Regular Model Checking for S1S + LTL

Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, Julien d'Orso, and Mayank Saksena

Dept. of Information Technology, P.O. Box 337, S-751 05 Uppsala, Sweden
{parosh,bengt,marcusn,juldor,mayanks}@it.uu.se

Abstract. Regular model checking is a form of symbolic model checking for parameterized and infinite-state systems whose states can be represented as words of arbitrary length over a finite alphabet, in which regular sets of words are used to represent sets of states. We present LTL(S1S), a combination of the logics S1S and LTL as an ideal logic for expressing temporal properties to be verified in regular model checking. LTL(S1S) is a two-dimensional modal logic, where S1S is used for specifying properties of system states and transitions, and LTL is used for specifying temporal properties. In addition, the first-order quantification in S1S can be used to express properties parameterized on a position or process. We give a technique for model checking LTL(S1S), which is adapted from the automata-theoretic approach: a formula is translated to a (Büchi) transducer with a regular set of accepting states, and regular model checking techniques are used to search for models. We have implemented the technique and show its application to a number of parameterized algorithms from the literature.

1 Introduction

Regular model checking is a framework for algorithmic symbolic verification of parameterized and infinite-state systems [BLW03,KMM$^+$01,WB98,BJNT00]. It considers systems whose states can be represented as finite words of arbitrary length over a finite alphabet, including array or ring-formed parameterized systems with an arbitrary number of finite-state processes, and systems that operate on queues, stacks, integers, and other linear unbounded data structures. In a system description, the set of initial states is represented as a regular set of strings, and the transition relation is given as a finite regular length-preserving transducer. Previous work on regular model checking [JN00,BJNT00,AJNd02] has developed methods for computing the set of reachable states of a system description, as well as the set of reachable loops, obtained from the transitive closure of the transition relation. In general, this problem is undecidable, but decidable for certain classes [JN00].

The techniques for computing reachable states and transitive closures can in principle be used to verify both safety and liveness properties of parameterized system descriptions, but do not provide a convenient approach for checking arbitrary temporal logic properties, comparable to the automata-theoretic approach for propositional model checking [VW86]. Additional transformations must be performed manually in order to transform the temporal property into a property of transitive closures and reachable states. Concrete problems that arise are the following.

(a) **Computing arbitrary compositions of relations.** In previous work, safety properties are verified as follows. The transition relation is given as a union of a finite set $\mathcal{A}$ of *actions* where each action $\alpha \in \mathcal{A}$ is given as a regular length-preserving transducer. For each action $\alpha \in \mathcal{A}$, its transitive closure α^+ is computed, and their union $\bigcup_{\alpha \in \mathcal{A}} \alpha^+$ is combined with the initial condition I to compute the set Inv of reachable states. This approach has several shortcomings. First, the transitive closure α^+ of an action $\alpha \in \mathcal{A}$ need not to be regular even though $(\alpha \ \cap \ Inv)^+$ is regular. Secondly, the verification becomes dependent on how the actions are chosen. Sometimes, it succeeds if we compute $(\alpha_1 \cup \alpha_2)^+$ and not just α_1^+ and α_2^+ separately. It would be desirable to directly compute the regular set or relation denoted by an arbitrary relational composition, e.g., $I \circ (\bigcup_{\alpha \in \mathcal{A}} \alpha)^*$ for the set of reachable states, so that e.g., constraints given by I can be exploited when computing the transitive closure.

(b) **Handling temporal properties.** A natural approach for verifying linear-time temporal properties is to adapt the automata-theoretic approach [VW86]. The negation of the temporal property is represented by an adapted Büchi automaton, which we here call a *Büchi transducer*, consisting of a regular set I of initial states, a regular length-preserving transducer T, and a regular set F of accepting states. Accepting runs of the Büchi transducer are infinite sequences of words, where the first word is in I, consecutive words satisfy T, and infinitely many words are in F. Since T is length-preserving, the existence of an accepting run can be checked by searching for a reachable loop which contains a state in F. It would be desirable to have an automated translation from temporal properties to Büchi transducers. If we have a solution to the previous Problem (a), then we can furthermore directly compute accepting loops, e.g., from the composition $F \circ T^* \circ F$.

(c) **Handling parameterization.** In the verification of parameterized systems, one is interested in stating temporal properties that are parameterized on a specific position in the string that models the state vector. For a parameterized mutual exclusion algorithm, a typical property one would want to express is the following

 If there is weak fairness for all processes, then each process that is interested in entering its critical section will eventually do so.

