A compact linear translation for bounded model checking [1]

Paul B. Jackson [2]

School of Informatics, University of Edinburgh,
Kings Buildings, Edinburgh EH9 3JZ, United Kingdom

Daniel Sheridan [3]

Adelard LLP, 10 Northampton Square,
London EC1V 0HB, United Kingdom

Abstract

We present a syntactic scheme for translating future-time LTL bounded model checking problems into propositional satisfiability problems. The scheme is similar in principle to the *Separated Normal Form* encoding proposed in [5] and extended to past time in [3]: an initial phase involves putting LTL formulae into a normal form based on linear-time fixpoint characterisations of temporal operators.

As with [3] and [7], the size of propositional formulae produced is linear in the model checking bound, but the constant of proportionality appears to be lower.

A denotational approach is taken in the presentation which is significantly more rigorous than that in [5] and [3], and which provides an elegant alternative way of viewing fixpoint based translations in [7] and [1].

Key words: Bounded Model Checking, Linear Temporal Logic, Fixpoints, SAT, Denotational Semantics

1 Introduction

Frisch, Sheridan and Walsh [5] proposed a scheme for translating LTL bounded model checking problems into satisfiability problems that is significantly different from the original bounded model checking encoding scheme presented in [2]. This scheme involves simplifying temporal formulae using rules based on fixpoint characterisations of temporal operators to put formulae into a

[1] This research was funded in part by UK EPSRC Grant GR/N64243/01
[2] Email: pbj@inf.ed.ac.uk
[3] Email: dan.sheridan@contact.org.uk

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

separated normal form (SNF) similar to that used by Fisher in his temporal resolution work [4]. Frisch, Sheridan and Walsh [5] showed that this new scheme had significant advantages in terms of compactness of propositional formulae generated and SAT solver run times. This SNF approach smoothly extends to handle past time LTL [3] and has similarities with automata-based translations [8].

In this paper, we present an alternate set of simplification rules for future time LTL that again exploits fixpoint characterisations, but is simpler to describe. As with [3] and [7], the size of propositional formulae produced is linear in the model checking bound,[4] and the constant of proportionality is smaller than with [7].

A major contribution of the paper is in providing a denotational semantics approach to justifying the encoding. This justification is much more complete and rigorous than that in [5] and [3], and it enables easy exploration of variations on these and other encodings.

A minor novelty is that we experiment with using an abstract symbolic representation of Kripke structures. Most formal presentations of BMC conflate a description of the BMC translation from LTL syntax to propositional logic syntax with a description of its semantics, and only informally refer to possible symbolic representations (for example, using propositional formulae, BDDs or Boolean circuits) of Kripke structures. Our approach allows us to keep the translation and semantics distinct. While our approach is more verbose, we argue that it is easier to understand, especially when handling the auxiliary variables introduced by our translation.

Our implementation is not yet complete so we do not have empirical data on SAT solver performance on the resulting encoded problems. We certainly expect the performance to be no worse than with SNF because of the similarity.

The structure of the rest of the paper is as follows. In Section 2 we present the foundations of our denotational approach, closely following the logic of the original BMC translation from [2]. Section 3 then gives a high-level overview of our new translation. The translation is split into two phases: the normalisation phase is covered in Section 4 and the translation to propositional logic phase in Section 5. Section 6 covers related work and we draw our conclusions in Section 7.

2 Preliminaries

2.1 Syntax for LTL

Fix some set V of Boolean-valued state variables. We use these as the atomic propositions of our LTL formulae. We initially consider LTL formulae de-

[4] The super-linear behaviour of the SNF encoding as noted in [7] was obtained with an older version of the SNF code than that presented in [3]

scribed by the grammar

$$\phi ::= v \mid \neg v \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathbf{X}\,\phi \mid \mathbf{F}\,\phi \mid \mathbf{G}\,\phi \mid \phi\,\mathbf{U}\,\phi \mid \phi\,\mathbf{R}\,\phi$$

where $v \in V$. Such formulae are in *negation normal form* (NNF): negations are only applied to state variables. Any LTL formula can be transformed into an equivalent NNF formula by pushing negations inwards. We use $\phi \Rightarrow \psi$ as an abbreviation for $\neg\phi \vee \psi$.

2.2 Kripke structures

The set of states S associated with a set of state variables V is the set of valuations $V \to \mathbb{B}$ of those variables. A *Kripke structure* M over a set of state variables V is a pair $\langle I, T\rangle$ where $I \subseteq S$ is a set of initial states and $T \subseteq S \times S$ is a transition relation which has to be total. Many treatments of Kripke Structures consider the set of states S more abstractly and introduce a labelling function specifying which atomic propositions are true in each state. It is straightforward to adapt our presentation to this more general approach, but, for simplicity, we do not. A similar simplification is common in automata-based approaches to LTL model checking. An *unconstrained path* π over M is an infinite sequence of states $\pi = s_0, s_1, \ldots$ where $s_i \in S$. A *constrained path* or simply a *path* over M must satisfy the constraints $s_0 \in I$ and, for every $i \geq 0$, $\langle s_i, s_{i+1}\rangle \in T$. Let $\mathsf{Paths}(M)$ be the set of all paths over M. A *finite path* is a finite prefix of a path. A finite path $s_0, s_1, \ldots, s_{k-1}$ *has bound* k.

We denote distinct copies of the set of state variables using superscripts. For example, V', V^i for $i \in \mathbb{N}$. If $v \in V$, the corresponding variable in V' is v' and in V^i is v^i. A *symbolic Kripke structure* $\hat{M}$ over a set of state variables V is a pair $\langle \hat{I}, \hat{T}\rangle$ where $\hat{I}(V)$ is a symbolic representation of the set of initial states and $\hat{T}(V, V')$ is a symbolic representation of the transition relation. The notation $A(V)$ here indicates that the symbolic representation A is over the variables V. We then write elsewhere $A(W)$ for A with the variables V replaced with the variables W. The Kripke structure corresponding to $\hat{M}$ has $I \doteq \{s \in S \mid s \models \hat{I}\}$ and $T \doteq \{\langle s, t\rangle \in S \times S \mid s, t \models \hat{T}\}$. This definition uses satisfiability relations $\models$ for single states and pairs of states satisfying a propositional formula defined in the expected way.

