Algorithmic Improvements in Regular Model Checking ⋆

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

Uppsala University, Dept. of Information Technology
P.O. Box 337, S-751 05 Uppsala, Sweden
{parosh,bengt,marcusn,juldor}@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 finite strings of arbitrary length over a finite alphabet, in which regular sets of words are used to represent sets of states. In earlier papers, we have developed methods for computing the transitive closure (or the set of reachable states) of the transition relation, represented by a regular length-preserving transducer. In this paper, we present several improvements of these techniques, which reduce the size of intermediate approximations of the transitive closure: One improvement is to pre-process the transducer by *bi-determinization*, another is to use a more powerful equivalence relation for identifying histories (columns) of states in the transitive closure. We also present a simplified theoretical framework for showing soundness of the optimization, which is based on commuting simulations. The techniques have been implemented, and we report the speedups obtained from the respective optimizations.

1 Introduction

Regular model checking has been proposed as a uniform paradigm for algorithmic verification of parameterized and infinite-state systems [KMM+01,WB98,BJNT00]. In regular model checking, states are represented as finite strings over a finite alphabet, while regular sets of words are used as a symbolic representation of sets of states. Furthermore, regular length-preserving transducers represent transition relations. A generic task in regular model checking is to compute a representation for the set of reachable states, or of the transitive closure of the transition relation. Since we are dealing with an infinite state space, standard iteration-based methods for computing transitive closures (e.g., [BCMD92]) are not guaranteed to terminate.

In previous work [JN00,BJNT00,AJNd02], we have developed methods for computing the transitive closure (or the set of reachable states) of a transducer,

⋆ This work was supported in part by the European Commission (FET project ADVANCE, contract No IST-1999-29082), and by the the Swedish Research Council (http://www.vr.se/)

⋆⋆ This author is supported in part by ARTES, the Swedish network for real-time research (http://www.artes.uu.se/).

which are complete for transition relations that satisfy a condition of *bounded local depth* (this can be seen as a non-trivial generalization, to the case of transducers, of the condition of "bounded-reversal" for Turing machines [Iba78]).

These techniques (and those by Dams et al. [DLS01]) input a transducer T, and attempt to construct a finite representation of the union $T^* = \cup_{i=0}^{\infty} T^i$ of the finite compositions T^i of T. The states of T^i can be seen as "histories" (which we will call *columns*) of length i of transducer states. There are infinitely many such columns in T^*, and the challenge is to find a finite-state automaton which is equivalent to T^*. In [BJNT00], we presented a technique which directly determinizes T^* by the subset construction, where subsets are represented as regular sets of columns. In addition, subsets are identified using a pre-defined equivalence relation; this ensures termination if T has bounded local depth. In [AJNd02], we presented a more light-weight technique, which successively computes the approximations $T^{\leq n} = \cup_{i=1}^{n} T^i$ for $n = 1, 2, \ldots$, while quotienting the set of columns with a certain equivalence relation. For implementation, this technique represented a substantial simplification over the heavy automata-theoretic constructions of [BJNT00], and gave significant performance improvements on most of the examples that we have considered.

In this paper, we present several improvements to the techniques of [AJNd02]. Here, we describe the most important improvements (illustrated in an example in the next section):

– *Bi-determinization*: A key element of our techniques [JN00,BJNT00,AJNd02] is the equivalence relation used to identify columns of T^*. Ideally, this equivalence should be as large as possible while still respecting the constraint that the quotient of T^* is equivalent to T^*. Our equivalences are based on so-called *left-* or *right-copying* states. Roughly, a path in the transducer from an initial to a left-copying state, or from a right-copying to a final state, may only copy symbols. Borrowing intuition from rewriting theory, the copying states of a transducer define the "context" for the actual transformation of the word. The equivalence relation in [AJNd02] is based on ignoring the number of successive repetitions of the same left- or right-copying state in columns, but does not cope with sequences of *distinct* left-copying (or right-copying) states. To overcome this, we now pre-process the transducer by *bi-determinization* before the actual computation of the transitive closure. The effect of bi-determinization is that columns with successive distinct left-copying (or right-copying) states can be safely ignored, since they will never occur in any accepting run of T^*.
– *Coarser equivalence relations*: In addition to adding the bi-determinization phase, we have increased the power of the equivalence relations in [AJNd02]: whereas before we identified columns with one or more repetitions of a left- or right-copying state, we can now in certain contexts also identify columns with zero or more such repetitions, and identify states with k alternations between left- and right-copying states with states that have 3 alternations, if $k > 3$.

