "**O** and **H,**" "**Y** and **Z,**" "**X** and **X** itself," "**U** and **R,**" and "**T** and **S**" are pairwise the *dual operators*. Then, the *duality principle* could be described as follows:

$$\frac{\mathrm{Cond} \rhd \varphi \approx \psi}{\overline{\mathrm{Cond}} \rhd \mathsf{dual}(\varphi) \approx \mathsf{dual}(\psi)}. \tag{9}$$

Table 6: Reduction rules of (U) and (R).

$\rho \vdash \theta_1 \vee \theta_2 \rhd \theta_1 \mathbf{U}\theta_2 \approx \mathbf{F}\theta_2$ (U)
$\rho \vdash \theta_2 \rightarrow \theta_1 \vee \theta_2' \rhd \theta_1 \mathbf{R}\theta_2 \approx \theta_2$ (R)

Table 7: Reduction rules for formulae involving adjacent **U** operators.

$\rho \vdash \theta_1 \rightarrow \theta_2 \rhd \theta_1 \mathbf{U}\theta_2 \approx \theta_2$	$(\mathrm{U}[1 \rightarrow 2])$
$\rho \vdash \theta_1 \rightarrow \theta_3 \rhd (\theta_1 \mathbf{U}\varphi_2)\mathbf{U}\theta_3 \approx \varphi_2 \mathbf{U}\theta_3$	$(\mathrm{U}^{\mathrm{U}}[1 \rightarrow 3])$
$\rho \vdash \theta_2 \rightarrow \theta_3 \rhd (\varphi_1 \mathbf{U}\theta_2)\mathbf{U}\theta_3 \approx \theta_3 \vee (\varphi_1 \mathbf{U}\theta_2)$	$(\mathrm{U}^{\mathrm{U}}[2 \rightarrow 3])$
$\rho \vdash \theta_3 \rightarrow \theta_2 \rhd (\varphi_1 \mathbf{U}\theta_2)\mathbf{U}\theta_3 \approx (\varphi_1 \vee \theta_2)\mathbf{U}\theta_3$	$(\mathrm{U}^{\mathrm{U}}[3 \rightarrow 2])$
$\rho \vdash \theta_2 \rightarrow \theta_3' \rhd (\varphi_1 \mathbf{U}\theta_2)\mathbf{U}\theta_3 \approx (\varphi_1 \vee \theta_2)\mathbf{U}\theta_3$	$(\mathrm{U}^{\mathrm{U}}[2 \rightarrow 3'])$
$\rho \vdash \neg\theta_2 \rightarrow \theta_3 \rhd (\varphi_1 \mathbf{U}\theta_2)\mathbf{U}\theta_3 \approx \mathbf{F}\theta_3$	$(\mathrm{U}^{\mathrm{U}}[\neg 2 \rightarrow 3])$
$\rho \vdash \theta_1 \rightarrow \theta_2 \rhd \theta_1 \mathbf{U}(\theta_2 \mathbf{U}\varphi_3) \approx \theta_2 \mathbf{U}\varphi_3$	$(\mathrm{U}_{\mathrm{U}}[1 \rightarrow 2])$
$\rho \vdash \theta_1 \rightarrow \theta_3 \rhd \theta_1 \mathbf{U}(\varphi_2 \mathbf{U}\theta_3) \approx \varphi_2 \mathbf{U}\theta_3$	$(\mathrm{U}_{\mathrm{U}}[1 \rightarrow 3])$
$\rho \vdash \theta_2 \rightarrow \theta_1 \rhd \theta_1 \mathbf{U}(\theta_2 \mathbf{U}\varphi_3) \approx \theta_1 \mathbf{U}\varphi_3$	$(\mathrm{U}_{\mathrm{U}}[2 \rightarrow 1])$

Table 8: Reduction rules for formulae involving adjacent **U** and **R** operators.

$\rho \vdash \theta_1 \rightarrow \theta_3 \rhd (\theta_1 \mathbf{R}\varphi_2)\mathbf{U}\theta_3 \approx ((\theta_1 \mathbf{R}\varphi_2) \vee \theta_3) \wedge \mathbf{F}\theta_3$	$(\mathrm{U}^{\mathrm{R}}[1 \rightarrow 3])$
$\rho \vdash \neg\theta_1 \rightarrow \theta_3 \rhd (\theta_1 \mathbf{R}\varphi_2)\mathbf{U}\theta_3 \approx \varphi_2 \mathbf{U}\theta_3$	$(\mathrm{U}^{\mathrm{R}}[\neg 1 \rightarrow 3])$
$\rho \vdash \theta_1 \rightarrow \theta_3 \rhd \theta_1 \mathbf{U}(\varphi_2 \mathbf{R}\theta_3) \approx \varphi_2 \mathbf{R}\theta_3$	$(\mathrm{U}_{\mathrm{R}}[1 \rightarrow 3])$

We require that the condition "Cond" should be either trivial (in this case, the rule is a model-independent one) or of the form $\rho \vdash \theta_1 \rightarrow \theta_2$, and in the latter case we have $\overline{\mathrm{Cond}} = \rho \vdash \theta_2 \rightarrow \theta_1$. The formulae $\mathsf{dual}(\varphi)$ and $\mathsf{dual}(\psi)$ are obtained from φ and ψ by switching the operators with their dual.

For example, apply the duality principle to the rules (FU), $(\mathrm{U}_{\mathrm{F}})$, (FF), and (GFG), one may get the following new rules:

$$\begin{aligned} &\mathbf{G}(\varphi\mathbf{R}\psi) \approx \mathbf{G}\psi \quad \text{(GR)}, \qquad \varphi\mathbf{R}(\mathbf{G}\psi) \approx \mathbf{G}\psi \quad (\mathrm{R_G}), \\ &\mathbf{GG}\varphi \approx \mathbf{G}\varphi \quad \text{(GG)}, \qquad \mathbf{FGF}\varphi \approx \mathbf{GF}\varphi \quad \text{(FGF)}. \end{aligned} \tag{10}$$

Note that when applying the duality principle to model-dependent rules, besides switching the operators, we also need to exchange the antecedent and subsequent on the righthand of $\vdash$ in the condition part, in the case that the condition is of the form $\rho \vdash \theta_1 \rightarrow \theta_2$. As an example, we may obtain the reduction rule

$$\begin{aligned} \rho \vdash \theta_3 \longrightarrow \theta_2 \rhd (\varphi_1 \mathbf{R}\theta_2)\mathbf{R}\theta_3 \approx \theta_3 \wedge (\varphi_1 \mathbf{R}\theta_2) \\ \left(\mathrm{R}^{\mathrm{R}}[3 \longrightarrow 2]\right), \end{aligned} \tag{11}$$

by applying the duality principle to $(\mathrm{U}^{\mathrm{U}}[2 \rightarrow 3])$.