 In some temporal logic notation, this could look something like

$$\forall i.[\Box\Diamond(\text{blocked}(i) \vee \text{progressing}(i))] \longrightarrow \forall i.[\Box(\text{trying}(i) \rightarrow \Diamond\text{critical}(i))]$$

 where i ranges over positions in the string, and each position represents a process. If the number of processes is fixed, the quantifiers can be replaced by explicit conjunctions to obtain a standard model checking problem in propositional temporal logic. However, in regular model checking the number of processes is arbitrary, and so we must use a slightly more involved construction. Previous work [BJNT00,PS00] presents a manual construction for representing the above formula by a number of "local" Büchi automata, which are added to the local state of each process i, and for combining these into a global Büchi transducer. However, this approach is quite tedious and error-prone. It is highly desirable to develop an approach for performing regular model checking directly on temporal logic formulas with parameterization.

In this paper, we address Problems (b) and (c) by presenting a generic approach to regular model checking of properties expressed in a combination of the logics S1S and LTL. We use S1S for specifying sets of states and transition relations and LTL for specifying temporal constraints. The result is a two-dimensional modal logic, where S1S is used in the "space" (system state) dimension and LTL is used in the "time" dimension. Models of the logic are infinite sequences of (constant-length) words, representing computations of the specified system. By simply combining S1S with LTL, we get the possibility to express parameterized properties for free: S1S offers variables to represent positions and quantify over them, which can be interleaved with temporal operators. A parameterized temporal property such as the one in Problem (c) above is expressed by applying temporal operators to formulas with free S1S variables, and later using S1S quantification to express existential or universal quantification. In our logic LTL(S1S) temporal operators can be applied to formulas with at most one free first-order (S1S) variable and no free second-order variables. This restriction allows to express parameterized temporal properties (e.g., fairness constraints) of individual processes in a parameterized system, as well as temporal properties of pairs of adjacent processes (in positions i and $i+1$ using one free variable i).

We have implemented a tool for checking formulas in LTL(S1S) by implementing the translation from LTL(S1S) to Büchi transducers, and by extending our tool for regular model checking [AJNd03] to check whether Büchi transducers have accepting runs, thereby addressing Problem (a) above. This is done in two steps. First, the set of reachable states are computed as $Inv = I \circ T^*$. Secondly, loops are found by identifying identical pairs in $(F \cap T \cap Inv) \circ T^*$. In comparison with the ad hoc approach to verification of temporal logic properties outlined in [BJNT00] which builds on computation of the transitive closure T^+ of the transducer relation, this computation is much more efficient. We have verified safety properties with the tool for many of the examples in our previous work, as well as liveness properties for some of the examples. As a result from using the entire transition relation instead of transitive closures of individual actions, verification of some properties takes longer time, but does not suffer from Problem (a).

As special cases, when the formula contains no temporal operators, our method specializes into a decision procedure for S1S similar to that of MONA

[HJJ+96], and when the formula contains no quantifiers our method specializes to ordinary LTL model checking.

The remainder of the paper is structured as follows. In the next section, we present the logic $LTL(S1S)$. Section 3 illustrates how it can be used to model and specify parameterized algorithms. The model checking technique, including the translation to Büchi transducer is presented and proven correct in Section 4. New techniques and evaluations of our extended implementation are presented in Section 5.

Related Work There is a large body of research on the problem of model checking parameterized systems of *identical* processes, in which there is no ordering between processes, and hence the system state can be represented as a multiset of process states (e.g., [BLS01,Del00,EK03,EK00,GS92]). This problem is substantially simpler, since ordering between processes need not be considered. Emerson and Namjoshi [EN95] give a technique for verifying a restricted class of parameterized token-passing algorithms by reducing an arbitrary ring to a small fixed-size ring under certain conditions.

The problem of checking liveness properties of array-shaped parameterized systems was considered by Pnueli and Shahar [PS00], who presented a technique for computing the transitive closure of a restricted class of transition relations. They also first manually employ abstractions to make the implementation terminate.

Pnueli, Xu, and Zuck [PXZ02] present an interesting use of specialized abstractions in order to prove absense of starvation properties for Szymanski's algorithm and the Bakery algorithm. The abstractions keep track of the number of processes with certain properties, and generate a finite-state system, which can be model-checked. The presented abstraction is specialized to prove non-starvation, and loses much information so that e.g., safety properties can no longer be checked.

2 The logic LTL(S1S)

We present the syntax and semantics of $LTL(S1S)$. This logic combines the operators of S1S and LTL, under the restriction that temporal operators can be applied to formulas with at most one free first-order (S1S) variable and no free second-order variables. In this paper, we restrict the presentation to the temporal operators $\Box$ (always) and $\Diamond$ (eventually), but this is not a fundamental restriction.