2.3 Infinite path semantics

A common approach to LTL semantics is to define an inductive relation $\pi \models^i \phi$ indicating at which positions $i \in \mathbb{N}$ on path π the LTL formula ϕ is satisfied. We give an exactly equivalent definition in a denotational style. We define the infinite denotation ${}^{\pi}[\![\phi]\!]$ of formula ϕ to be an infinite sequence of $a_0, a_1, \ldots$ of Boolean values, elements of $\mathbb{B} = \{\bot, \top\}$, such that a_i is true just when ϕ is satisfied at position i of path π. We write the set of all such infinite boolean sequences as $\mathbb{B}^\omega$. We often view a sequence $a \in \mathbb{B}^\omega$ as a function of type

$\mathbb{N} \to \mathbb{B}$, and refer to element i as $a(i)$. When we say that a formula is satisfied by a path without indicating an explicit position on the path, we mean that the formula is satisfied at position 0. Formally, the *infinite denotation* of an LTL formula is given inductively by:

$$
\begin{array}{llll}
{}^{\pi}[\![v]\!](i) = s_i(v) & {}^{\pi}[\![\mathbf{O}\,\phi]\!] & = [\![\mathbf{O}]\!]({}^{\pi}[\![\phi]\!]) & \text{for } \mathbf{O} \in \{\mathbf{X}, \mathbf{F}, \mathbf{G}\} \\
{}^{\pi}[\![\neg v]\!] = [\![\neg]\!]({}^{\pi}[\![v]\!]) & {}^{\pi}[\![\phi\,\mathbf{O}\,\psi]\!] & = [\![\mathbf{O}]\!]({}^{\pi}[\![\phi]\!], {}^{\pi}[\![\psi]\!]) & \text{for } \mathbf{O} \in \{\wedge, \vee, \mathbf{U}, \mathbf{R}\}
\end{array}
$$

where the individual operator denotations are given by

$$
\begin{array}{llll}
[\![\neg]\!](a)(i) \doteq \neg a(i) & & [\![\mathbf{F}]\!](a)(i) \doteq \exists j \geq i.\ a(j) \\
[\![\wedge]\!](a,b)(i) \doteq a(i) \wedge b(i) & & [\![\mathbf{G}]\!](a)(i) \doteq \forall j \geq i.\ a(j) \\
[\![\vee]\!](a,b)(i) \doteq a(i) \vee b(i) & & [\![\mathbf{U}]\!](a,b)(i) \doteq \exists j \geq i.\ b(j) \wedge \forall n \in \{i\,..\,j-1\}.\ a(n) \\
[\![\mathbf{X}]\!](a)(i) \doteq a(i+1) & & [\![\mathbf{R}]\!](a,b)(i) \doteq \forall j \geq i.\ b(j) \vee \exists n \in \{i\,..\,j-1\}.\ a(n)
\end{array}
$$

Here $a, b \in \mathbb{B}^\omega$ are infinite denotations and $i \in \mathbb{N}$ indexes positions in denotations. These explicit denotations for operators help simplify the presentation later. Their use emphasises that the meaning of operators is dependent only on the meaning of subformulae, not on the syntactic structure of subformulae.

Let us write $\phi \equiv \psi$ when LTL formulae ϕ and ψ have the same infinite denotation for all Kripke structures M and paths π over those structures.

2.4 Finite denotations when paths are looping

In producing finite propositional encodings of model checking problems, bounded model checking works with finite representations of infinite paths and infinite denotations. In this subsection we consider *loop case* representations. In the next subsection we consider *prefix case* representations.

In the loop case with bound k and loop start l where $0 \leq l < k$, a finite path $\dot{\pi} = s_0, \ldots, s_{k-1}$ such that $T(s_{k-1}, s_l)$ represents the infinite path $s_0 \ldots s_{l-1}(s_l \ldots s_{k-1})^\omega$. We call such infinite paths *(k, l) loop paths*. Similarly, finite loop-case denotations such as $\dot{a} = a_0, \ldots, a_{k-1}$ where $a_i \in \mathbb{B}$ represent infinite denotations $a_0 \ldots a_{l-1}(a_l \ldots a_{k-1})^\omega$. A *loop-case inflation* function $\uparrow_\circ^\infty$ maps finite paths and denotations to the corresponding infinite paths and denotations. A *restriction* function $|_k$ maps (k, l) loop paths and infinite loop-case denotations to their finite representations.

When working with loop paths and their finite denotations, we can define a *finite loop-case denotation function* ${}^{\dot{\pi}}_{l}{}^{\mathrm{F}}[\![\phi]\!]_k$ with range $\mathbb{B}^k$ that exactly mimics the infinite denotation function:

$$
{}^{\dot{\pi}\uparrow_\circ^\infty}[\![\phi]\!] = {}^{\dot{\pi}}_{l}{}^{\mathrm{F}}[\![\phi]\!]_k \uparrow_\circ^\infty
$$

where $\dot{\pi}$ is a k-bounded path representing a (k, l) loop path. The definition is similar to that of the infinite denotation with the following changes:

$$
\begin{array}{rcl}
{}^{\mathrm{F}}_{l}[\![\mathbf{X}]\!]_k(\dot{a})(i) & \doteq & \begin{cases} \dot{a}(i{+}1) & \text{if } i < k{-}1 \\ \dot{a}(l) & \text{if } i = k{-}1 \end{cases} \\
{}^{\mathrm{F}}_{l}[\![\mathbf{F}]\!]_k(\dot{a})(i) & \doteq & \exists j \in \{\min(i,l) \mathinner{..} k{-}1\}.\ \dot{a}(j) \\
{}^{\mathrm{F}}_{l}[\![\mathbf{G}]\!]_k(\dot{a})(i) & \doteq & \forall j \in \{\min(i,l) \mathinner{..} k{-}1\}.\ \dot{a}(j) \\
{}^{\mathrm{F}}_{l}[\![\mathbf{U}]\!]_k(\dot{a},\dot{b})(i) & \doteq & (\exists j \in \{i \mathinner{..} k{-}1\}.\ \dot{b}(j) \wedge \forall n \in \{i \mathinner{..} j{-}1\}.\ \dot{a}(n)) \\
 & & \vee\, \exists j \in \{l \mathinner{..} i{-}1\}.\ \dot{b}(j) \wedge \forall n \in \{i \mathinner{..} k{-}1\} \cup \{l \mathinner{..} j{-}1\}.\ \dot{a}(n) \\
{}^{\mathrm{F}}_{l}[\![\mathbf{R}]\!]_k(\dot{a},\dot{b})(i) & \doteq & (\forall j \in \{i \mathinner{..} k{-}1\}.\ \dot{b}(j) \vee \exists n \in \{i \mathinner{..} j{-}1\}.\ \dot{a}(n)) \\
 & & \wedge\, \forall j \in \{l \mathinner{..} i{-}1\}.\ \dot{b}(j) \vee \exists n \in \{i \mathinner{..} k{-}1\} \cup \{l \mathinner{..} j{-}1\}.\ \dot{a}(n)
\end{array}
$$

where $\dot{a}, \dot{b} \in \mathbb{B}^k$ are finite denotations and index i is in range $\{0 \mathinner{..} k{-}1\}$.