3.2.3. The Composition Principle. The *composition principle* can be formally described as follows:

$$\frac{\begin{array}{c}\mathrm{Cond}_1 \rhd \varphi_1 \approx \psi_1 \\ \mathrm{Cond}_2 \rhd \varphi_2 \approx \psi_2\end{array}}{\mathrm{Cond}_1 \;\&\; \mathrm{Cond}_2 \rhd \varphi_1 \approx \psi_{1\psi_2}^{\varphi_2}}; \tag{12}$$

that is, if φ_i and ψ_i are interreduceable under the conditin Cond_i $(i = 1, 2)$, then φ_1 and $\psi_{1\psi_2}^{\varphi_2}$ are also interreduceable. In using this principle, we have the following constraints.

Input: The original specification φ.
Output: The reduced specification.
(1) **let** $\Gamma := \emptyset$; /* Γ memorizes the sub-formulae with infeasible condition */
(2) **let** $\Delta := \{\psi \in (\text{sub}(\varphi) \backslash \Gamma)$ such that ψ matches some reduction rule(s)$\}$;
(3) **foreach** $\psi_1, \psi_2 \in \Delta$ s.t. $\psi_1 \neq \psi_2$ **do**
(4) **if** $\psi_1 \in \text{sub}(\psi_2)$ **then**
(5) $\Delta := \Delta \setminus \{\psi_1\}$; /* that is, we only proceed "max" subformulae */
(6) **end**
(7) **end**
(8) **if** $\Delta = \emptyset$ **then**
(9) **return** φ;
(10) **end**
(11) **foreach** $\psi \in \Delta$ **do**
(12) **let** $\Theta :=$ the set of rules that can be applied to ψ;
(13) /* note that we have $|\Theta| \leq 5$ for each ψ */
(14) **while** $\Theta \neq \emptyset$ **do**
(15) **choose** $R := (\text{Cond} \rhd \psi \approx \eta)$ **in** Θ;
(16) **if** Cond *is stated* **then**
(17) $\varphi := \varphi^{\psi}_{\eta}$; /* φ^{ψ}_{η} is obtained from φ by replacing ψ with η */
(18) **break**;
(19) **end**
(20) $\Theta := \Theta \setminus \{R\}$;
(21) **end**
(22) $\Delta := \Delta \setminus \{\psi\}$;
(23) **if** $\Theta = \emptyset$ **then**
(24) $\Gamma := \Gamma \cup \{\psi\}$; /* ψ would be excluded in the next iteration */
(25) **end**
(26) **end**
(27) **goto** 2;

Algorithm 1: The "max-match" rule-selection strategy.

(1) Cond_2 must be θ_0-free; that is, it must be irrelevant to the initial condition.

(2) If the second rule is one of these listed in Table 4 (or the inversion/duality version), then we require that the (designated) occurrences of φ_2 in ψ_1 must be *temporally outermost*; that is, φ_2 does not appear in the scope of any temporal operators.

(3) Since $\eta \approx \eta$ trivially holds, we have the following special case of the composition principle:

$$\frac{\text{Cond} \rhd \varphi \approx \psi}{\text{Cond} \rhd \eta \approx \eta^{\varphi}_{\psi}}, \tag{13}$$

and call it the local application of (the above) CePRe rule.

To explaint the reason why we need to impose the second constraint, just consider the following fact: we have $\mathbf{Y}\varphi \approx \bot$ according to (Y); yet this does not imply that $\mathbf{FY}\varphi \approx \mathbf{F}\bot$ holds.

3.2.4. The Decomposition Principle. The last principle, which could be corporately used in *modular model checking* [14], is described as follows:

$$\frac{\begin{array}{c}\text{Cond}_1 \rhd \varphi_1 \approx_{M_1} \psi_1 \\ \text{Cond}_2 \rhd \varphi_2 \approx_{M_2} \psi_2\end{array}}{\text{Cond}_1 \ \& \ \text{Cond}_2 \rhd \varphi_1 \approx_{M_1 \| M_2} \psi_1{}^{\varphi_2}_{\psi_2}}, \tag{14}$$

where all constraints imposed to the *composition principle* are still required. $M_1 \parallel M_2$ is the *synchronized composition* of M_1 and M_2; that is, $M_1 \parallel M_2 = \langle \mathscr{V}_1 \cup \mathscr{V}_2, \rho_1 \wedge \rho_2, \theta_{0,1} \wedge \theta_{0,2}, \mathscr{F}_1 \cup \mathscr{F}_2 \rangle$ if $M_i = \langle \mathscr{V}_i, \rho_i, \theta_{0,i}, \mathscr{F}_i \rangle$.

3.3. Reduction Strategy. We show the usage of CePRe reduction rules by illustrating the reduction process of $M \models (\theta_1 \mathbf{U} \theta_2)\mathbf{U}\theta_3$; see Algorithm 1.

(1) We may first try with the rule $(\mathrm{U}^{\mathrm{U}}[1 \rightarrow 3])$ by inquiring the SAT-solver if $\rho \vdash \theta_1 \rightarrow \theta_3$ holds.

(2) If the SAT-solver returns "unsatisfiable" with the input $\rho \wedge \theta_1 \wedge \neg\theta_3$, then it implies that the additional condition is stated, and we may replace the specification with $\theta_2 \mathbf{U} \theta_3$.

(3) Otherwise, we will try with another reduction rule, such as $(\mathrm{U}^{\mathrm{U}}[2 \rightarrow 3])$.

Compositional use of reduction rules may lead to a more aggressive reduction. For example,

(1) for the task of model checking $M \models \mathbf{FO}p$, we may firstly change the goal as $M \models \mathbf{F}p \vee \mathbf{O}p$, according to the rule (FO);

(2) now, the subformula $\mathbf{O}p$ is a temporally outermost one; hence we may make a local application of (O), and then the goal becomes $M \models \mathbf{F}p \vee p$;

(3) finally, we may change the model checking problem into $M \vDash \mathbf{F}p$ via the rule (Dis$_J$).

In the real implementation, we perform a "*max-match*" rule-selection strategy, as depicted in Algorithm 1. Note that in this algorithm, only

(1) basic rules,

(2) rules obtained from the inversion principle or the duality principle

could be directly used (maybe in a local manner). This guarantees the finiteness for rule selection.