– *Improved theoretical framework*: The equivalences are proven sound by an improved theoretical framework, based on simulations which are also rewrite relations. This framework is a generalization and simplification of the work by Dams et al. [DLS01], who use bisimulations. However, the bisimulation-based framework can not prove that the reduction of alternation (to at most 3) is sound.

We have implemented these optimizations, and tested them on a number of parameterized mutual exclusion protocols. The performance improvement is at least an order of magnitude.

Related Work Regular model checking was advocated by Kesten et al. [KMM+01], and implemented in the Mona [HJJ+96] package. Techniques for accelerating regular model checking have been considered in our earlier work [JN00,BJNT00,AJNd02].

Dams et al. [DLS01] present a related approach, introducing a generic framework for quotienting of the transducer T^* to find a finite-state representation, and which is not limited to length-preserving transductions. Dams et al. find such an equivalence by a global analysis of the current approximation of the transitive closure. It appears that this calculation is very expensive, and the paper does not report successful experimental application of the techniques to examples of similar complexity as in our work. In this paper, we present a generalization of the equivalences defined by Dams et al.

Touili [Tou01] presents a technique for computing transitive closures of regular transducers based on *widening*, and shows that the method is sufficiently powerful to simulate earlier constructions described in [ABJN99] and [BMT01].

The works in [AJMd02,BT02] extend regular model checking to the context of tree transducers. When applied on words, these techniques correspond to the acceleration and widening techniques described in [BJNT00] and therefore do not cover the optimizations presented in this paper.

Outline In the next section, we illustrate the problem and the main ideas through an example. In Section 3, we present a general framework to derive equivalence relations which are language preserving under quotienting. We use this framework in Section 4 to define a new equivalence relation. Finally, in Section 6, we report experimental results of an implementation based on the new techniques.

2 An Example

In this section, we will present and illustrate the problem and the techniques through a token passing system with a ring topology.

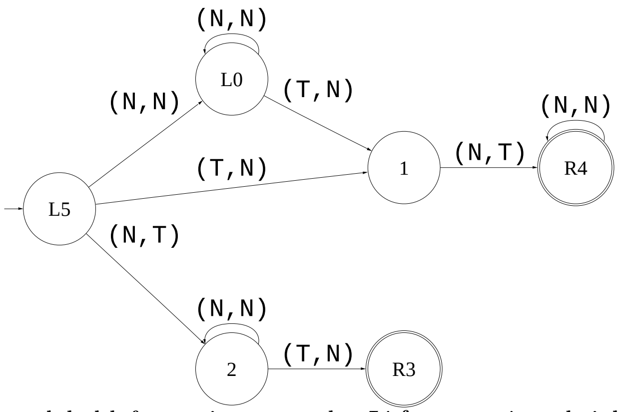
Preliminaries Let Σ be a finite alphabet of symbols. Let R be a regular relation on Σ, represented by a finite-state *transducer* $T = \langle Q, q_0, \longrightarrow, F \rangle$ where Q is the (finite) set of states, q_0 is the initial state, $\longrightarrow \subseteq Q \times (\Sigma \times \Sigma) \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. Note that T is not required to be deterministic. We use $q_1 \xrightarrow{(a,b)} q_2$ to denote that $(q_1, (a,b), q_2) \in \longrightarrow$. We use a similar infix notation also for the other types of transition relations introduced later in the paper.

The set of *left-copying* states in Q is the largest subset Q_L of Q such that whenever $q \xrightarrow{(a,a')} q'$ and $q' \in Q_L$, then $a = a'$ and $q \in Q_L$. Analogously, the set of *right-copying* states in Q is the largest subset Q_R of Q such that whenever $q \xrightarrow{(a,a')} q'$ and $q \in Q_R$, then $a = a'$ and $q' \in Q_R$. Intuitively, prefixes of left-copying states only copy input symbols to output symbols, and similarly for suffixes of right-copying states. We shall assume that $Q_L \cap Q_R = \emptyset$ (if not, we can simply decrease Q_L and Q_R to satisfy the assumption).

Example As an example, we use a token ring passing protocol. In the system, there is an arbitrary number of processes in a ring, where a token is passed in one direction. In our framework, a *configuration* of the system is represented by a finite word $a_1 a_2 \cdots a_n$ over the alphabet $\Sigma = \{N, T\}$ where each a_i represents:

- T - Process number i has a token.
- N - Process number i does not have a token.