Syntax We denote *first-order variables* Var_{FO} by lowercase letters $i, j, k, \ldots$, *second-order variables* Var_{SO} by uppercase letters $I, J, K, \ldots$ and *configuration variables* V by lowercase letters $x, y, z, \ldots$

$$
\begin{array}{ll}
\varphi ::= i \in I \mid I \subseteq J \mid i = j+1 \mid x[i] \mid x'[i] & \text{Atomic S1S formulas} \\
\quad\; true \mid false & \text{Boolean constants} \\
\quad\; \varphi \sim \varphi,\ \sim \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \mid \neg\varphi & \text{Propositional connectives} \\
\quad\; \exists i\varphi \mid \forall i\varphi \mid \exists I\varphi \mid \forall I\varphi & \text{S1S Quantification} \\
\quad\; \Box\varphi \mid \Diamond\varphi & \text{Temporal operators}
\end{array}
$$

No negation should have any temporal operator within its scope, which is why we provide dual versions of each operator. We further require that in every subformula of the form $\Box\varphi$ or $\Diamond\varphi$, there is at most one free first-order variable (and no free second-order variables). More operators on first-order variables, such as $<$, $\leq$, $=$, can be expressed in S1S using standard encodings.

Semantics Closed LTL(S1S) formulas are interpreted over *matrices* M over 2^V of dimension $\infty \times n$ where $n \in \mathbf{N}$ is a parameter that can take an arbitrary value. We call the vertical (first) dimension *time*, and the horizontal (second) dimension *space*. Let $\mathbf{Z}_n = \{0, 1, \ldots, n-1\}$. The element $M(t, i) \subseteq 2^V$ for $t \in \mathbf{N}$ and $i \in \mathbf{Z}_n$ represents the system configuration at time t of position (or subsystem) i, which assigns truth values to the configuration variables V. Thus, each row $M(t, 0) \ \cdots \ M(t, n)$ represents the system configuration at time t.

In general, a formula φ depends on its free first- and second-order variables, which are interpreted in the space dimension, and on a time point t. A model $\mathcal{M}$ for an arbitrary $LTL(S1S)$-formula ϕ is a triple $(M, \mathcal{I}, t)$, where $M : (\mathbf{N} \times \mathbf{Z}_n) \mapsto 2^V$ is a matrix over 2^V, where $\mathcal{I}$ is a valuation $\mathcal{I}$: $\mathrm{Var}_{FO} \rightarrow \mathbf{N}$ and $\mathcal{I}$: $\mathrm{Var}_{SO} \rightarrow 2^{\mathbf{N}}$ of first-order and second-order variables, and where t is a time point $t \in \mathbf{N}$. Satisfaction of formulas is defined as follows.

$$
\begin{array}{lll}
(M, \mathcal{I}, t) \not\models false & & \\
(M, \mathcal{I}, t) \models true & & \\
 & i \in I & \text{if } \mathcal{I}(i) \in \mathcal{I}(I) \\
 & I \subseteq J & \text{if } \mathcal{I}(I) \subseteq \mathcal{I}(J) \\
 & i = j+1 & \text{if } \mathcal{I}(i) = \mathcal{I}(j) + 1 \\
 & x[i] & \text{if } x \in M(t, \mathcal{I}(i)) \\
 & x'[i] & \text{if } x \in M(t+1, \mathcal{I}(i)) \\
 & \varphi_1 \ \vee \ \varphi_2 & \text{if } (M, \mathcal{I}, t) \models \varphi_1 \text{ or } (M, \mathcal{I}, t) \models \varphi_2 \\
 & \neg\varphi & \text{if } (M, \mathcal{I}, t) \not\models \varphi \\
 & \exists i\varphi & \text{if } \exists m \in \mathbf{Z}_n \ (M, \mathcal{I}[i \mapsto m], t) \models \varphi \\
 & \forall i\varphi & \text{if } \forall m \in \mathbf{Z}_n \ (M, \mathcal{I}[i \mapsto m], t) \models \varphi \\
 & \exists I\varphi & \text{if } \exists S \in 2^{\mathbf{Z}_n} \ (M, \mathcal{I}[I \mapsto S], t) \models \varphi \\
 & \forall I\varphi & \text{if } \forall S \in 2^{\mathbf{Z}_n} \ (M, \mathcal{I}[I \mapsto S], t) \models \varphi \\
 & \Box\varphi & \text{if } \forall t' \geq t \ (M, \mathcal{I}, t') \models \varphi \\
 & \Diamond\varphi & \text{if } \exists t' \geq t \ (M, \mathcal{I}, t') \models \varphi
\end{array}
$$

where $\mathcal{I}[i \mapsto m]$ is the valuation which sends i to m and otherwise acts as $\mathcal{I}$, and $\mathcal{I}[I \mapsto S]$ is defined analogously.

3 Modeling in $LTL(S1S)$

In this section, we discuss how to generate system models and set up a verification problem using $LTL(S1S)$.

A set of initial states is given by a formula φ_I without temporal operators. A formula φ_T without temporal operators, depending only on the current and the next timepoint, is used to represent a transition relation. $\Box\varphi_T$ then states that the transition relation should hold for each timepoint t. A *property* is given as a formula φ. To prove that the property φ holds for the system, we search for models of the formula

$$\varphi_I \ \wedge \ \Box\varphi_T \ \wedge \ \neg\varphi$$

A safety property is of the form $\Box\varphi_{safe}$, where φ_{safe} contains no temporal operators.