In line 15, a rule having simpler condition always has the higher precedence to be chosen. Hence, a model-independent rule always has a higher priority than a model-dependent one.

Lastly, we can see that the reduction can be accomplished within $\mathcal{O}(|\varphi|)$ iterations.

4. Performance Analysis of CePRe

We now try to answer the question "why we can gain a better performance during verification if CePRe is conducted first." To give a rigorous explanation, we briefly revisit the implementation of some symbolic model checking algorithms, and we are mainly concerned about the following techniques:

(1) the BDD-based model checking technique,

(2) the bounded model checking technique (BMC), for both syntactic encoding and semantic encoding approaches,

(3) the property directed reachability (alternatively, IC3) algorithm.

4.1. Analysis on BDD-Based Model Checking. The core procedure of BDD-based LTL symbolic model checking algorithm is to construct a *tableau* for the (negated) property. In the following, we refer the tableau of $\neg\varphi$ as $T_{\neg\varphi}$, and we would give an analysis on its major components affecting the cost of model checking.

State Space. The state space of $T_{\neg\varphi}$ consists of subsets of $el(\varphi)$, and the set $el(\varphi)$ can be inductively computed as follows:

(i) $el(\top) = el(\bot) = \emptyset$;

(ii) $el(p) = \{p\}$ if $p \in \mathscr{P}$;

(iii) $el(\varphi_1 \rightarrow \varphi_2) = el(\varphi_1) \cup el(\varphi_2)$;

(iv) $el(\mathbf{X}\psi) = \{\mathbf{X}\psi\} \cup el(\psi)$, and $el(\mathbf{Y}\psi) = \{\mathbf{Y}\psi\} \cup el(\psi)$;

(v) $el(\varphi_1 \mathbf{U} \varphi_2) = el(\varphi_1) \cup el(\varphi_2) \cup \{\mathbf{X}(\varphi_1 \mathbf{U} \varphi_2)\}$ and $el(\varphi_1 \mathbf{S} \varphi_2) = el(\varphi_1) \cup el(\varphi_2) \cup \{\mathbf{Y}(\varphi_1 \mathbf{S} \varphi_2)\}$.

We can see from the definition that $el(\varphi) = el(\neg\varphi)$ holds. With symbolic representation, each formula $\psi \in el(\varphi)$ corresponds to a proposition in building the tableau. Moreover, if $\psi \in \mathscr{P}$, then no new proposition needs to be introduced (since it has already been introduced in building the symbolic representation of M); otherwise, a fresh proposition p_ψ is required. Hence the total number of newly introduced propositions equals $|el(\varphi) \setminus \mathscr{P}|$. From an induction over formula's structure, we have the following claim.

Proposition 2. *$|el(\varphi) \setminus \mathscr{P}|$ equals the number of temporal operators in φ.*

Transitions. The transition relation of $T_{\neg\varphi}$ is a conjunction of a set of constraints, and each constraint is either of the form $p_{\mathbf{X}\psi} \leftrightarrow (\sigma(\psi))'$ or $p'_{\mathbf{Y}\eta} \leftrightarrow \sigma(\eta)$, where $\mathbf{X}\psi, \mathbf{Y}\eta \in el(\varphi)$, and the function σ can be inductively defined as follows:

(i) $\sigma(\bot) = \bot$ and $\sigma(\top) = \top$;

(ii) $\sigma(p) = p$ for each $p \in \mathscr{P}$;

(iii) $\sigma(\psi_1 \rightarrow \psi_2) = \sigma(\psi_1) \rightarrow \sigma(\psi_2)$;

(iv) $\sigma(\mathbf{X}\psi_1) = p_{\mathbf{X}\psi_1}$ and $\sigma(\mathbf{Y}\psi_2) = p_{\mathbf{Y}\psi_2}$;

(v) $\sigma(\psi_1 \mathbf{U} \psi_2) = \sigma(\psi_2) \vee \sigma(\psi_1) \wedge p_{\mathbf{X}(\psi_1 \mathbf{U} \psi_2)}$ and $\sigma(\psi_1 \mathbf{S} \psi_2) = \sigma(\psi_2) \vee \sigma(\psi_1) \wedge p_{\mathbf{Y}(\psi_1 \mathbf{U} \psi_2)}$.

According to the definition of *el*, we can see that each $\psi \in \mathrm{sub}(\varphi)$ rooted at a future (resp., past) temporal operator exactly produces one formula $\mathbf{X}\eta$ (resp., $\mathbf{Y}\eta$) in $el(\varphi)$, and hence a new proposition $p_{\mathbf{X}\eta}$ (resp., $p_{\mathbf{Y}\eta}$) would be introduced. Subsequently, each such $p_{\mathbf{X}\eta}$ (resp., $p_{\mathbf{Y}\eta}$) adds exactly one constraint to the transition relation. Hence, we have the following claim.

Proposition 3. *The number of constraints in the transition relation of $T_{\neg\varphi}$ equals the number of temporal operators occurring in φ (alternatively, $|el(\varphi) \setminus \mathscr{P}|$).*

Fairness Constraints. According to the tableau construction, each $\psi \in \mathrm{sub}_{\mathbf{U}}(\neg\varphi)$ would impose a fairness constraint to $T_{\neg\varphi}$. Hence, the number of fairness constraints equals $|\mathrm{sub}_{\mathbf{U}}(\neg\varphi)|$.

With a case-by-case checking, we can show the following theorem.

Theorem 4. *Let "Cond $\triangleright$ $\varphi \approx \psi$" be a reduction rule; then we have $|el(\psi) \setminus \mathscr{P}| \leq |el(\varphi) \setminus \mathscr{P}|$ and $|sub_{\mathbf{U}}(\psi)| \leq |sub_{\mathbf{U}}(\neg\varphi)|$.*

4.2. Analysis on Bounded Model Checking. In contrast, the cost of BMC is quite sensitive to the encoding approach. In a broad sense, we can categorize the encoding approaches into two fashions.

Syntactic Encodings. Such kind of encodings are inductively produced w.r.t. the formula's structure. The very original one is presented in [7], and this is improved in [15] by observing some properties of that encoding. In [16, 17], a linear incremental syntactic encoding is suggested, and see [18] for the translation for ECTL*.

Semantic Encodings. In [19], an alternative BMC technique is provided: it mimics the tableau-based model checking process, but it expresses the fair-path detection upon the product model with Boolean formula. (In [20], a "fixpoint"-based encoding is proposed, and it can also be subsumed to semantic encodings.)