The transition relation of this system is represented by the transducer below. For example, there is an accepting run $L5 \xrightarrow{(N,N)} L0 \xrightarrow{(T,N)} 1 \xrightarrow{(N,T)} R4 \xrightarrow{(N,N)} R4$ accepting the word $(N,N)(T,N)(N,T)(N,N)$. This run means that from the configuration $NTNN$ we can get to the configuration $NNTN$ in one transition, i.e. process 2 passes the token to process 3.



Note that we label left-copying states by Li for some i, and right-copying states by Ri for some i.

Column transducer Our goal is to construct a transducer that recognizes the relation R^*, where $R^* = \cup_{i \geq 0} R^i$.

Starting from T, we can in a straight-forward way construct (see also [BJNT00]) a transducer for R^* whose states, called *columns*, are sequences of states in Q, where runs of transitions between columns of length i accept pairs of words in R^i. More precisely, define the *column transducer* for T as the tuple $T^* = \langle Q^*, q_0^*, \Longrightarrow, F^* \rangle$ where

- Q^* is the set of sequences of states of T,
- q_0^* is the set of sequences of the initial state of T,
- $\Longrightarrow \subseteq (Q^* \times (\Sigma \times \Sigma)) \times Q^*$ is defined as follows: for any columns $q_1 q_2 \cdots q_m$ and $r_1 r_2 \cdots r_m$, and pair (a, a'), we have $q_1 q_2 \cdots q_m \overset{(a,a')}{\Longrightarrow} r_1 r_2 \cdots r_m$ iff there are $a_0, a_1, \ldots, a_m$ with $a = a_0$ and $a' = a_m$ such that $q_i \overset{(a_{i-1},a_i)}{\longrightarrow} r_i$ for $1 \leq i \leq m$,
- F^* is the set of sequences of accepting states of T.

It is easy to see that T^* accepts exactly the relation R^*: runs of transitions from q_0^i to columns in F^i accept transductions in R^i. The problem is that T^* has infinitely many states. Our approach is to define an *equivalence* $\simeq$ between columns of the column transducer and construct the column transducer with equivalent states merged. Under some conditions, the set of equivalence classes we construct is finite. More precisely, given an equivalence relation $\simeq$, the *quotient transducer* $T_\simeq$ is defined as $T_\simeq = \langle Q^*/\simeq, \{q_0\}^*/\simeq, \Longrightarrow_\simeq, F_\simeq \rangle$ where

- $Q^*/\simeq$ is the set of equivalence classes of columns,
- $q_0^*/\simeq$ is the partitioning of q_0^* with respect to $\simeq$, where we assume that q_0^* is a union of equivalence classes.
- $\Longrightarrow_\simeq \subseteq (Q^*/\simeq) \times (\Sigma \times \Sigma) \times (Q^*/\simeq)$ is defined in the natural way as follows. For any columns x, x' and symbols a, a':

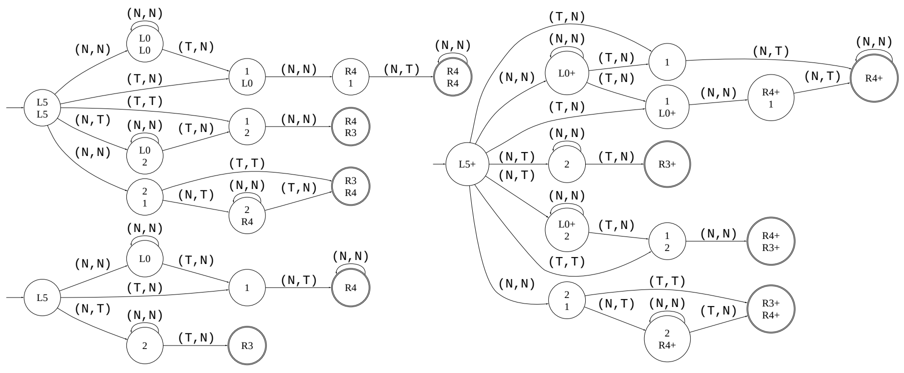
$$x \overset{(a,a')}{\Longrightarrow} x' \quad \Rightarrow \quad [x]_\simeq \overset{(a,a')}{\Longrightarrow}_\simeq [x']_\simeq$$

- $F_\simeq$ is the set of equivalence classes in $Q^*/\simeq$ that have a non-empty intersection with F^*.