To express liveness properties we need fairness. Process fairness can be expressed as

$$\varphi_{fair} = \forall i \Box\Diamond(\mathcal{A}(i) \ \vee \ \neg\text{enabled}(\mathcal{A}(i)))$$

where $\mathcal{A}(i)$ represents all actions of process i, and enabled$(\mathcal{A}(i))$ represents the set of states where the action $\mathcal{A}(i)$ can be taken, obtained using an existential quantification of the primed configuration variables in $\mathcal{A}(i)$. To verify that φ_{live} holds under the fairness condition φ_{fair}, we use as the property φ the formula $\varphi_{fair} \longrightarrow \varphi_{live}$, i.e. we search for models of the formula

$$\varphi_I \ \wedge \ \Box\varphi_T \ \wedge \ \neg(\varphi_{fair} \longrightarrow \varphi_{live})$$

Example To illustrate modeling in $LTL(S1S)$, we model the Bakery algorithm for mutual exclusion in Fig. 1. In the algorithm, there are an arbitrary number of processes waiting to get a "ticket" to get into the critical section. Each process which wants to get into the critical section receives a ticket which is the maximum of all the outstanding tickets plus one. When a process has the lowest outstanding ticket, it enters the critical section and drops the ticket when leaving. Each process i has an integer-valued variable $ticket_i$, whose value is initially 0.

Idle:	$ticket_i := 1 + \max_j ticket_q$
Waiting:	await $\forall j \neq i : (ticket_i < ticket_j \ \vee \ ticket_j = 0)$
Critical:	$ticket_i := 0$

Fig. 1. Bakery Algorithm

In our model in $LTL(S1S)$ of the Bakery algorithm, we use a configuration variable q to model the states of the processes. For each i, the value of $q[i]$ is

- $\perp$ if there is no process that has ticket $i+1$, and
- W if some process with ticket $i+1$ is waiting, and
- C if some process with ticket $i+1$ is in critical

Our model of Bakery is as follows. The formulas **safety** and **liveness** are used to verify safety (mutual exclusion) and liveness (non-starvation) respectively.

$$
\begin{array}{lcl}
\mathbf{ticket}(i) & = & q[i] = \perp \ \wedge q'[i] = W \ \wedge \ \forall j > i \ q[j] = \perp \ \wedge \\
 & & i \neq 0 \longrightarrow q[i-1] \neq \perp \\
\mathbf{enter}(i) & = & q[i] = W \ \wedge \ q'[i] = C \ \wedge \ \forall j < i \ q[j] = \perp \\
\mathbf{exit}(i) & = & q[i] = C \ \wedge \ q'[i] = \perp \\
\mathbf{copy}(i) & = & q[i] = q'[i] \\
\mathbf{copy\text{-}all} & = & \forall i \ \mathbf{copy}(i) \\
\mathbf{a}(i) & = & (\mathbf{ticket}(\mathrm{i}) \ \vee \ \mathbf{enter}(i) \ \vee \ \mathbf{exit}(i)) \ \wedge \ \forall j \neq i \ \mathbf{copy}(j) \\
\mathbf{initial} & = & \forall i \ q[i] = \perp \\
\mathbf{sys} & = & \mathbf{initial} \ \wedge \ \Box(\exists i \ \mathbf{a}(i) \ \vee \ \mathbf{copy\text{-}all}) \\
\mathbf{fairness} & = & \forall i \Box\Diamond(\mathbf{a}(i) \ \vee \ \neg\mathrm{enabled}(\mathbf{a}(i))) \\
\mathbf{safety} & = & \mathbf{sys} \ \wedge \ \neg\Box\neg(\exists i \exists j \ i \neq j \ \wedge \ q[i] = C \ \wedge \ q[j] = C) \\
\mathbf{liveness} & = & \mathbf{sys} \ \wedge \ \neg(\mathbf{fairness} \longrightarrow \forall i \Box \ (q[i] = W \longrightarrow \Diamond q[i] = C))
\end{array}
$$

4 Translating $LTL(S1S)$ to Büchi Transducers

In this section, we describe how to transform a formula in $LTL(S1S)$ into a Büchi transducer, consisting of a regular set I of initial states, a regular length-preserving transducer T, and a regular set F of accepting states. In this section, we will represent such a Büchi transducer by the $LTL(S1S)$ formula $\phi_I \wedge \Box\phi_T \wedge \Box\Diamond\phi_F$, where ϕ_I and ϕ_F are S1S formulas denoting the regular sets of initial and accepting states, and ϕ_T is an S1S formula over primed and unprimed configuration variables denoting the regular transition relation. The accepting runs of the Büchi transducer accept exactly the models of the temporal formula, so in the following we will consider $\phi_I \wedge \Box\phi_T \wedge \Box\Diamond\phi_F$ to be a Büchi transducer where I, T, and F are regular transducers expressed in S1S.