For the semantic encodings, the reason that we can benefit from CePRe is exactly the same as that for BDD-based

approach. Because, the encoding is a conjunction of a k-step unrolling of M and a k-step unrolling of $T_{\neg\varphi}$ (an unrolling is either a partial derived linear structure or one ending with a lasso). The former is usually in a fixed pattern, and for the latter, we need $k\times|el(\varphi)\setminus\mathscr{P}|$ new propositions, and the sizes of Boolean formulae w.r.t the transition and fairness constraints (note that the part w.r.t. fairness constraints can be linearized) are, respectively, $\mathcal{O}(k\times|el(\varphi)\setminus\mathscr{P}|)$ and $\mathcal{O}(k^2\times|\mathrm{sub}_{\mathbf{U}}(\neg\varphi)|)$.

For a syntactic BMC encoding, one needs to generate a Boolean formula of the form $E_M^k\wedge E_{\neg\varphi}^k$, where E_M^k is the unrolling of M with k steps, and $E_{\neg\varphi}^k$ describes that such k-step unrolling would cause a violation of φ. In general, E_M^k is almost the same in all kinds of syntactic encodings, and the key factor affecting the cost lies in $E_{\neg\varphi}^k$.

Given a subformula ψ of φ, if we use $||E_\psi^k||$ to denote the maximum length of the Boolean formula describing that ψ is initially satisfied upon a k-step unrolling, then it can be inductively computed as follows:

(i) $||E_\bot^k||=||E_\top^k||=0$; (this is just for the case when $\bot$ or $\top$ appears as a subformula in the specification and hence can be optimized; otherwise, we have $||E_\bot^k||=||E_\top^k||=1$);

(ii) $||E_p^k||=1$ for each $p\in\mathscr{P}$;

(iii) $||E_{\varphi_1\to\varphi_2}^k||=||E_{\varphi_1}^k||+||E_{\varphi_2}^k||+1$;

(iv) $||E_{\mathbf{X}\psi}^k||=||E_{\mathbf{Y}\psi}^k||=||E_\psi^k||$;

(v) $||E_{\varphi_1\mathbf{U}\varphi_2}^k||=||E_{\varphi_1\mathbf{S}\varphi_2}^k||=L(k)\times||E_{\varphi_1}^k||+k\times||E_{\varphi_2}^k||$. (Note that this case does not imply that further blowup would be caused with deeper nesting of temporal operators. For example, in [17], by introducing fresh propositions and reusing, it still leads to a linear encoding for the whole formula.)

Here, $L(k)$ is some polynomial about k, related to the encoding approach. For example, with the technique proposed in [7, 15], we have $L(k)\in\mathcal{O}(k^2)$, whereas $L(k)\in\mathcal{O}(k)$ in [16, 17]. This partly explains the reason that we tend to change temporal nestifications with Boolean combinations, as done in $(\mathrm{U}^{\mathrm{U}}[3\to 2])$ and so forth.

Another feature affecting the cost is the scale of propositions required for the encoding. If we denote by $\mathrm{var}_k(\varphi)$ the set of additional propositions which only takes part in the encoding of $E_{\neg\varphi}^k$, then we have the following conclusions.

(i) For the techniques proposed in [7, 15], we have $\mathrm{var}_k(\varphi)=0$. i.o.w.; all propositions required in encoding $E_{\neg}^k\varphi$ can be shared with those for E_M^k.

(ii) In term of the encoding presented in [17], we need to add $\mathcal{O}(k)$ new propositions to $\mathrm{var}_k(\varphi)$ for each **U**-subformula and for each **S**-subformula.

Theorem 5. *Let "$Cond\rhd\varphi\approx\psi$" be a reduction rule; then we have $||E_\psi^k||\le||E_\varphi^k||$ and $|var_k(\psi)|\le|var_k(\varphi)|$ in syntactic encodings.*

4.3. Analysis on Property Directed Reachability Algorithm. Property directed reachability (PDR or IC3) is probably the most significant breakthrough in SAT based model checking. The aim of PDR is to show that the system is *safe* w.r.t. the property θ; that is, each reachable state satisfies the boolean constraint θ.

The process of PDR retains a sequence $\Gamma_0,\ldots,\Gamma_k$ of clause sets fulfilling the following.

(1) $\Gamma_0=\{\theta_0\}$; and for each $i\leqslant k$ we have

(2) $\vdash\wedge\Gamma_i\to\wedge\Gamma_{i+1}$;

(3) $\vdash\wedge\Gamma_i\wedge\rho\to\wedge\Gamma'_{i+1}$;

(4) $\vdash\wedge\Gamma_i\to\theta$.

This process stops and reports an affirmative answer if there exists some $i>0$ having $\Gamma_i=\Gamma_{i-1}$; and it reports failure when some reachable "unsafe" state is detected. Therefore, it intends to find a (intermediate) reachable-closure enclosed (or *inductive set*) with θ, as described in Algorithm 2.

This algorithm employees two important subroutines—the strengthen process (Line 7) and the propagate process (Line 10):

(i) The process strengthen aims to disprove the reachability of the point $\mathscr{U}$ which violates θ. If it succeeds, some new constraints would be added to some specific clause sets; if this process fails, then it implies that we have a feasible path which leads to an "unsafe" state. In [21], an effective strengthen algorithm is provided, and it results in a linear complexity in time w.r.t. the size of variable set.

(ii) The process propagate is used to "compact" the clause sets and to accelerate the termination. Its purpose is to "push out" the clauses in a set as possible as they can. Suppose that there is a clause $\eta\in\Gamma_i$, in the case that $\vdash\wedge\Gamma_i\wedge\rho\to\eta'$ holds; since we have $\vdash\wedge\Gamma_i\wedge\rho\to\wedge\Gamma'_{i+1}$, it implies that $\vdash\wedge\Gamma_i\wedge\rho\to\wedge\Gamma'_{i+1}\wedge\eta'$ also holds. In such situation, the clause η could be definitely propagated to Γ_{i+1}.

Since the PDR algorithm could not directly handle general temporal properties, we need to first translate the LTL model checking problem to PDR solving, via the following steps.

(1) Build the tableau $T_{\neg\varphi}$ from the LTL formula φ, as described in Section 4.1.

(2) Thus, the problem is boiled down to a fair-circle detection problem on the production of M and $T_{\neg\varphi}$.

(3) This problem could be further translated to $n+1$ PRD-solving problems, where n is the number of fairness constraints in the production. Interested readers may refer to [22] for the translation details.