Example (ctd.) The part of the column transducer for the token ring system, consisting of states labeled by columns of length one or two, is shown in Figure 1(a). The equivalence $\simeq$ used in [AJNd02] ignores successive repetitions of the same left-copying and right-copying state, e.g., the column $R4\ R3$ belongs to the equivalence class $R4^+\ R3^+$, and the column $1\ L0$ belongs to the equivalence class $1\ L0^+$. The quotient of the transducer in Figure 1(a) under $\simeq$ is shown in Figure 1(b).

In this paper, we improve on the previous results in the following ways.

Bi-determinization In our previous work, the equivalence relation handled the case of repeating the same left-copying or right-copying state, but did not handle sequences of distinct left-copying states or distinct right-copying states. For example, in Figure 1(b), we note that the equivalence classes $R4^+\ R3^+$ and $R3^+\ R4^+$ are disjoint, although they are "equivalent", and intuitively should be merged.



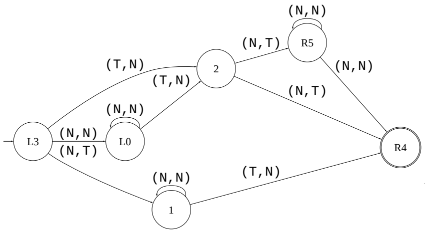
(a) $T^{1,2}$: Part of the column transducer (b) $T^{1,2}$ under the old equivalence

Fig. 1. Part of column transducer under old equivalence

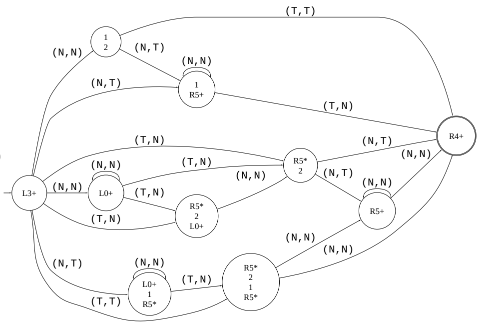
Rather than working out an equivalence relation that identifies suitable sequences of distinct copying states, we solve this problem by preprocessing the transducer by *bi-determinization*. Bi-determinization produces a transducer where the sub-automaton consisting only of the left-copying states is deterministic, and the sub-automaton consisting only of the right-copying states is reverse deterministic. Bi-determinization can be done by a reverse subset construction on the right-copying states. For a bi-deterministic transducer, columns with successive distinct left-copying states are not reachable since two different left-copying states will not have any common prefix. Analogously, columns with successive distinct right-copying states are not productive since two different right-copying states will not have any common suffix. In Figure 2(a), we show the result of bi-determinization of the token ring passing example.

A new coarser equivalence In Section 4, we will derive a new equivalence, with coarser equivalence classes. The part of the column transducer for the bi-determinized token ring system, consisting of states labeled by columns of length one or two, optimized with the new equivalence relation, is shown in Figure 2(b). There are two new features in this equivalence:

1. In certain contexts, equivalence classes can now ignore zero or more repetitions of the same copying state. For example, in Figure 2(b) we have the equivalence class $R5^*$ 2. Using the old equivalence relation (the one in [AJNd02]), we had to represent this equivalence class by the two classes 2 and $R5^+$ 2.
2. For any left-copying state L and right-copying state R, the column $R\ L\ R\ L\ R$ will be equivalent to $R\ L\ R$. In Figure 2(b), there are no such columns present. This becomes important, however, for higher-order approximations of the column transducer.



(a) Bi-deterministic version of token ring passing



(b) Bi-deterministic $T^{1,2}$ under new equivalence

Fig. 2. Part of column transducer under the new equivalence

3 Soundness

In this section, we present a general framework to derive equivalence relations which are language preserving under quotienting, and therefore, sufficient for proving soundness of our construction method.

We introduce finitely generated *rewrite relations* and *simulations* to obtain an equivalence relation which can be safely used to quotient the transducer under construction.

Simulations A relation $\preceq_F$ on the set Q^* of columns is a *forward simulation* if whenever $x \preceq_F x'$, then

- if $x \overset{(a,b)}{\Longrightarrow} y$ for some column y and symbols a, b, there is a column y' such that $x' \overset{(a,b)}{\Longrightarrow} y'$ and $y \preceq_F y'$, and
- if x is accepting, then x' is accepting.

Analogously, a relation $\preceq_B$ on the set of columns is a *backward simulation* if whenever $y \preceq_B y'$, then

- if $x \overset{(a,b)}{\Longrightarrow} y$ for some column x, and symbols a, b, there is a column x' such that $x' \overset{(a,b)}{\Longrightarrow} y'$ and $x \preceq_B x'$, and
- if y is initial, then y' is initial.