All quantifications in the finite loop-case denotation function are over finite ranges. Following the denotation function's structure, we can define an executable *loop-case translation function* ${}_{l}[\phi]^{i}_{k}$ that can translate a question concerning the existence of a finite path loop-case satisfying a formula into a propositional satisfiability question. The relationship between the loop-case denotation and translation functions is expressed by:

$$
{}^{\dot{\pi}}_{l}[\![\phi]\!]^{\mathrm{F}}_{k}(i) \quad \Leftrightarrow \quad \dot{\pi} \models {}_{l}[\phi]^{i}_{k}
$$

where the relation $\dot{\pi} \models q$ for when a finite path $\dot{\pi}$ satisfies a propositional formula q is defined in the expected way. Representative cases of the definition of the loop-case translation function are

$$
{}_{l}[v]^{i}_{k} \doteq v^i \qquad\qquad {}_{l}[\mathbf{F}\,\phi]^{i}_{k} \doteq \bigvee\nolimits_{j=\min(i,l)}^{k-1} {}_{l}[\phi]^{j}_{k}
$$

In the original paper introducing BMC [2] and many other papers in the BMC literature, symbolic Kripke structures are not explicitly introduced and confusing semantic notations occur in translation function definitions. For example, the base case of the translation function might be written as ${}_{l}[v]^{i}_{k} \doteq v(s_i)$ where a state variable v is treated as function which it is not, and a state s_i is introduced which is part of the semantic presentation, not part of the language of propositional logic that is being translated into.

2.5 Finite denotations when paths have common prefix

In the prefix case with bound k, a finite path $\dot{\pi} = s_0, \ldots, s_{k-1}$ represents the set of all paths that have it as a prefix. Prefix-case denotations such as $\dot{a} = a_0, \ldots, a_{k-1}$ where $a_i \in \mathbb{B}$ represent infinite denotations $\dot{a}\perp^{\omega}$. A *prefix-case inflation* function $\uparrow^{\infty}$ maps finite denotations to the corresponding infinite denotations. The restriction function $\pi|_k$ introduced in the last section is also used to select the k-bounded prefix of an infinite path π.

We define a *finite prefix-case denotation function* $\dot{\pi}\overset{\text{F}}{[\![\phi]\!]}_k$ (or sometimes $\dot{\pi}\overset{\text{F}}{[\![\phi]\!]}_k$) in a similar way to the the finite loop-case denotation function. In this case, the denotations for the LTL temporal operators are given by:

$$\overset{\text{F}}{[\![\mathbf{X}]\!]}_k(\dot{a})(i) \doteq \begin{cases} \dot{a}(i+1) & \text{if } i < k-1 \\ \bot & \text{if } i = k-1 \end{cases} \qquad \overset{\text{F}}{[\![\mathbf{F}]\!]}_k(\dot{a})(i) \doteq \exists j \in \{i \mathinner{..} k-1\}.\ \dot{a}(j)$$

$$\overset{\text{F}}{[\![\mathbf{G}]\!]}_k(\dot{a})(i) \doteq \bot$$

$$\overset{\text{F}}{[\![\mathbf{U}]\!]}_k(\dot{a}, \dot{b})(i) \doteq \exists j \in \{i \mathinner{..} k-1\}.\ \dot{b}(j) \wedge \forall n \in \{i \mathinner{..} j-1\}.\ \dot{a}(n)$$

$$\overset{\text{F}}{[\![\mathbf{R}]\!]}_k(\dot{a}, \dot{b})(i) \doteq \exists j \in \{i \mathinner{..} k-1\}.\ \dot{a}(j) \wedge \forall n \in \{i \mathinner{..} j\}.\ \dot{b}(n)$$

The prefix case denotation underapproximates the standard infinite denotation and so is sound. We can express this by the assertion

$$^{\pi|_k}\overset{\text{F}}{[\![\phi]\!]}_k \uparrow^\infty \sqsubseteq\ ^{\pi}[\![\phi]\!]$$

where π is any infinite path and we are treating the domain of infinite denotations $\mathbb{B}^\omega$ as a lattice with order relation $a \sqsubseteq b \doteq \forall i \in \mathbb{N}.\ a(i) \Rightarrow b(i)$. As with the loop-case, we can derive a prefix translation function $[\phi]^i_k$ (sometimes written as $\_[\phi]^i_k$) from the prefix-case denotation function.

3 The new full translation

We describe here the high-level structure and properties of our translation in order to motivate the details in subsequent sections. The translation takes an LTL formula ϕ, symbolic Kripke structure $\hat{M}$ and bound k and creates a propositional formula that is satisfiable just when some path in $\hat{M}$ with a k-bounded representation satisfies ϕ. Conceptually the translation proceeds in 3 stages:

1. Apply normalisation function $\mathcal{N}()$ to ϕ to create normalised temporal logic formula ψ.
2. Create a formula

$$[\hat{M}]_k \wedge \left([\psi]^0_k \vee \bigvee_{l=0}^{k-1} {}_lL_k(\hat{M}) \wedge {}_l[\psi]^0_k\right) \tag{1}$$

that brings together the prefix case and loop case translations of ψ and correspondingly checks that an unconstrained finite path is representing a prefix case or a loop case path. Here $[\hat{M}]_k \doteq \hat{I}(V^0) \wedge \bigwedge_{i=0}^{k-2} \hat{T}(V^i, V^{i+1})$ generates a proposition for checking that a finite state sequence is a finite path and ${}_lL_k(\hat{M}) \doteq \hat{T}(V^{k-1}, V^l)$ is a constraint for specifying that a finite path represents a (k, l) loop path.

3. Apply standard logic transformations so as to collect together common factors in the disjuncts of Formula (1) and ensure the formula's size is linear in k.

The original translation of [2] consists of step 2 without steps 1 and 3. Even with careful optimisations, the size of the original translation is claimed in [7] to be cubic in the worst case. Our normalisation in step 1 enables the factoring for linear size in step 3.