It could be seen that the features affecting the overhead of PDR-solving boiled from LTL input are the following:

(1) the scale of product system, which is determined by the number of variables in the tableau and model,

Input: The components $\mathcal{V}$, ρ, θ_0 of the model M; a safety property θ.
Output: The affirmative answer if M is safe w.r.t. θ; otherwise, a counterexample witnessing that $\neg\theta$ is reachable.

```
(1) let k := 0;
(2) let Γ_0 = {θ_0};
(3) let Q := ∅; /* Q is a priority queue */
(4) repeat
(5)     while there exists 𝒰 ⊆ 𝒱 s.t. 𝒰 ⊩ Γ_k ∧ ¬θ do
(6)         add (𝒰, k) to the head of Q;
(7)         if strengthen({Γ_i}_{i≤k}, Q, 𝒱) fails then
(8)             return counterexample extracted from Q;
(9)         end
(10)        propagate({Γ_i}_{i≤k});
(11)        if there exists some j ≤ k s.t. Γ_j = Γ_(j−1) then
(12)            return "M is safe from θ";
(13)        end
(14)        let Γ_{k+1} := ∅;
(15)        k := k + 1;
(16)    end
(17) until 1 ≠ 1;
```

Algorithm 2: Framework of the PDR algorithm.

(2) the size (or number of clauses) of the transition relation, which is a summation of these of the tableau and the model,

(3) the number of fairness constrains in the product, which is also the summation of constraint numbers of the model and the tableau.

For two LTL formulae φ and ψ such that $\varphi \approx_M \psi$, since the model is the same, the "variable number," "number of clauses in the transition relation," and "the number of fairness contraints" are determined by the tableaux $T_{\neg\varphi}$ and $T_{\neg\psi}$. Precisely like the analysis we have done in Section 4.1, we have the same conclusions shown in Propositions 2 and 3 and Theorem 4, and these would explain why we can also benefit from CePRe when using PDR as underlying checking approach.

5. Experimental Results

We have implemented CePRe as an upfront option in NuSMV (the tool is available on http://sourceforge.net/projects/nusmvwithcepre/), and we have also conducted experiments upon both industrial benchmarks and randomly generated cases in terms of BDD-based model checking, bounded model checking, and PDR-based model checking.

The BMC encoding approach we adopt here is that proposed in [15], which is the current BMC implementation of NuSMV. Since PDR verifiers could not directly take SMV models and/or LTL formulae as input, and they use AIGs (the And-Inverter Graphs) as standard input format, we first use the SMVtoAIG tool [23] to convert SMV-scripts into AIGs and then use iimc [24] to perform the PDR-based verification.

We conduct the experiments under such platform: CPU-Intel Core Duo2 E4500 2.2 GHz, Mem-2 G Bytes, OS-Ubuntu 10.04 Linux, Cudd-v2.4.1.1, Zchaff-v2007.3.12.

5.1. Experiments upon Industrial Benchmarks. The benchmarks we choose in this paper are suggested in [4], and most of them come from real hardware verification.

Table 9 provides the experimental results for BDD-based LTL symbolic model checking. The field time is the executing time totally elapsed, and the field R.S. refers to the number of reachable states. For the experiments "with CePRe," both the overheads of time and space are the summations of preprocessing and model checking. For Table 9, we have the following remarks.

(1) 8 out of 16 specifications could be reduced with CePRe (and these specifications are marked with †).

(2) For the specifications that can be reduced, considerable improvements are made during verification. For example, for the specification Pit.g.ltl, with CePRe, the number of BDD nodes is decreased to 12.5% of that without using CePRe.

(3) When a specification cannot be reduced with CePRe, it spends a very low extra overhead for doing preprocessings.

(4) Something noteworthy we do not provide here is that in the case that a violated LTL specification can be reduced, the newly generated counterexample is usually shorter than that of before. Among 8 specifications that can be reduced, counterexample lengths of Pti.nuv.ltl, Pit.g.ltl, P0.ltl and Seq.ltl are, respectively, shortened to 15, 10, and 194, opposing to the original values 16, 12, and 217. Meanwhile, counterexample lengths of others are kept unchanged.

Table 10 gives the experimental results for BMC-based model checking, and we here give some comments on that.

(1) With NuSMV, we need to preset a max-bound when doing bounded model checking. The column max-bound gives such values—a "star mark" means that

Table 9: Comparative results of BDD-based MC with/without CEPRE.

Model	Spec.	Without CEPRE			With CEPRE		
		BDD--Nodes	R.S.	Time (sec.)	BDD--Nodes	R.S.	Time (sec.)
srg5	Ptimo.ltl†	7946	720	0.024	2751	720	0.016
	Ptignv.ltl†	29704	11460	0.058	5712	2880	0.012
	Pti.g.ltl†	64749	130048	0.048	8119	32768	0.016
abp4	P2false.ltl	99577	559104	0.200	99625	559104	0.202
	P2true.ltl†	61209	904384	0.066	56494	419296	0.064
	Pold.ltl	52301	353536	0.060	52349	353536	0.064
	Ptimo.ltl	78098	219616	0.080	78146	219616	0.088
	Pti.g.ltl	8385	200704	0.060	8433	200704	0.062
dme3	P0.ltl†	889773	35964	5.756	527983	26316	5.096
	P1.ltl	455148	8775	0.460	409432	5505	0.374
dme5	Mdl.ltl†	793942	8.64316e + 06	167.346	814494	3.2097e + 06	114.599
	Wat.ltl†	412867	1.79217e + 07	302.005	967033	1.12567e + 07	286.850
	Ptimo.neg	508036	1.26202e + 06	3.260	508081	1.26202e + 06	3.280
msi_w-trans	Sched.ltl	2275558	7.31055e + 07	6.612	2275655	7.31055e + 07	6.632
	Safety.ltl	1213308	3.6528e + 07	7.568	1213460	3.6528e + 07	7.644
	Seq.ltl †	1921973	3.5946e + 07	93.570	1702585	1.7973e + 07	94.085

Table 10: Experimental results of BMC-based MC with/without CEPRE.