Rewritings We will work with simulation relations that are defined by rewrite relations on subcolumns. A *rewrite relation* $\mapsto$ is a reflexive and transitive relation on the set of columns, which is a congruence under composition. In other words, $x \mapsto y$ implies $v\ x\ w \ \ \mapsto v\ y\ w$ for any v, w. A *rewrite rule* is a pair (x, y) of columns. The rewrite relation $\mapsto$ is *generated* by a finite set $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ of rewrite rules if $\mapsto$ is the least rewrite relation such that $x_i \mapsto y_i$ for $i = 1, \ldots, m$. (This means that $\mapsto$ is the least reflexive and transitive relation such that $w\ x_i\ w' \mapsto w\ y_i\ w'$ for any columns w, w', and i).

We will use the fact that finite simulation relations can be extended to rewrite relations by making them congruences.

Lemma 1. *If $\leq_R$ is a finite forward simulation, then the rewrite relation generated by all pairs in $\leq_R$ is also a forward simulation. The analogous property holds for backward simulations.*

Using Lemma 1, we can assume that simulations are reflexive and transitive.

Two rewrite relations $\mapsto_L$ and $\mapsto_R$ satisfy the *diamond property* if whenever $x \mapsto_L y$ and $x \mapsto_R z$, then there is a w such that $y \mapsto_R w$ and $z \mapsto_L w$. For rewrite relations that are generated by sets of rewrite rules, the diamond property can be checked efficiently by inspecting all possible critical pairs, i.e., the possible overlaps of left-hand sides of rewrite rules generating $\mapsto_L$ with left-hand sides of rewrite rules generating $\mapsto_R$ (cf. [KB70]).

Lemma 2. *A pair of rewrite relations $(\mapsto_L, \mapsto_R)$ satisfying the diamond property* induces *the equivalence relation $\simeq$ defined by*

$x \simeq y$ if and only if there are
- *z such that $x \mapsto_L z$ and $y \mapsto_R z$, and*
- *z' such that $y \mapsto_L z'$ and $x \mapsto_R z'$.*

Lemma 3. *Let $\mapsto_L$ and $\mapsto_R$ be two rewrite relations that satisfy the diamond property. Let $\simeq$ be the equivalence relation they induce.*
Then, whenever $x \simeq y$ and $x \mapsto_R x'$, there exists y' such that $y \mapsto_R y'$ and $x' \mapsto_L y'$.

Proof. To see this, assume that $x \simeq y$ and $x \mapsto_R x'$. This means that there is a z such that $x \mapsto_L z$ and $y \mapsto_R z$. By the diamond property there is a y' such that $x' \mapsto_L y'$ and $z \mapsto_R y'$. By the transitivity of $\mapsto_R$ we infer (from $y \mapsto_R z$ and $z \mapsto_R y'$) that $y \mapsto_R y'$. □

Theorem 1. *Let $\mapsto_L$ and $\mapsto_R$ be two rewrite relations that satisfy the diamond property. Let $\simeq$ be the equivalence relation they induce. If in addition*

- *$\mapsto_R$ is a forward simulation,*
- *$\mapsto_L$ is included in a backward simulation $\preccurlyeq$,*

then $T_{\simeq}$ and T^* are equivalent.

Proof. Let

$$[x_0]_{\simeq} \xrightarrow{(a_1,b_1)} [x_1]_{\simeq} \xrightarrow{(a_2,b_2)} \cdots \xrightarrow{(a_n,b_n)} [x_n]_{\simeq}$$

be a run of $T_{\simeq}$. This means that for each i with $1 \leq i \leq n$ there are x_{i-1} and y_i such that $x_{i-1} \xrightarrow{(a_i,b_i)} y_i$ and $x_i \simeq y_i$. By induction for $i = 0, 1, 2, \ldots, n$, we find x_i' and y_i' such that $x_i' \xrightarrow{(a_{i+1},b_{i+1})} y_{i+1}'$ with $x_i \mapsto_R x_i'$ and $y_i' \preccurlyeq x_i'$, as follows:

- for $i = 0$ we take $x'_0 = y'_0 = x_0$,
- for $i = 1, \ldots, n$ we infer from $x_{i-1} \mapsto_R x'_{i-1}$ that y'_i exists such that $x'_{i-1} \xrightarrow{(a_i,b_i)} y'_i$, and $y_i \mapsto_R y'_i$, from which we use Lemma 3 to infer that x'_i exists with $x_i \mapsto_R x'_i$ and $y'_i \mapsto_L x'_i$; by inclusion, we get $y'_i \preccurlyeq x'_i$.