We formally write our full translation as:

$$\text{Full}[\hat{M}, \phi]_k \doteq \text{body}\big(\text{Norm}[\ \hat{M},\ \mathcal{N}(\phi)\ ]_k\big)$$

where the normalised-formula translation function $\text{Norm}[\hat{M}, \psi]_k$ groups together steps 2 and 3. This translation function produces formulae of form $\exists\, \overline{z}.\ q$ where $\overline{z}$ is a vector of propositional variables and q is a propositional logic formula. The function body() returns the body q of such formulae. This existential quantification $\exists \overline{z}$ arises because $\mathcal{N}()$ produces formulae in LTL extended with existential quantification. See Section 4 for a full definition of $\mathcal{N}()$ and Section 5 for a full definition of $\text{Norm}[\hat{M}, \psi]_k$.

To state the correctness of our full translation, we introduce a reference semantics which combines the infinite and finite prefix-case semantics. A (k, l) loop path π *satisfies at bound* k an LTL formula ϕ if π satisfies it in the standard infinite semantics (if ${}^{\pi}[\![\phi]\!](0)$ holds). If π is not a (k, l) loop path for any l, then π *satisfies at bound* k a formula ϕ if the k-bounded prefix of π satisfies ϕ in the finite prefix case semantics (if ${}^{\pi|_k}[\![\phi]\!]^{\text{F}}_k(0)$ holds). A formula ϕ is *existentially valid with bound* k in Kripke structure M, written $M \models_k \mathbf{E}\,\phi$, when some path π of M satifies ϕ at bound k. We can now state the overall correctness claim for our new translation as follows.

Theorem 3.1 ***(Correctness of new translation).*** *For any symbolic Kripke structure $\hat{M}$ with corresponding semantic structure M, any LTL formula ϕ and any bound $k > 0$, we have that*

$$M \models_k \mathbf{E}\,\phi \quad \Leftrightarrow \quad \text{Full}[\hat{M}, \phi]_k \ \textit{is satisfiable}$$

4 Formula normalisation

4.1 Overview

Normalisation proceeds in two main stages. Firstly the LTL operators **F**, **G**, **U**and **R** in the input formula are all converted into forms involving greatest fixpoint operators. Section 4.3 handles how this is done with **G** and **R**, operators with natural greatest fixpoint characterisations and Section 4.4 handles the more subtle case of **F** and **U** which have natural least fixpoint characterisations. Secondly, as described in Section 4.5, each greatest fixpoint expression is converted into a form involving existential quantification at the outermost level

of the formula. Section 4.5 also explains why least fixpoint characterisations cannot be handled. Normalisation also involves some renaming transforms on $\mathbf{X}$ in the input formula and on certain new formulae produced in the first normalisation stage. Section 4.6 covers renaming transforms in general. Finally Section 4.7 gives a self-contained summary of the normalisation function.

The interesting part of the proof of Theorem 3.1 involves showing the following two equations concerning normalisation:

$$\dot{\pi}\!\uparrow_{\circ}^{\infty}[\![\phi]\!] = \overset{\mathrm{F}}{\dot{\pi}}_{l}[\![\mathcal{N}(\phi)]\!]_k \uparrow_{\circ}^{\infty} \tag{2}$$

$$\overset{\mathrm{F}}{\dot{\pi}}[\![\phi]\!]_k = \overset{\mathrm{F}}{\dot{\pi}}[\![\mathcal{N}(\phi)]\!]_k \tag{3}$$

where ϕ is any LTL formula and $l \in \{0\,..\,k-1\}$. Equation (2) states that the finite loop-case denotation of normalized formulae is equivalent to their standard infinite denotation. Equation (3) states that the prefix-case denotation is preserved by normalisation. The subsections which follow include assertions of equalities which are intermediate steps in the proofs of Equation (2) and Equation (3).

We write $\phi \equiv_L \psi$ ($\phi \equiv_P \psi$) when two formulae always have the same finite loop-case (prefix-case) denotation, and $\phi \equiv_F \psi$ when they always have the same denotation under both finite semantics.

4.2 Extending LTL with a greatest fixpoint operator

We add to the syntax of LTL formulae *timed variables* α (also known as *flexible variables*), and greatest fixpoint expressions $\nu\alpha.\ \phi$ with infinite and finite semantics:

$$\begin{aligned}
{}^{\pi}[\![\alpha]\!]^{\rho} &= \rho(\alpha) & \overset{\mathrm{F}}{\pi}_{l}[\![\alpha]\!]^{\dot\rho}_{k} &= \dot\rho(\alpha)\\
{}^{\pi}[\![\lambda\alpha.\phi]\!]^{\rho} &= \lambda a \in \mathbb{B}^{\omega}.\ {}^{\pi}[\![\phi]\!]^{\rho[\alpha\mapsto a]} & \overset{\mathrm{F}}{\dot\pi}_{l}[\![\lambda\alpha.\phi]\!]^{\dot\rho}_{k} &= \lambda \dot a \in \mathbb{B}^{k}.\ \overset{\mathrm{F}}{\dot\pi}_{l}[\![\phi]\!]^{\dot\rho[\alpha\mapsto \dot a]}_{k}\\
{}^{\pi}[\![\nu\alpha.\phi]\!]^{\rho} &= \mathrm{gfp}({}^{\pi}[\![\lambda\alpha.\phi]\!]^{\rho}) & \overset{\mathrm{F}}{\dot\pi}_{l}[\![\nu\alpha.\phi]\!]^{\dot\rho}_{k} &= \mathrm{gfp}(\overset{\mathrm{F}}{\dot\pi}_{l}[\![\lambda\alpha.\phi]\!]^{\dot\rho}_{k})
\end{aligned}$$

where $l \in \{0\,..\,k-1\} \cup \{-\}$. Lambda abstractions $\lambda\alpha.\phi$ are examples of *unary function formulae*. To provide meaning in the semantics for free variables, we extend the semantic functions with an environment argument ρ or $\dot\rho$. An *unbounded environment* ρ maps each free variable to an infinite sequence in $\mathbb{B}^{\omega}$ and a *k-bounded environment* $\dot\rho$ maps each free variable to a finite sequence in $\mathbb{B}^{k}$. In the other previously-defined clauses of the semantic functions, the environments are recursively propagated down unchanged.

The greatest fixpoint operator gfp is given the standard definition from the Tarski-Knaster construction. Let F be a monotone function of type $D \to D$ on a complete lattice $\langle D, \sqsubseteq\rangle$ with least upper bound operator $\bigsqcup$. We have that $\mathrm{gfp}(F) \doteq \bigsqcup\{x \in D | x \sqsubseteq F(x)\}$. In all our semantics D is of form $R \to \mathbb{B}$. In the infinite semantics $R = \mathbb{N}$ and in both finite semantics $R = \{0\,..\,k-1\}$. The order relation $\sqsubseteq$ and least upper bound operation $\bigsqcup$ are defined pointwise:

$x \sqsubseteq y \doteq \forall i.\, x(i) \Rightarrow y(i)$ and $(\bigsqcup S)(i) \doteq \exists x \in S.\, x(i)$ where lattice elements $x, y \in R \to \mathbb{B}$, set of elements $S \subseteq R \to \mathbb{B}$ and index $i \in R$.