The idea of the construction is similar to that of the standard translation of propositional temporal logic to Büchi Automata [VW86]: the semantics of temporal operators is translated to additional state information in the Büchi automaton. In our case the operators are translated to new configuration variables that represent the values of certain temporal subformulas of ϕ. The semantics of these subformulas is represented by constraints on the values of the new configuration variables.

We assume that ϕ is in negative normal form. Define a *core subformula* of ϕ as a subformula of ϕ which has a temporal operator as its main connective. For each core subformula ψ of ϕ, we introduce an auxiliary configuration variable x_ψ. Intuitively, the value of x_ψ at time point t and space position i should represent the value of ψ at that entry in the matrix. For each core subformula $\Diamond\psi$ where the main connective is $\Diamond$ we introduce an auxiliary configuration variable $y_{\Diamond\psi}$ (called an *eventuality variable*). Intuitively, if the variable $y_{\Diamond\psi}$ is true, then the formula ψ should be true at some future time point.

Using the auxiliary configuration variables, the value of any subformula ψ can be represented by a formula in S1S over the extended set of configuration variables. Let us define this formula $\langle\langle\psi\rangle\rangle$ as follows:

$$\begin{array}{lcl}
\langle\langle\phi\rangle\rangle & = & \phi \text{ for } \phi \text{ in S1S} \\
\langle\langle\psi_1 \wedge \psi_2\rangle\rangle & = & \langle\langle\psi_1\rangle\rangle \wedge \langle\langle\psi_2\rangle\rangle \\
\langle\langle\psi_1 \vee \psi_2\rangle\rangle & = & \langle\langle\psi_1\rangle\rangle \vee \langle\langle\psi_2\rangle\rangle \\
\langle\langle\exists i\psi\rangle\rangle & = & \exists i\langle\langle\psi\rangle\rangle \\
\langle\langle\forall i\psi\rangle\rangle & = & \forall i\langle\langle\psi\rangle\rangle \\
\langle\langle\exists I\psi\rangle\rangle & = & \exists I\langle\langle\psi\rangle\rangle \\
\langle\langle\forall I\psi\rangle\rangle & = & \forall I\langle\langle\psi\rangle\rangle \\
\langle\langle\Box\psi\rangle\rangle & = & x_{\Box\psi} \\
\langle\langle\Diamond\psi\rangle\rangle & = & x_{\Diamond\psi}
\end{array}$$

Let $\textbf{localconstr}(\phi)$ define the conjunction of a set of constraints on the auxiliary variables, called *local constraints*, as defined below.

(a) For each auxiliary variable of form x_ψ, where i is the (possibly) free first-order variable in ψ, define its constraint as follows.

$$\begin{array}{ll}
\forall i.\left(x_{\Box\psi_1}[i] \leftrightarrow \langle\langle\psi_1\rangle\rangle[i] \wedge x'_{\Box\psi_1}[i]\right) & \quad \text{when } \psi \text{ is } \Box\psi_1 \\
\forall i.\left(x_{\Diamond\psi_1}[i] \leftrightarrow \langle\langle\psi_1\rangle\rangle[i] \vee x'_{\Diamond\psi_1}[i]\right) & \quad \text{when } \psi \text{ is } \Diamond\psi_1
\end{array}$$

(b) Let $y_{\Diamond\psi_1}, \ldots, y_{\Diamond\psi_k}$ be the set of eventuality variables. We define their local constraint as:

$$\bigwedge_{m=1}^{k} \forall i.\left(y_{\Diamond\psi_m}[i] \;\wedge\; \neg y'_{\Diamond\psi_m}[i] \longrightarrow \langle\langle\psi_m\rangle\rangle[i]\right)$$

Furthermore, we require that all eventuality variables are false infinitely often and that they load new eventualities: let $\textbf{evconstr}(\phi)$ define the conjunction of a set of constraints on the auxiliary variables, called *eventuality constraint*, defined below.

$$\bigwedge_{m=1}^{k} \forall i.\left(\neg y_{\Diamond\psi_m}[i] \;\wedge\; (y'_{\Diamond\psi_m}[i] \leftrightarrow x'_{\Diamond\psi_m}[i])\right)$$

The goal of this section is to prove that for each formula ϕ there is a model M such that $(M, \mathcal{I}, t) \models \phi$ iff there is a model M' such that $(M', \mathcal{I}, t) \models \langle\langle\phi\rangle\rangle \;\wedge\; \Box\textbf{localconstr}(\phi) \;\wedge\; \Box\Diamond\textbf{evconstr}(\phi)$ where M' differs from M only on the auxiliary variables. By its form, the latter formula can be regarded as a Büchi transducer, and so we have translated ϕ into an equivalent Büchi transducer. We first prove the following lemma.