We can now by induction for $i = n, n-1, \ldots, 1$ find z_i such that $z_{i-1} \xrightarrow{(a_i,b_i)} z_i$, and $y'_i \preccurlyeq z_i$, as follows:

- for $i = n$ we take $z_n = x'_n$
- for $i = n-1, \ldots, 0$ we infer from $y'_{i+1} \preccurlyeq z_{i+1}$ and $x'_i \xrightarrow{(a_{i+1},b_{i+1})} y'_{i+1}$, that z_i exists such that $z_i \xrightarrow{(a_{i+1},b_{i+1})} z_{i+1}$, and $y'_i \preccurlyeq x'_i \preccurlyeq z_i$.

Since x_0 by definition is initial and also $x_0 = y'_0 \preccurlyeq z_0$, we conclude that z_0 is an initial state. Similarly, because x_n can be chosen to be accepting (there is one such representative in $[x_n]_\simeq$) and $x_n \mapsto_R x'_n = z_n$, we conclude that z_n is accepting.

Therefore, there is a run in T^* of form

$$z_0 \xrightarrow{(a_1,b_1)} z_1 \xrightarrow{(a_2,b_2)} \cdots \xrightarrow{(a_n,b_n)} z_n$$

□

4 A coarse equivalence

In this section, we derive an equivalence relation using the framework of the previous section.

Let the input transducer T be the tuple $\langle Q, q_0, \longrightarrow, F\rangle$, where $Q_L \subseteq Q$ is the set of left-copying states, $Q_R \subseteq Q$ is the set of right-copying states, and $Q_N = Q \setminus (Q_L \cup Q_R)$ is the set of non-copying states. We shall assume that $Q_R \cap Q_L = \emptyset$. For a set Q_0, let Q_0^ϵ denote $\{\epsilon\} \cup Q_0$.

Without loss of generality, we can assume that T^* does not contain any columns with alternations of distinct left- or right-copying states:

Lemma 4. *Columns with two consecutive and distinct left- or right-copying states can be removed without changing the language of T^*.*

Proof. A column with two distinct left-copying states is never reachable in T^* since the left-copying part of T is deterministic. Similarly, a column with two distinct right-copying states is never productive since the right-copying part is reverse-deterministic. □

We proceed by defining rewrite relations $\mapsto_R$ and $\mapsto_L$.

1. Let $\mapsto_R$ be the rewrite relation generated by the set of rules of the form (q_R, ϵ) and $(q_R, q_R\, q_R)$, where q_R is a right-copying state.

2. (a) Let $\preccurlyeq$ be the maximal backward simulation on the states of T^*. This simulation will contain at least the following:
   - $q_L \preccurlyeq \epsilon$ for any left-copying state q_L;
   - $q_L \preccurlyeq q_L\ q_L$, for any left-copying state q_L.

   (b) For columns $x, y \in Q_L^\epsilon$, and z_1, z_2, let $x \sim_{z_1,z_2} y$ denote that $z_1\ x\ z_2 \preccurlyeq z_1\ y\ z_2$ and $z_1\ y\ z_2 \preccurlyeq z_1\ x\ z_2$, in other words, x and y simulate each other w.r.t. $\preccurlyeq$ in the context of z_1 and z_2. Note that $\sim_{z_1,z_2}$ is an equivalence relation on Q_L^ϵ for any z_1, z_2.

   (c) We define $\mapsto_L$ to be the rewrite relation generated by the rules
   - (q_L, ϵ) and $(q_L, q_L\ q_L)$ for any left-copying state q_L,
   - $(z_1\ x\ z_2, z_1\ y\ z_2)$ for any $x, y \in Q_L^\epsilon$ and $z_1, z_2 \in Q_N^\epsilon$ such that $x \sim_{z_1,z_2} y$.

The following Theorem shows that the rewrite relations induce an equivalence relation with the properties of Theorem 1.

Theorem 2. *1. $\mapsto_R$ is a forward simulation.*
2. $\mapsto_L$ is included in the backward simulation $\preccurlyeq$.
3. $\mapsto_L$ and $\mapsto_R$ have the diamond property.