4.3 Greatest fixpoint characterisations for $\boldsymbol{G}$ *and* $\boldsymbol{R}$

Fixpoint versions of the *globally* operator $\mathbf{G}$ and the *release* operator $\mathbf{R}$ are

$$\tilde{\mathbf{G}}\,\beta \doteq \nu\alpha.\, \beta \wedge \mathbf{X}\,\alpha \qquad\qquad \beta\,\tilde{\mathbf{R}}\,\gamma \doteq \nu\alpha.\, \gamma \wedge (\beta \vee \mathbf{X}\,\alpha)$$

Our following discussion focusses the $\tilde{\mathbf{G}}$ operator. It extends very straightforwardly to cover the $\tilde{\mathbf{R}}$ operator too.

It is well known that the standard $\mathbf{G}$ is equivalent to this fixpoint version in the infinite semantics: $\mathbf{G}\,\beta \equiv \tilde{\mathbf{G}}\,\beta$. It is straightforward to check that this equivalence also holds in the prefix semantics. For example, it is easy to show $\mathbf{G}\,\beta \equiv_P \tilde{\mathbf{G}}\,\beta$ once one observes that $\lambda\alpha.\, \beta \wedge \mathbf{X}\,\alpha$ has a unique fixpoint in the prefix semantics when a binding for β is fixed. Indeed, if one adds least fixpoint operators $\mu\alpha.\, \phi$ to LTL, one can also make the definitions

$$\tilde{\mathbf{F}}\,\beta \doteq \mu\alpha.\, \beta \vee \mathbf{X}\,\alpha \qquad\qquad \beta\,\tilde{\mathbf{U}}\,\gamma \doteq \mu\alpha.\, \gamma \vee (\beta \wedge \mathbf{X}\,\alpha)$$

and show $\mathbf{F}\,\beta \equiv_P \tilde{\mathbf{F}}\,\beta$ and $\beta\,\mathbf{U}\,\gamma \equiv_P \beta\,\tilde{\mathbf{U}}\,\gamma$. This provides some justification for the naturalness of the prefix-case semantics of the LTL operators.

In the proof of Equation (2), an appropriate stage of normalisation for shifting to the finite semantics is after $\tilde{\mathbf{G}}$ and $\tilde{\mathbf{R}}$ have been introduced. With $l \in \{0\,..\,k{-}1\}$ and $\dot{b} \in \mathbb{B}^k$, we have the following:

$$[\![\tilde{\mathbf{G}}]\!](\dot{b}\uparrow^\infty_\circ) = \big({}^{\mathrm{F}}_{l}[\![\tilde{\mathbf{G}}]\!]_k(\dot{b})\big)\uparrow^\infty_\circ$$

4.4 Greatest fixpoint characterisations for $\boldsymbol{F}$ *and* $\boldsymbol{U}$

As noted in the previous section, in the prefix case the fixpoint with operators $\tilde{\mathbf{F}}$ and $\tilde{\mathbf{U}}$ is unique, the least and greatest fixpoints are the same. For example, we have that: $\tilde{\mathbf{F}}\,\beta \equiv_P \nu\alpha.\, \beta \vee \mathbf{X}\,\alpha$. The loop-case is not so simple. Consider the loop-case semantics for $\mathbf{F}$.

$${}^{\mathrm{F}}_{l}[\![\mathbf{F}]\!]_k(\dot{a})(i) = \exists j \in \{\min(i,l)\,..\,k{-}1\}.\, \dot{a}(j)$$

The right-hand side here is equivalent to

$$(\exists j \in \{i\,..\,k{-}1\}.\, \dot{a}(j)) \;\vee\; (\exists j \in \{l\,..\,k{-}1\}.\, \dot{a}(j))$$

Each disjunct here is an instance of the prefix semantics for $\mathbf{F}$ and, as above, we know we can switch to greatest fixpoints in the prefix semantics. We craft some definitions of new operators to take advantage of this observation.

Let us introduce variations $\mathbf{X}^\top$ and $\mathbf{X}^\bot$ on the next step operator that have non-looping semantics even in the loop case. Their finite semantics is

$$ {}^{\mathrm{F}}_{l}[\![\mathbf{X}^{\top}]\!]_k(\dot{a})(i) \doteq \begin{cases} \dot{a}(i+1) & \text{if } i < k-1 \\ \top & \text{if } i = k-1 \end{cases} \quad {}^{\mathrm{F}}_{l}[\![\mathbf{X}^{\bot}]\!]_k(\dot{a})(i) \doteq \begin{cases} \dot{a}(i+1) & \text{if } i < k-1 \\ \bot & \text{if } i = k-1 \end{cases} $$

where $l \in \{0 \mathinner{..} k-1\} \cup \{-\}$. We use these in the definitions

$$ \tilde{\mathbf{F}}^{\bot}\,\beta \doteq \nu\alpha.\ \beta \vee \mathbf{X}^{\bot}\,\alpha \qquad \tilde{\mathbf{G}}^{\top}\,\beta \doteq \nu\alpha.\ \beta \wedge \mathbf{X}^{\top}\,\alpha $$
$$ \beta\,\tilde{\mathbf{U}}^{\bot}\,\gamma \doteq \nu\alpha.\ \gamma \vee (\beta \wedge \mathbf{X}^{\bot}\,\alpha) $$

These newly introduced fixpoint operators have the following semantic characterisations in both the prefix and the loop cases.

$$ \begin{aligned} {}^{\mathrm{F}}_{l}[\![\tilde{\mathbf{F}}^{\bot}]\!]_k(\dot{a})(i) &= \exists j \in \{i \mathinner{..} k-1\}.\ \dot{a}(j) \\ {}^{\mathrm{F}}_{l}[\![\tilde{\mathbf{G}}^{\top}]\!]_k(\dot{a})(i) &= \forall j \in \{i \mathinner{..} k-1\}.\ \dot{a}(j) \\ {}^{\mathrm{F}}_{l}[\![\tilde{\mathbf{U}}^{\bot}]\!]_k(\dot{a},\dot{b})(i) &= \exists j \in \{i \mathinner{..} k-1\}.\ \dot{b}(j) \wedge \forall n \in \{i \mathinner{..} j-1\}.\ \dot{a}(n) \end{aligned} $$