Lemma 1. *If* $(M', \mathcal{I}, t) \models \Box\textbf{localconstr}(\phi) \;\wedge\; \Box\Diamond\textbf{evconstr}(\phi)$*, then for all auxiliary variables of* ϕ *we have*

(a) $(M', \mathcal{I}, t) \models \forall i.x_{\Box\psi}[i] \longrightarrow \Box\langle\langle\psi\rangle\rangle$

(b) $(M',\mathcal{I},t) \models \forall i.x_{\diamond\psi}[i] \longrightarrow \diamond\langle\langle\psi\rangle\rangle \ \vee \ \diamond y_{\diamond\psi}[i]$
(c) $(M',\mathcal{I},t) \models \forall i.y_{\diamond\psi}[i] \longrightarrow \diamond\langle\langle\psi\rangle\rangle$

Proof. (a) Let $(M',\mathcal{I}',t) \models x_{\Box\psi}[i]$ for some $\mathcal{I}' = \mathcal{I}[i \mapsto m]$. Since $(M',\mathcal{I},t) \models \Box\mathbf{localconstr}(\phi)$, we have

$$(M',\mathcal{I}',t) \models \Box\left(x_{\Box\psi}[i] \leftrightarrow \langle\langle\psi\rangle\rangle[i] \wedge x'_{\Box\psi}[i]\right)$$

It follows that $(M',\mathcal{I}',t') \models \langle\langle\psi\rangle\rangle[i]$ for every $t' \geq t$.

(b) Let $(M',\mathcal{I}',t) \models x_{\diamond\psi}[i]$ for some $\mathcal{I}' = \mathcal{I}[i \mapsto m]$. Since $(M',\mathcal{I},t) \models \Box\mathbf{constr}(\phi) \wedge \Box\diamond\mathbf{evconstr}(\phi)$, we have
- $(M',\mathcal{I}',t) \models \Box\left(x_{\diamond\psi}[i] \leftrightarrow \langle\langle\psi\rangle\rangle[i] \vee x'_{\diamond\psi}[i]\right)$.
- $(M',\mathcal{I}',t') \models y'_{\diamond\psi}[i] \leftrightarrow x'_{\diamond\psi}[i]$ for some $t' \geq t$

If $(M',\mathcal{I}',t'') \models \langle\langle\psi(i)\rangle\rangle$ for some $t'' \geq t$ we are done, suppose that no such t'' exists. Then it follows that $(M',\mathcal{I}',t') \models y'_{\diamond\psi}[i]$ and thus $(M',\mathcal{I}',t'+1) \models y_{\diamond\psi}[i]$.

(c) Let $(M',\mathcal{I}',t) \models y_{\diamond\psi}[i]$ for some $\mathcal{I}' = \mathcal{I}[i \mapsto m]$, and let $t' > t$ be the earliest point in time when

$$(M',\mathcal{I}',t') \models \neg y_{\diamond\psi}[i]$$

. Then we have

$$(M',\mathcal{I}',t'-1) \models y_{\diamond\psi}[i] \ \wedge \ \neg y'_{\diamond\psi}[i]$$

and by the eventuality constraint of $y_{\diamond\psi}$ we get

$$(M',\mathcal{I}',t'-1) \models \langle\langle\psi\rangle\rangle(i)$$

Since $t'-1 \geq t$ we conclude that

$$(M',\mathcal{I}',t) \models \diamond\langle\langle\psi(i)\rangle\rangle$$

Theorem 1. *For each subformula ψ of ϕ, interpretation I and timepoint t we have there is a model M such that $(M,\mathcal{I},t) \models \psi$ iff there is a model M' such that $(M',\mathcal{I},t) \models \langle\langle\psi\rangle\rangle \ \wedge \ \Box\mathbf{localconstr}(\phi) \ \wedge \ \Box\diamond\mathbf{evconstr}(\phi)$ where M differs from M' only in the auxiliary variables of ϕ.*

Proof. $\Longrightarrow$: This case is rather straight-forward. Define $M^{\phi}(i)$ by letting $x_{\psi(i)}$ be true at point $M^{\phi}(t,i)$ iff $(M,\mathcal{I},t) \models \psi(i)$. Then the local constraints follow by the definitions of the temporal operators. The values of variables $y_{\diamond\psi(i)}$ are then defined in a way that satisfies the eventuality constraint:

- at time t, all variables $y_{\diamond\psi}[i]$ are set to false;
- if all $y_{\diamond\psi}[i]$ are false at time t', then set the value of each $y_{\diamond\psi}[i]$ at time $t'+1$ to the value of corresponding $x_{\diamond\psi}[i]$ at time $t'+1$;
- if variable $y_{\diamond\psi}[i]$ is true at time t', and $\langle\langle\psi(i)\rangle\rangle$ holds, then set $y_{\diamond\psi}[i]$ to false at time $t'+1$.