Model	Spec.	Without CEPRE		With CEPRE		Max-bound
		N.O.C.	Time (sec.)	N.O.C.	Time (sec.)	
srg5	Ptimo.ltl†	272567	67.391	1371	0.143	20
	Pti.gnv.ltl†	2101	0.116	299	0.024	6
	Pti.g.ltl†	21	0.016	21	0.016	1
abp4	P2false.ltl	7532	3.972	7532	3.972	17
	P2true.ltl†	12639	8.145	9369	7.753	20*
	Pold.ltl	7499	9.087	7499	9.488	20*
	Ptimo.ltl	6332	2.500	6332	2.512	16
	Pti.g.ltl	11952	0.841	11952	0.976	20*
dme3	P0.ltl†	—	—	35102	524.207	62
	P1.ltl	216	0.036	167	0.048	1
dme5	Mdl.ltl†	90	0.044	90	0.048	0
	Wat.ltl†	367	0.048	274	0.052	1
	Ptimo.neg	367	0.050	277	0.058	1
msi_w-trans	Sched.ltl	14235	1.076	14235	1.078	20*
	Safety.ltl	12439	8.441	12439	8.448	20*
	Seq.ltl†	1907	0.064	81	0.052	3

this bound does not reach the completeness threshold. The field N.O.C. designates the number of clauses generated during model checking.

(2) From Table 10, we can see that without CEPRE the specification Pti.gnv.ltl generates 2101 clauses when the verification stops; in contrast, it only produces 299 clauses if CEPRE is switched on.

(3) Another comparison is for P0.ltl upon dme3: If we do not do any reduction, the SAT solver reports a SEGMENTATION FAULT at Step 35. In contrast, using CEPRE, a counterexample could be found at Step 62.

(4) Since the encoding approach we use is taken from [15], propositions used in the encoding are only determined by the model and the bound; thus

Table 11: Experimental result of PDR-based MC with/without CEPRE.

Model	Spec.	PDR time without CEPRE (sec.)	PDR time with CEPRE (sec.)	Extratime for CEPRE (sec.)
srg5	Ptimo.ltl†	0.117	0.056	<0.001
	Pti.gnv.ltl†	0.084	0.043	<0.001
	Pti.g.ltl †	0.070	0.032	<0.001
abp4	P2false.ltl	3.187	3.185	0.004
	P2true.ltl†	2.996	1.239	0.004
	Pold.ltl	6.937	6.939	<0.001
	Ptimo.ltl	0.910	0.912	<0.001
	Pti.ltl	5.812	5.816	<0.001
dme3	P0.ltl†	9.815	6.086	<0.001
	P1.ltl	0.147	0.142	0.004
dme5	Mdl.ltl†	3.072	1.731	0.004
	Wat.ltl†	4.575	3.043	0.004
	Ptimo.neg	0.073	0.073	0.004
msi_wtrans	Sched.ltl	0.135	0.136	<0.001
	Safety.ltl	0.200	0.201	0.004
	Seq.ltl†	0.091	0.045	0.006

the number of required propositions does not change. For this reason, the corresponding experimental results on proposition numbers are not provided.

Experimental results of PDR-based model checking are shown in Table 11.

(1) For PDR-based model checking, we are mainly concerned about the time-overhead. Since that the PDR algorithm consumes memory evenly during verification, we do not provide the space-overhead here.

(2) Because the CEPRE process is done in an individual phase in the PDR-based verification, we here also list the overhead for doing CEPRE.

(3) We can see that a significant performance improvement could be made if we use CEPRE in PDR-based model checking, and the extra overhead for preprocessing is still relatively negligible.

Note that both model-independent and model-dependent rules contribute to the reductions. For example, for the model srg5 and the specification Pti.g.ltl, the rules (FS) and (S) are applied; meanwhile, for the model msi_wtrans and the specification Seq.ltl, the application of $(U^{U}[\neg 2 \rightarrow 3])$ is invoked.

5.2. Experiments w.r.t. Random Models and Specifications. We have also performed experiments upon randomly generated models and specifications with the tool LBTT [25] and with the methodology suggested in [17].

For BDD-based MC and bounded model checking, we conduct the comparison in the following manner. For each $3 \leq \ell \leq 7$, we randomly generate 40 specifications having length ℓ. Subsequently, for each specification, we generate two models, respectively, for the BDD-based model checking and for BMC. Hence, we totally have 200 specifications and 400 models.

For the BDD-based model checking, we give the comparative results on (1) the scale of BDD-nodes, (2) the number of reachable states, and (3) the time consumed, and the experimental results are, respectively, shown in Figures 1, 2, and 3. For bounded model checking, we have set the maximum bound to 20 and we have compared (1) the number of clauses and (2) the executing time; the results are, respectively, shown in Figures 4 and 5. Each value here we provide is the average of the 40 executions.

For the BDD-based model checking, there are 123 (out of 200) specifications that can be reduced, whereas for bounded model checking, the number of specifications that can be reduced is 118. Note that in this experiment, when CEPRE is switched on, extra overheads (such as time) have also been taken into account.

For the PDR-based model checking, the experiments are conducted as follows. We first generate 50 SMV models with LBTT and then we randomly generate LTL specifications for each model and each designated length. Next, we batchly do CEPRE upon one group of specification copies and then obtain the product models (for each model and each specification). Subsequently, we convert the product models into AIG format with SMVTOAIG.

We here compare the number of verfication obligations that could be accomplished within the preset time bound (i.e., 600 sec.), and the results are shown in Figure 6. From that, we can see that with the increment of the specification's length, the ratio (that could be done with CEPRE to without CEPRE) also monotonically increases.

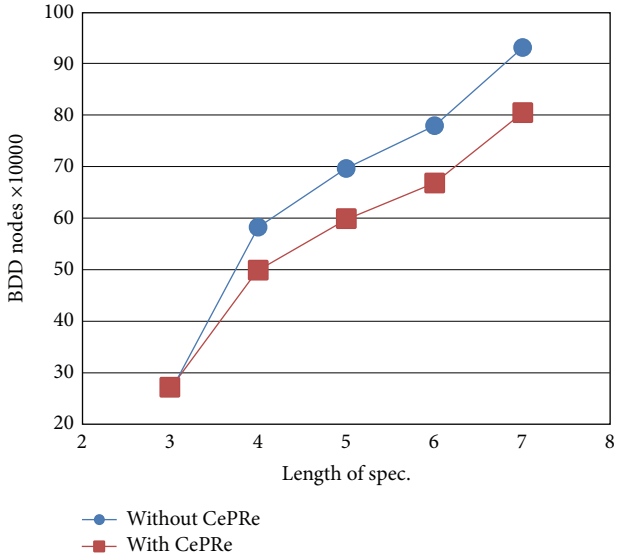

Figure 1: Results on the scale of BDD nodes in random BDD-MC experiments.