Proof. 1. Follows from the fact that q_R is right-copying.
2. Follows from the fact that rules of $\mapsto_L$ are taken from $\preccurlyeq$.
3. Let $x \mapsto_L y$ and $x \mapsto_R z$. Then $x = x_1 q_R x_2$ and $z = x_1 z' x_2$ for $z' \in \{\epsilon, q_R\ q_R\}$. Since the left hand side of each rule of $\mapsto_L$ does not contain any state from Q_R, we conclude that $y = y_1 q_R y_2$ where $x_1 \mapsto_L y_1$ and $x_2 = y_2$, or $x_2 \mapsto_L y_2$ and $x_1 = y_1$. In either case, $z = x_1 z' x_2 \mapsto_L y_1 z' y_2$. Furthermore, $y = y_1 q_R y_2 \mapsto_R y_1 z' y_2$, completing the diamond.

□

Corollary 1. *From Theorem 1 and Theorem 2 it follows that the relation $\simeq$ induced by $\mapsto_L$ and $\mapsto_R$ is an equivalence relation such that $T_\simeq$ and T^* are equivalent.*

5 Implementation of the equivalence relation

In the implementation, we have used an approximation of $\simeq$ which is a finer equivalence relation than $\simeq$. Each equivalence class in this approximation is of the form:

$$z_0\ e_0\ z_1\ e_1\ z_2 \cdots e_{n-1}\ z_n$$

where each $z_i \in Q_N^\epsilon$ and each e_i is of the form

$$f\ f_1 \cdots f_m\ f'$$

where

- f is of the form q_L^*, q_L^+, ϵ for some $q_L \in Q_L$ such that f is an equivalence class of $\sim_{z_i,\epsilon}$.

- f' is of the form q_L^*, q_L^+, ϵ for some $q_L \in Q_L$ such that f' is an equivalence class of $\sim_{\epsilon,z_{i+1}}$.
- If $m = 0$, then $f\ f'$ is also in an equivalence class of $\sim_{z_i,z_{i+1}}$.
- For $0 < j \leq m$, the equivalence class f_j is one of:
  - $q_R^+\ q_L^+\ (q_R^+\ q_L^+)^* q_R^+$ for some $q_L \in Q_L$, or
  - q_L^+ for some $q_L \in Q_L$, or
  - q_R^+ for some $q_R \in Q_R$.

For example, a typical equivalence class is $q_0\ q_L^*\ q_R^+\ q_1$, where $z_0 = q_0$, $z_1 = q_1$ and $e_0 = q_L^*\ q_R^+$. In this case $q_L \sim_{q_0,\epsilon} \epsilon$ which means that $q_0\ q_L$ simulates q_0 backward. In [AJNd02], we used an equivalence relation with equivalence classes q_L^+ for left-copying states q_L, and q_R^+ for right-copying states q_R.

In what follows, we explore properties of $\simeq$ to justify that equivalence classes of $\simeq$ is indeed union of equivalence classes of the form above. The following Lemma shows that it is enough to prove that for each i any two columns in $z_i\ e_i\ z_{i+1}$ are equivalent.

Lemma 5. *Let $z \in Q_N^\epsilon$ be a non-copying state or the empty column and x, x', y, y' be columns such that $x\ z \simeq y\ z$ and $z\ x' \simeq z\ y'$. Then $x\ z\ x' \simeq y\ z\ y'$.*

Proof. Let z_1, z_2 be such that $x\ z \mapsto_L z_1$, and $y\ z \mapsto_R z_1$, and $z\ x' \mapsto_L z_2$, and $z\ y' \mapsto_R z_2$. Since $z \in Q_N^\epsilon$ and $\mapsto_R$ do not contain states from Q_N in the left-hand side, $z_1 = z_1'\ z$ and $z_2 = z\ z_2'$ for some z_1', z_2'. Then, $x\ z\ x' \mapsto_L z_1'\ z\ x' \mapsto_L z_1'\ z\ z_2'$ and $y\ z\ y' \mapsto_R z_1'\ z\ y' \mapsto_R z_1'\ z\ z_2'$. □

Corollary 2. *Let*

$$x = z_0\ x_0\ z_1\ x_1 \cdots z_{n-1}\ x_{n-1}\ z_n$$

and

$$y = z_0\ y_0\ z_1\ y_1 \cdots z_{n-1}\ y_{n-1}\ z_n$$

be columns with $z_0, z_1, \ldots, z_n \in Q_N^\epsilon$ such that $z_i\ x_i\ z_{i+1} \simeq z_i\ y_i\ z_{i+1}$ for all i with $0 \leq i < n$. Then $x \simeq y$.