$\Longleftarrow$: Let $(M',\mathcal{I},t) \models \langle\langle\psi\rangle\rangle \ \wedge \ \Box\mathbf{localconstr}(\phi) \ \wedge \ \Box\diamond\mathbf{evconstr}(\phi)$. We prove by induction over the structure of ψ.

$\psi \in S1S$: Since $\langle\!\langle\psi\rangle\!\rangle = \psi$, we get $(M', \mathcal{I}, t) \models \psi$.

$\psi = \exists i\psi_0$: We get $(M', \mathcal{I}, t) \models \exists i\langle\!\langle\psi_0\rangle\!\rangle$ and by semantics $(M', \mathcal{I}[i \mapsto m], t) \models \langle\!\langle\psi_0\rangle\!\rangle$ for some $m \in \mathbf{Z}$. Since $\Box\mathbf{localconstr}(\phi) \wedge \Box\Diamond\mathbf{evconstr}(\phi)$ is a closed formula and thus does not depend on i, it follows that $(M', \mathcal{I}[i \mapsto m], t) \models \Box\mathbf{localconstr}(\phi) \;\wedge\; \Box\Diamond\mathbf{evconstr}(\phi)$. By induction there is some M that differs from M' only in the auxiliary variables of ψ_0 such that $(M, \mathcal{I}[i \mapsto m], t) \models \psi_0$. By semantics we get $(M, \mathcal{I}, t) \models \exists i\psi_0$.

$\psi = \{\exists I\psi_0, \forall i\psi_0, \forall I\psi_0\}$ Similar to $\psi = \exists i\psi_0$.

$\psi = \Box\psi_0$: We get $(M', \mathcal{I}, t) \models x_{\Box\psi_0}$. By Lemma 1 it follows that $(M', \mathcal{I}, t) \models \Box\langle\!\langle\psi_0\rangle\!\rangle$. Then by semantics for all $t' \geq t$, we have $(M', \mathcal{I}, t') \models \langle\!\langle\psi_0\rangle\!\rangle$ and then by induction there is some M that differs from M' only in the auxiliary variables of ψ_0 such that $(M, \mathcal{I}, t') \models \psi_0$. Since this holds for all $t' \geq t$ we have by semantics that $(M, \mathcal{I}, t) \models \Box\psi_0$.

$\psi = \Diamond\psi_0$: We get $(M', \mathcal{I}, t) \models x_{\Diamond\psi_0}$. By Lemma 1 it follows that $(M', \mathcal{I}, t) \models \Diamond\langle\!\langle\psi_0\rangle\!\rangle \;\vee\; \Diamond y_{\Diamond\psi_0}$. We have $(M', \mathcal{I}, t') \models \langle\!\langle\psi_0\rangle\!\rangle$ for some $t' \geq t$ in both cases as follows. In the case $(M', \mathcal{I}, t) \models \Diamond y_{\Diamond\psi_0}$ holds, it follows by Lemma 1, and in the case $(M', \mathcal{I}, t) \models \Diamond\langle\!\langle\psi_0\rangle\!\rangle$ holds it follows from semantics. Then by induction there is some M that differs from M' only in the auxiliary variables of ψ_0 such that $(M, \mathcal{I}, t') \models \psi_0$. Since $t' \geq t$ we have by semantics that $(M, \mathcal{I}, t) \models \Diamond\psi_0$. $\Box$

Corollary 1. *For a formula ϕ, interpretation I and timepoint t we have that there is a model M such that $(M, \mathcal{I}, t) \models \phi$ iff there is a model M' such that $(M', \mathcal{I}, t) \models \langle\!\langle\phi\rangle\!\rangle \;\wedge\; \Box\mathbf{localconstr}(\phi) \;\wedge\; \Box\Diamond\mathbf{evconstr}(\phi)$ where M differs from M' only in the auxiliary variables of ϕ.*

5 Implementation

The transformation of the previous section results in a formula of the form

$$I \;\wedge\; \Box T \;\wedge\; \Box\Diamond F$$

where the formulas I, T, and F are in S1S and can be translated into transducers. The transducers can be seen as binary relations on words that are closed under standard operations such as union, intersection and composition. For the sake of presentation, we will assume that I only contains non-primed configuration variables and look at I as a set instead of a binary relation. The general case does not introduce any significant additional problems.

In previous work we have developed algorithms for computing the transitive closure T^+ for a transducer T. We have generalized the algorithm given in [AJNd02] to handle regular expressions over transducers, where concatenation means composition and Kleene star means reflexive and transitive closure. In particular, given two transducers T_1 and T_2 the algorithm can now handle $T_1 \circ T_2^*$. This operation is used to find models as follows.

(a) Compute the set of reachable states $Inv = I \circ T^*$.