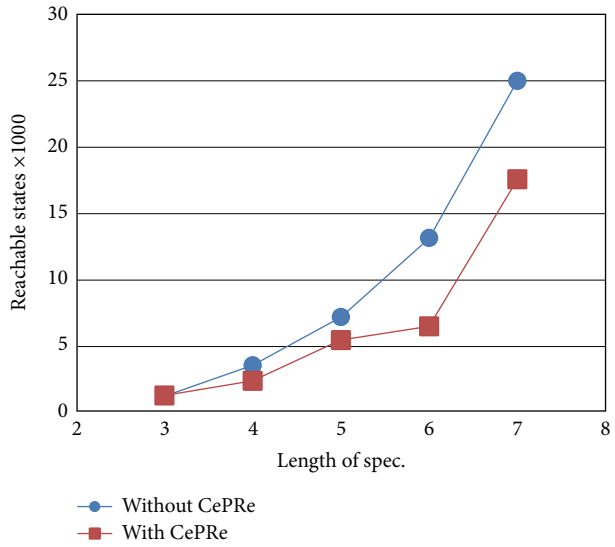

Figure 2: Results on reachable states in random BDD-MC experiments.

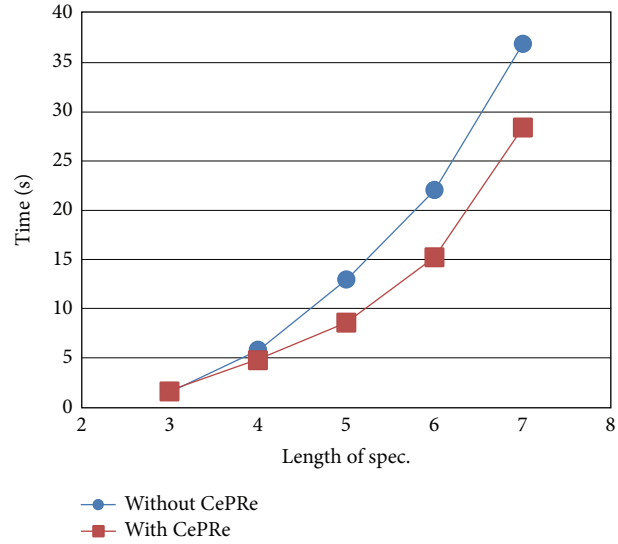

Figure 3: Time overhead in random BDD-MC experiments.

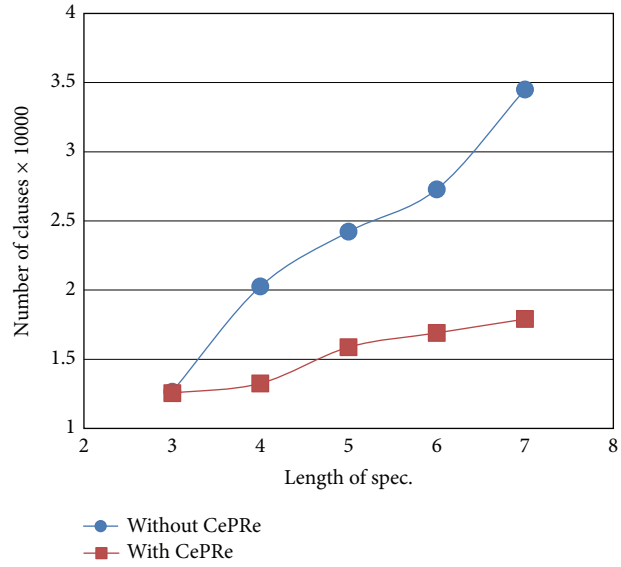

Figure 4: The scale of clauses in random BMC experiments.

6. Concluding Remarks

In this paper, we present a new technique to reduce LTL specifications' complexity towards symbolic model checking, namely, CePRe. The novelty in this technique is that the reduced formula needs not be logically equivalent with the original one but just preserves the counterexample set. Moreover, the condition enabling such a reduction can be usually detected with lightweight approaches, such as SAT-solving. Hence, this technique could leverage the power of SAT-solvers.

The central part of CePRe is a set of reduction rules, and soundness of these reduction rules is fairly easy to check. For the model-dependent rules, additional conditions mainly concern the invariants and transitions, and we do not make a sufficient use of other features, such as fairness. In this paper, the rules are given by enumerating all possible combinations of (at most two) temporal operators. Indeed, there might be some other reduction schemas we are not aware of.

From the experimental results, we can see that in a statistical perspective, a better performance and lower overhead can be achieved with CePRe. For the future work, we would consider how to extend our idea to symbolic model checking upon rich assertional languages, such as PSL.

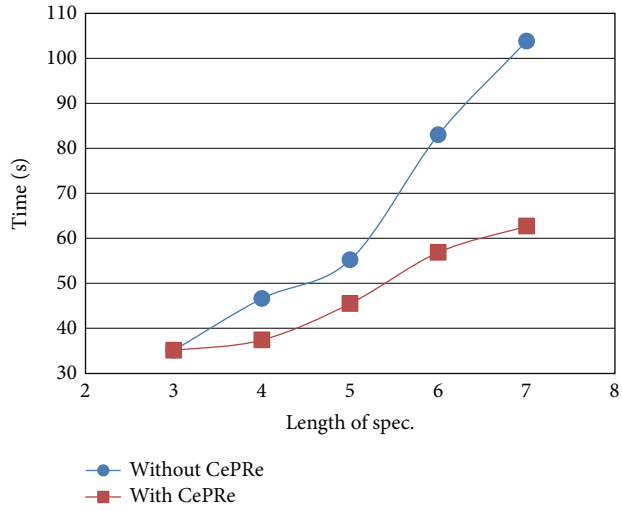

Figure 5: Time overhead in random BMC experiments.

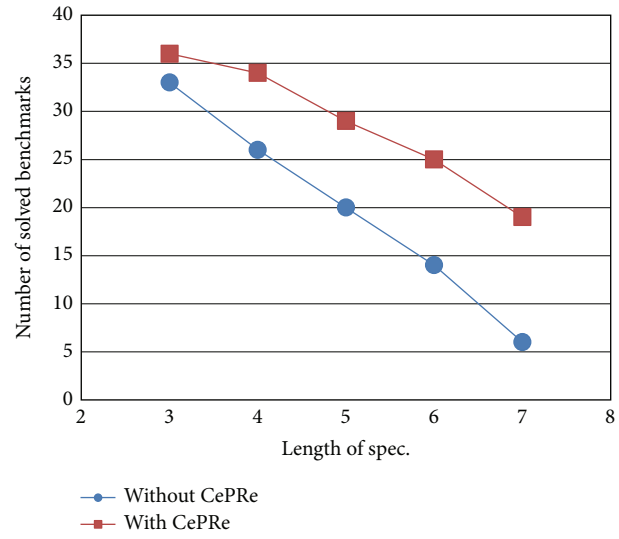

Figure 6: Experimental results of random PDR-based model checking.