To force consideration of the semantics of an operator at the loop start in the loop case, we introduce a unary LTL operator **loopstart** with semantics

$$ {}^{\mathrm{F}}_{l}[\![\mathbf{loopstart}]\!]_k(\dot{a})(i) \doteq \begin{cases} \dot{a}(l) & \text{if } l \in \{0 \mathinner{..} k-1\} \\ \bot & \text{if } l = - \end{cases} $$

With these new operators at hand, we have the following identities allowing us to replace $\mathbf{F}$ and $\mathbf{U}$ with expressions involving greatest fixpoint operators.

$$ \begin{aligned} \mathbf{F}\,\alpha &\equiv_F \tilde{\mathbf{F}}^{\bot}\,\alpha \vee \mathbf{loopstart}\,\tilde{\mathbf{F}}^{\bot}\,\alpha \\ \alpha\,\mathbf{U}\,\beta &\equiv_F \alpha\,\tilde{\mathbf{U}}^{\bot}\,\beta \vee (\tilde{\mathbf{G}}^{\top}\,\alpha \wedge \mathbf{loopstart}(\alpha\,\tilde{\mathbf{U}}^{\bot}\,\beta)) \end{aligned} $$

The identity involving $\mathbf{F}$ can readily be derived using facts presented above. The main steps are:

$$ \begin{aligned} {}^{\pi}_{l}[\![\tilde{\mathbf{F}}^{\bot}\,\alpha \vee \mathbf{loopstart}\,\tilde{\mathbf{F}}^{\bot}\,\alpha]\!]_k(i) &= {}^{\pi}_{l}[\![\tilde{\mathbf{F}}^{\bot}\,\alpha]\!]_k(i) \vee {}^{\pi}_{l}[\![\tilde{\mathbf{F}}^{\bot}\,\alpha]\!]_k(l) \\ = (\exists j \in \{i \mathinner{..} k-1\}.\ {}^{\pi}_{l}[\![\alpha]\!]_k(j)) \vee (\exists j \in \{l \mathinner{..} k-1\}.\ {}^{\pi}_{l}[\![\alpha]\!]_k(j)) &= {}^{\pi}_{l}[\![\mathbf{F}\,\alpha]\!]_k(i) \end{aligned} $$

These identities are also closely related to those discussed in Section 6.1.

4.5 Expressing greatest fixpoints using existential operators

We focus on the cases of the two finite semantics since these are the cases we need. A similar discussion applies with the infinite semantics.

Let us augment our LTL syntax with existential quantification over timed variables $\exists\alpha.\ \phi$ and a *globally from the start* operator $\mathbf{G}_0$ which have finite semantics

$$\overset{\dot{\pi}}{}_{l}^{\mathrm{F}}[\![\exists\alpha.\ \phi]\!]_k^{\dot{\rho}}(i) \ \doteq\ \exists \dot{a} \in \mathbb{B}^k.\ \overset{\dot{\pi}}{}_{l}^{\mathrm{F}}[\![\phi]\!]_k^{\dot{\rho}[\alpha \mapsto \dot{a}]}(i)$$
$$\overset{\mathrm{F}}{}_{l}[\![\mathbf{G}_0]\!]_k(\dot{a})(i) \ \doteq\ \forall j \in \{0\,..\,k-1\}.\ \dot{a}(j)$$

where $0 \leq i < k$, $\dot{a} \in \mathbb{B}^k$ and $l \in \{0..k-1\} \cup \{-\}$. Note that the globally-from-the-start operator $\mathbf{G}_0$ always quantifies over the full time range, no matter what index i we consider its value at, even in the prefix case.

Using these definitions we can phrase an identity for eliminating greatest fixpoint expressions occurring in contexts, buried under other operators:

$$\Psi[\nu\alpha.\ \phi] \equiv_F \exists\alpha.\ \mathbf{G}_0\,(\alpha \Rightarrow \phi) \wedge \Psi[\alpha]$$

where context expression Ψ is a unary function formula with monotone denotation, and the notation $\cdot[\cdot]$ is the application operator for such functions.

The existential quantification derives from the least-upper-bound operator in the definition of the gfp operator, and semantics of the formula $\mathbf{G}_0\,(\alpha \Rightarrow \phi)$ captures the $x \sqsubseteq F(x)$ constraint in the definition body (see Section 4.2).

The corresponding identity for an lfp (least-fixpoint) operator involves a universal quantification derived from the greatest-lower-bound operator in the lfp operator definition. Since our goal is to eventually produce satisfiability problems, we cannot make use of this identity.

4.6 Renamings

An LTL formula in some context is *renamed* if it replaced by a new timed variable which is asserted equivalent to it. When contexts are monotone, it is sufficient to assert an implicational relationship between the new variable and the renamed formula. We have that

$$\Psi[\phi] \ \equiv_F\ \exists\alpha.\ \mathbf{G}_0\,(\alpha \Rightarrow \phi) \wedge \Psi[\alpha]$$

where Ψ is a monotone unary function formula.

In some cases, the formula to be replaced is *time invariant*: it has denotation $\bot^k$ or $\top^k$. In these cases, it is sufficient to replace it by an *untimed variable* (sometimes called a rigid variable) and use existential quantification over untimed variables. Let us add untimed variables x to the LTL syntax and existential quantification over them $\exists x.\ \phi$ with semantics:

$$\overset{\dot{\pi}}{}_{l}^{\mathrm{F}}[\![x]\!]_k^{\dot{\rho}}(i) \ =\ \dot{\rho}(x) \qquad\qquad \overset{\dot{\pi}}{}_{l}^{\mathrm{F}}[\![\exists x.\ \phi]\!]_k^{\dot{\rho}}(i) \ =\ \exists a_0 \in \mathbb{B}.\ \overset{\dot{\pi}}{}_{l}^{\mathrm{F}}[\![\phi]\!]_k^{\dot{\rho}[x \mapsto a_0]}(i)$$

where $0 \leq i < k$, $\dot{a} \in \mathbb{B}^k$ and $l \in \{0\,..\,k-1\} \cup \{-\}$, and we extend the notion of environment $\dot{\rho}$ to provide Boolean-valued bindings for untimed variables. We then have:

$$\Psi[\phi] \ \equiv_F\ \exists x.\ (x \Rightarrow \phi) \wedge \Psi[x]$$

where Ψ is a monotone unary function formula and ϕ a time invariant formula.