(b) Define T_{Inv} as $\{(w, w') \in T \ : \ w \in Inv\}$ and compute the set of loops by finding identical pairs in $(T_{Inv} \ \cap \ F) \circ T^*$

We use an optimized version of the above scheme as follows. When computing the invariant $I \circ T^*$ the transition relation T contains local constraints for eventuality variables. For eventualities that hold infinitely often, such local constraints are only needed when checking for loops through F and not when computing the set of reachable states. To simplify the transition relation used when computing the set of reachable states, the formula is transformed into the following form:

$$I \ \wedge \ \Box T_0 \ \wedge \ \Diamond(F \ \wedge \ \Box T \ \wedge \ \Box\Diamond F)$$

where T_0 is like T but without the local constraints on eventuality variables for eventualities that hold infinitely often. The final procedure is then as follows.

(a) Compute the set of reachable states $Inv = I \circ T_0^*$.
(b) As before, define T_{Inv} as $\{(w, w') \in T \ : \ w \in Inv\}$ and compute the set of loops by finding identical pairs in $(T_{Inv} \ \cap \ F) \circ T^*$

We have successfully verified safety properties for many of the examples in our previous work, as well as liveness properties for some of the smaller examples. Execution times are given in the table below.

Algorithm	Safety (ms)	Liveness (ms)
Tokenpass	3930	14790
Tokenring	4840	29590
Bakery	4840	23070
Burns	38560	
Dijkstra	593220	
Szymanski	1073200	

We experience high execution times for checking safety properties for some of the algorithms. This is because we use the entire transition as opposed to computing the transitive closure of separate actions and using them for reachability analysis. This reachability analysis does not always terminate, however, since some algorithms require the transitive closure of combinations of actions for the analysis to terminate. Using the entire transition relation is a more general technique, but potentially more costly.

References

[AJNd02] Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Julien d'Orso. Regular model checking made simple and efficient. In *Proc. CONCUR 2002, 13^{th} Int. Conf. on Concurrency Theory*, volume 2421 of *Lecture Notes in Computer Science*, pages 116–130, 2002.

[AJNd03] Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Julien d'Orso. Algorithmic improvements in regular model checking. In *Proc. 14^{th} Int. Conf. on Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, 2003.

[BJNT00] A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In Emerson and Sistla, editors, *Proc. 12^{th} Int. Conf. on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 403–418. Springer Verlag, 2000.

[BLS01] K. Baukus, Y. Lakhnech, and K. Stahl. Verification of parameterized networks. *Journal of Universal Computer Science*, 7(2), 2001.

[BLW03] Bernard Boigelot, Axel Legay, and Pierre Wolper. Iterating transducers in the large. In *Proc. 14^{th} Int. Conf. on Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 223–235, 2003.

[Del00] G. Delzanno. Automatic verification of cache coherence protocols. In Emerson and Sistla, editors, *Proc. 12^{th} Int. Conf. on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 53–68. Springer Verlag, 2000.

[EK00] E.A. Emerson and V. Kahlon. Reducing model checking of the many to the few. In *Proc. 17th International Conference on Automated Deduction*, volume 1831 of *Lecture Notes in Computer Science*, pages 236–254. Springer Verlag, 2000.

[EK03] E.A. Emerson and V. Kahlon. Rapid parameterized model checking of snoopy cache coherence protocols. In *Proc. TACAS '03, 9^{th} Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, 2003.

[EN95] E.A. Emerson and K.S. Namjoshi. Reasoning about rings. In *Proc. 22^{th} ACM Symp. on Principles of Programming Languages*, 1995.

[GS92] S. M. German and A. P. Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.

[HJJ+96] J.G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, B. Paige, T. Rauhe, and A. Sandholm. Mona: Monadic second-order logic in practice. In *Proc. TACAS '95, 1^{th} Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1019 of *Lecture Notes in Computer Science*, 1996.

[JN00] Bengt Jonsson and Marcus Nilsson. Transitive closures of regular relations for verifying infinite-state systems. In S. Graf and M. Schwartzbach, editors, *Proc. TACAS '00, 6^{th} Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1785 of *Lecture Notes in Computer Science*, 2000.

[KMM+01] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. *Theoretical Computer Science*, 256:93–112, 2001.

[PS00] A. Pnueli and E. Shahar. Liveness and acceleration in parameterized verification. In *Proc. 12^{th} Int. Conf. on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 328–343. Springer Verlag, 2000.

[PXZ02] A. Pnueli, J. Xu, and L. Zuck. Liveness with $(0, 1, \infty)$-counter abstraction. In Brinskma and Larsen, editors, *Proc. 14^{th} Int. Conf. on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 107–122. Springer Verlag, 2002.

[VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. LICS '86, 1^{st} IEEE Int. Symp. on Logic in Computer Science*, pages 332–344, June 1986.

[WB98] Pierre Wolper and Bernard Boigelot. Verifying systems with infinite but regular state spaces. In *Proc. 10th Int. Conf. on Computer Aided Verifi-*

cation, volume 1427 of *Lecture Notes in Computer Science*, pages 88–97, Vancouver, July 1998. Springer Verlag.