Conflict of Interests

A conference version of this paper [26] is firstly published in the proceeding of ICTAC 2013, and a corresponding unpublished version is put on CoRR (http://arxiv.org/abs/1301.3299). The authors have extended more than 1/3 new material in this submission. The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper is the extension of [26], and the authors would to thank the anonymous reviewers for the helpful comments of their conference version. This work is supported by NSFC (No. 61103012), the 973 Project (no. 2014CB340703), NSFC (nos. 61379054 and 61120106006), and the Program for New Century Excellent Talents in University.

References

[1] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS '77)*, pp. 46–57, IEEE Computer Society, 1977.

[2] M. Y. Vardi, "Branching vs. linear time: final showdown," in *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 2031 of *Lecture Notes in Computer Science*, pp. 1–22, Springer, 2001.

[3] F. Somenzi and R. Bloem, "Efficient B üchi automata from LTL formulae," in *Computer Aided Verification*, E. A. Emerson and A. P. Sistla, Eds., vol. 1855 of *Lecture Notes in Computer Science*, pp. 53–65, Springer, 2000.

[4] A. Biere, K. Heljanko, T. Junttila, T. Latvala, and V. Schuppan, "Linear encodings of bounded LTL model checking," *Logical Methods in Computer Science*, vol. 2, no. 5, pp. 1–64, 2006.

[5] K. L. McMillan, *Symbolic model checking, an approach to the state explosion problem [Ph.D. thesis]*, Carnegie Mellon University, Kluwer Academic Publishers, 1993.

[6] E. M. Clarke, O. Grumberg, and K. Hamaguchi, "Another look at LTL model checking," in *Formal Methods in System Design*, vol. 818 of *Lecture Notes in Computer Science*, pp. 415–427, Springer, 1994.

[7] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," in *Proceedings of the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '99)*, vol. 1579 of *Lecture Notes in Computer Science*, pp. 193–207, Springer, 1999.

[8] A. R. Bradley, "SAT-based model checking without unrolling," in *Verification, Model Checking, and Abstract Interpretation*, vol. 6538 of *Lecture Notes in Computer Science*, pp. 70–87, Springer, 2011.

[9] F. Somenzi and A. R. Bradly, "IC3: where monolithic and incremental meet," in *Proceedings of the International Conference on Formal Methods in Computer-Aided Design (FMCAD '11)*, P. Bjesse and A. Sloodova, Eds., pp. 3–8, FMCAD, 2011.

[10] N. Een, A. Mishchenko, and R. Brayton, "Efficient implementation of property directed reachability," in *Proceedings of the Formal Methods in Computer-Aided Design (FMCAD '11)*, pp. 125–134, Austin, Tex, USA, November 2011.

[11] A. R. Bradley, "Understanding IC_3," in *Theory and Applications of Satisfiability Testing—SAT 2012*, vol. 7317 of *Lecture Notes in Computer Science*, pp. 1–14, Springer, 2012.

[12] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 10^{20} states and beyond," *Information and Computation*, vol. 98, no. 2, pp. 142–170, 1992.

[13] D. Gabbay, "The declarative past and imperative future. Executable temporal logic for interactive systems," in *Temporal Logic in Specification*, vol. 398 of *Lecture Notes in Computer Science*, pp. 409–448, Springer, 1989.

[14] O. Kupferman and M. Y. Vardi, "Modular model checking," in *Compositionality: The Significant Difference*, vol. 1536 of *Lecture Notes in Computer Science*, pp. 381–401, Springer, 1998.

[15] A. Cimatti, M. Pistore, M. Roveri, and R. Sebastiani, "Improving the encoding of LTL model checking into SAT," in *Verification, Model Checking, and Abstract Interpretation*, vol. 2294 of *Lecture Notes in Computer Science*, pp. 196–207, Springer.

[16] T. Latvala, A. Biere, K. Heljanko, and T. Junttila, "Simple bounded LTL model checking," in *Formal Methods in Computer-Aided Design*, A. Hu and A. Martin, Eds., vol. 3312 of *Lecture Notes in Computer Science*, pp. 186–200, Springer, 2004.

[17] T. Latvala, A. Biere, K. Heljanko, and T. Junttila, "Simple is better: efficient bounded model checking for past LTL," in *Verification, Model Checking, and Abstract Interpretation*, vol. 3385 of *Lecture Notes in Computer Science*, pp. 380–395, Springer, 2005.

[18] A. Zbrzezny, "A new translation from ETCL* to SAT," in *Proceedings of the International Workshop CS&P*, M. Szczuka, Ed., pp. 589–600, September 2011.

[19] E. Clarke, D. Kroening, J. Ouaknine, and O. Strichman, "Completeness and complexity of bounded model checking," in *Verification, Model Checking, and Abstract Interpretation*, vol. 2937 of *Lecture Notes in Computer Science*, pp. 85–96, Springer, 2004.

[20] A. Frisch, D. Sheridan, and T. Walsh, "A fixpoint encoding for bounded model checking," in *Formal Methods in Computer-Aided Design*, vol. 2517 of *Lecture Notes in Computer Science*, pp. 238–255, 2002.

[21] A. R. Bradley and Z. Manna, "Checking safety by inductive generalization of counterexamples to induction," in *Proceedings of the Formal Methods in Computer Aided Design (FMCAD '07)*, pp. 173–180, November 2007.

[22] A. R. Bradley, F. Somenzi, Z. Hassan, and Y. Zhang, "An incremental approach to model checking progress properties," in *Proceedings of the Formal Methods in Computer-Aided Design (FMCAD '11)*, pp. 144–153, November 2011.

[23] AIGER, the SMVtoAIG toolkit, 2007, http://fmv.jku.at/aiger/.

[24] The IIMC tool, 2013, http://ecee.colorado.edu/wpmu/iimc/.

[25] H. Taurainen and K. Heljanko, "Testing LTL formula translation into Büchi automata," *International Journal on Software Tools For Technology Transfer*, vol. 4, no. 1, pp. 57–70, 2002.

[26] W. Liu, R. Wang, X. Fu et al., "Conterexamplepreserving reduction for symbolic model checking," in *Proceedings of the 10th International Colloquium on Theoretical Aspects of Computing (ICTAC '13)*, Z. Liu, J. Woodcock, and H. Zhu, Eds., vol. 8049 of *Lecture Notes in Computer Science*, pp. 249–266, Springer, Shanghai, China, 2013.