4.7 The normalisation function

We assemble here the results from the previous subsections into a single overall definition of the normalisation function $\mathcal{N}()$. Assume that formulae to start are in negation normal form. $\mathcal{N}()$ applies the following transformation rules:

$$\Psi[\mathbf{G}\, f] \quad \longrightarrow \exists\alpha.\ \Psi[\alpha] \ \wedge\ \mathbf{G}_0\,(\alpha \ \Rightarrow\ f \wedge \mathbf{X}\,\alpha)$$

$$\Psi[f\ \mathbf{R}\ g] \longrightarrow \exists\alpha.\ \Psi[\alpha] \ \wedge\ \mathbf{G}_0\,(\alpha \ \Rightarrow\ g \wedge (f \vee \mathbf{X}\,\alpha))$$

$$\Psi[\mathbf{X}\, f] \quad \longrightarrow \exists\alpha.\ \Psi[\alpha] \ \wedge\ \mathbf{G}_0\,(\alpha \ \Rightarrow\ \mathbf{X}\, f)$$

$$\Psi[\mathbf{F}\, f] \quad \longrightarrow \exists\alpha, x.\ \Psi[\alpha \vee x] \ \wedge\ \mathbf{G}_0\,(\alpha \ \Rightarrow\ f \vee \mathbf{X}^{\perp}\,\alpha) \wedge (x \Rightarrow \mathbf{loopstart}\,\alpha)$$

$$\begin{aligned}\Psi[f\ \mathbf{U}\ g] \longrightarrow \exists\alpha, \beta, x.\ \ & \Psi[\alpha \vee (\beta \wedge x)] \ \wedge\ \mathbf{G}_0\,(\alpha \ \Rightarrow\ g \vee (f \wedge \mathbf{X}^{\perp}\,\alpha)) \\ & \wedge\, \mathbf{G}_0\,(\beta \ \Rightarrow\ f \wedge \mathbf{X}^{\top}\,\beta) \ \wedge\ (x \ \Rightarrow\ \mathbf{loopstart}\,\alpha)\end{aligned}$$

These rules are applied in a single bottom-up pass over the initial formula. To suggest this bottom-up direction, the subformulae f and g are required to be propositional, free from temporal operators. Rules are not applied to any of the new generated structure, for example, new $\mathbf{X}$s. Usual assumptions are made about variables bound by the existential quantifiers being suitably renamed to avoid any unintentional capture of variables. An example of applying the normalisation function is

$$\begin{aligned}\mathbf{F}\,\mathbf{G}\,\neg p \ &\longrightarrow\ \exists\alpha.\ \underline{\mathbf{F}\,\alpha} \ \wedge\ \mathbf{G}_0\,(\alpha \Rightarrow \neg p \ \wedge\ \mathbf{X}\,\alpha) && \textit{by } \mathbf{G} \textit{ rule} \\ &\longrightarrow\ \exists\alpha, \beta, x.\ \underline{(\beta \vee x)} \ \wedge \mathbf{G}_0\,(\beta \Rightarrow \alpha \vee \mathbf{X}^{\perp}\,\beta) \\ &\qquad \wedge\, (x \Rightarrow \mathbf{loopstart}\,\beta) \ \wedge\ \mathbf{G}_0\,(\alpha \Rightarrow \neg p \wedge \mathbf{X}\,\alpha) && \textit{by } \mathbf{F} \textit{ rule}\end{aligned}$$

where, in the intermediate expression, we have underlined the partially reduced input formula that is about to be transformed by a second rule, and, in the final expression, the propositional residue of the input formula.

The resulting formulae have normal form

$$\exists\overline{\alpha},\, \overline{x}.\ R \ \wedge\ LS \ \wedge\ \mathbf{G}_0\,(X \ \wedge\ X^{*})$$

where $\overline{\alpha}$ is a vector of timed variables, $\overline{x}$ is a vector of untimed variables, R is the residual top-level propositional structure of the initial formula, LS is a conjunction of formulae of form $x \Rightarrow \mathbf{loopstart}\,\alpha$, X is a conjunction of formulae of form $\alpha \Rightarrow f[\mathbf{X}\, g]$ where context f and formula g are propositional, and X^{*} is a conjunction of formulae of form $\alpha \Rightarrow f[\mathbf{X}^{\top}\, g]$ and $\alpha \Rightarrow f[\mathbf{X}^{\perp}\, g]$ where again context f and formula g are propositional.

The function $\mathcal{N}(\phi)$ can be computed in time linear in the size ϕ.

5 Translation of normalised formulae

The loop-case and prefix-case translation functions over the syntax of the components R, LS, X and X^* of our normalised formulae are as follows:

$$
\begin{array}{lll}
{}_l[\alpha]^i_k = \alpha_i & {}_l[v]^i_k = v^i & {}_l[\phi \wedge \psi]^i_k = {}_l[\phi]^i_k \wedge {}_l[\psi]^i_k \\
{}_l[x]^i_k = x & {}_l[\neg\phi]^i_k = \neg {}_l[\phi]^i_k & {}_l[\phi \vee \psi]^i_k = {}_l[\phi]^i_k \vee {}_l[\psi]^i_k
\end{array}
$$

$$
{}_l[\mathbf{X}\,\phi]^i_k = \begin{cases} {}_l[\phi]^{i+1}_k & \text{if } i < k-1 \\ \bot & \text{if } i = k-1 \text{ and } l = - \\ {}_l[\phi]^l_k & \text{if } i = k-1 \text{ and } l \in \{0\,..\,k-1\} \end{cases}
$$

$$
{}_l[\mathbf{X}^\top\,\phi]^i_k = \begin{cases} {}_l[\phi]^{i+1}_k & \text{if } i < k-1 \\ \top & \text{if } i = k-1 \end{cases}
$$

$$
{}_l[\mathbf{X}^\bot\,\phi]^i_k = \begin{cases} {}_l[\phi]^{i+1}_k & \text{if } i < k-1 \\ \bot & \text{if } i = k-1 \end{cases}
$$

$$
{}_l[\mathbf{loopstart}\,\phi]^i_k = \begin{cases} \bot & \text{if } l = - \\ {}_l[\phi]^l_k & \text{if } l \in \{0\,..\,k-1\} \end{cases}
$$

where $l \in \{0\,..\,k-1\}$ for the loop case and $l = -$ for the prefix case. The translation function for formulae ψ in the normal form described at the end of the last section is:

$$
\mathrm{Norm}[\hat{M}, \psi]_k \doteq \exists \overline{z}.\ [\hat{M}]_k \ \wedge\ [R]^0_k \ \wedge\ \textstyle\bigwedge_{i=0}^{k-2} [X]^i_k \ \wedge\ \bigwedge_{i=0}^{k-1} [X^*]^i_k \ \wedge
$$

$$
\left(\left([LS]^0_k \ \wedge\ [X]^{k-1}_k\right) \ \vee\ \textstyle\bigvee_{l=0}^{k-1} \left({}_lL_k(\hat{M}) \ \wedge\ {}_l[LS]^0_k \ \wedge\ {}_l[X]^{k-1}_k\right) \right)
$$

where $[\hat{M}]_k$ and ${}_lL_k(\hat{M})$ are as defined in Section 3 and the vector of propositional variables $\overline{z}$ contains variables $\alpha_0, \ldots, \alpha_{k-1}$ for each timed variable α in $\overline{\alpha}$ and a variable x for each untimed variable x in $\overline{x}$. The resulting formula has size linear in k, $|\phi|$ and $|\hat{M}|$. More precisely, its size is $O(|\hat{I}| + k \cdot (|\phi| + |\hat{T}|))$.

6 Related work

6.1 Helsinki work

The BMC translations closest to ours are those of [7] and [6]. These translations are also linear in k and they exploit fixpoint characterisations of operators. A core observation in [7] from the viewpoint of this paper is that the loop-case denotations of the LTL operators **F**, **G**, **U** and **R** are all equivalent to the restriction to bound k of the denotation of non-looping versions of the operators at bound $k + (k-l)-1$. For example:

$$
{}_l\overset{\mathrm{F}}{[\![\mathbf{G}]\!]}_k(\dot{a}) = \left({}_l\overset{\mathrm{F}}{[\![\tilde{\mathbf{G}}^\top]\!]}_k(\dot{a}\uparrow^{k+(k-l)-1}_\circ)\right)|_k
$$

where $\dot{a} \in \mathbb{B}^k$, $l \in \{0 \mathinner{..} k-1\}$, $\dot{a}\uparrow_\circ^{k'} \doteq \dot{a}\uparrow_\circ^\infty|_{k'}$ unrolls a loop denotation to bound k' and $\tilde{\mathbf{G}}^\top$ is as defined in Section 4.4. The justification in [7] for these identities is rather indirect and involves appealing to arguments about fixpoints in CTL. However, we note that we can prove them straightforwardly using some of the same insights as are necessary to prove the identity

$$\dot{\pi}\uparrow_\circ^\infty [\![\phi]\!] \;=\; \overset{\mathrm{F}}{\dot{\pi}}_l [\![\phi]\!]_k \uparrow_\circ^\infty$$

introduced in Section 2.4 which is at the heart of the justification of the original bounded model checking translation of [2].

A major apparent difference is that the approach in [7] introduces very few auxiliary variables by encoding to reduced Boolean circuits (RBCs), a DAG representation of Boolean formulae. However, when these circuits are subsequently translated into CNF, auxiliary variables are introduced for many of the internal nodes of the circuits, and we guess that one gets roughly one new auxiliary variable per fixpoint step, the same as what we use.

Comparing the sizes of resulting propositional formulae in the approach of [7] to ours, we observe that our encoding for **G** and **R** involves unrolling the fixpoint functions for k rather than $2k$ steps, and so involves introducing about half the number of $\wedge$s and $\vee$s. For **F** and **U** the number of $\wedge$s and $\vee$s introduced appears to be more similar, though for **F** we introduce roughly half the number of auxiliary variables into the final CNF formulae.

The approach of more recent work [6] from the same group is more similar to an automata-based approach in that the fixpoint constraints on auxiliary variables in the loop case also have a loop shape. Ignoring the incremental and past-time aspects of [6], the numbers of operators and auxiliary variables introduced seem to be slightly closer to those with our approach.

Experimentation and more detailed analysis are needed to sharpen the above preliminary remarks and importantly to compare how the approaches affect SAT run times.

6.2 Other work

The BMC journal paper [1] gives a translation exploiting fixpoint characterisations, though the encoding size is not linear in the bound. As written, the translation is not sound: it appears to be using a greatest fixpoint characterisation for *all* the LTL operators which is clearly unsound for **F** and **U**. We speculate that this mistake could have been avoided if the translation had been derived within a formal framework such as presented in this paper.

We observe that a recent NuSMV release (V2.3.1, Nov 2005) seems to use a similar translation that is sound. This translation has some similarities to that of [7] discussed above in Section 6.1 in that the loop case translation is calculated using non-looping fixpoint constraints.

7 Conclusions

We have presented a translation for future time LTL bounded model checking that is linear in the bound k and more compact than competing translations, in particular that of [7].

We have also presented a rigorous framework for analysing translations. Both the body of the paper and the discussion of related work show the usefulness of the framework, and it is expected that it will be of significant use in exploring future variations on and extensions to BMC translations.

References

[1] Biere, A., A. Cimatti, E. M. Clarke, O. Strichman and Y. Zhu, *Bounded model checking*, Advances in Computers **58** (2003).

[2] Biere, A., A. Cimatti, E. M. Clarke and Y. Zhu, *Symbolic model checking without BDDs*, in: W. Cleaveland, editor, *Tools and Algorithms for the Construction and Analysis of Systems. 5th International Conference, TACAS 99*, Lecture Notes in Computer Science **1579** (1999), pp. 193–207.

[3] Cimatti, A., M. Roveri and D. Sheridan, *Bounded verification of past LTL*, in: A. J. Hu and A. K. Martin, editors, *Proceedings of the 5th International Conference on Formal Methods in Computer Aided Design (FMCAD 2004)*, Lecture Notes in Computer Science (2004).

[4] Fisher, M., *A resolution method for temporal logic*, in: *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)* (1991).

[5] Frisch, A., D. Sheridan and T. Walsh, *A fixpoint based encoding for bounded model checking*, in: M. D. Aagaard and J. W. O'Leary, editors, *Formal Methods in Computer-Aided Design; 4th International Conference, FMCAD 2002*, Lecture Notes in Computer Science **2517** (2002), pp. 238–254.

[6] Heljanko, K., T. Junttila and T. Latvala, *Incremental and complete bounded model checkinfg for full pltl*, in: K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification: 17th International Conference, CAV 2005*, Lecture Notes in Computer Science **3576** (2005), pp. 98–111.

[7] Latvala, T., A. Biere, K. Heljanko and T. Junttila, *Simple bounded LTL model checking*, in: *Formal Methods in Computer-Aided Design; 5th International Conference, FMCAD 2004*, Lecture Notes in Computer Science **3312** (2004), pp. 186–200.

[8] Sheridan, D., *Bounded model checking with SNF, alternating automata and Büchi automata*, in: *Second International Workshop on Bounded Model Checking*, Electronic Notes in Theoretical Computer Science (2004).