Proof. Induction on n. □

The following Lemma states that any two columns of one atomic part of an equivalence class are equivalent.

Lemma 6. *For columns $z, z' \in Q_N^\epsilon$, any left-copying state $q_L \in Q_L$ and right-copying state $q_R \in Q_R$, and columns x, x', y, y', the following hold:*

1. *If $x \sim_{z,z'} y$, then $z\ x\ z' \simeq z\ y\ z'$.*
2. *$q_R\ q_L\ q_R \simeq q_R\ q_L\ q_R\ q_L\ q_R$.*
3. *$q_L^i \simeq q_L^j$, for $i, j > 0$.*
4. *$q_R^i \simeq q_R^j$, for $i, j > 0$.*

Proof. 1. If $x \sim_{z,z'} y$, then $z\ x\ z' \mapsto_L z\ y\ z'$. By reflexivity, $z\ y\ z' \mapsto_R z\ y\ z'$.

2. Let $x = q_R\ q_L\ q_R$ and $y = q_R\ q_L\ q_R\ q_L\ q_R$. For $z = q_R\ q_L\ q_L\ q_R$, we have $x \mapsto_L z$ and $y \mapsto_R z$. For $z' = q_R\ q_R\ q_L\ q_R$, we have $x \mapsto_R z'$ and $y \mapsto_L z'$.

3. Since $i, j > 0$, we have $q_L^i \mapsto_L q_L^j$. By reflexivity, $q_L^j \mapsto_R q_L^j$.
4. Since $i, j > 0$, we have $q_R^i \mapsto_R q_R^j$. By reflexivity, $q_R^j \mapsto_L q_R^j$.

□

Finally, we conclude that any two columns in $z_i \; e_i \; z_{i+1}$ are equivalent.

Theorem 3. *Let*

$$x_0 = z_1 \; x \; x_1 \; x_2 \; \cdots x_m \; x' \; z_2$$

and

$$y_0 = z_1 \; y \; y_1 \; y_2 \; \cdots y_m \; y' \; z_2$$

be columns in the equivalence class

$$z_1 \; f \; f_1 \; f_2 \; \cdots f_m \; f' \; z_2$$

such that $x, y \in f$ *and* $x_i, y_i \in f_i$ *and* $x', y' \in f'$. *Then* $x_0 \simeq y_0$.

Proof. If $m = 0$ the result follows from Lemma 6. If $m > 0$, it follows from Lemma 6 that

- $z_1 \; x \simeq z_1 \; y$,
- $x_j \simeq y_j$ for all j with $1 \leq j \leq m$, and
- $x' \; z_2 \simeq y' \; z_2$.

Applying Lemma 5 (with $z = \epsilon$) gives us $x_0 \simeq y_0$. □

6 Implementation

We have implemented the equivalence defined in Section 4. We have run several examples to measure the effectiveness of the method, comparing the number of states built. At the end of this section, we mention some more technical improvements.

In our previous work[AJNd02] we have for each algorithm computed the transitive closures for individual *actions* representing one class of statements in the program. Here, we compute the transitive closure of the transducer representing the *entire program.*

We have compared the number of states generated under the following conditions:

- **Old equivalence** This is the old equivalence used in [AJNd02].
- **Bi-determinization** Using the old equivalence but on a bi-deterministic transducer.
- **New equivalence** Using the new equivalence on a bi-deterministic transducer.

The results can be found in Table 1. The computation time for Bakery is about two minutes with the new techniques, implying a tenfold performance improvement. To reduce the states of the transducer for Szymanski, it was intersected with an invariant while Bakery was not, hence the huge difference in the number of states. The invariant was computed using standard reachability analysis augmented with transitive closures of individual statements. Note that this can not be done in general. For example, computing the invariant this way does not work for Bakery.

Transducer / Method	Old equivalence	Bi-determinization	New equivalence
Token passing	6 (15)	6 (15)	4 (10)
Token ring passing	68 (246)	58 (230)	25 (86)
Bakery	1793 (5719)	605(1332)	335(813)
Szymanski	20 (47)	16(30)	11(22)

Table 1. Number of live (and total) number of states generated

Dead by label In the algorithm, we might generate some states that can be declared dead by only looking at the label of the state. For a bi-deterministic transducer, any label of the form $L1\ L2$ and $R1\ R2$ can be declared dead, see Lemma 4.

Caching Many transitions are merged, and many transitions have the same edge. Thus, caching the result of composing edges provides a substantial runtime improvement.

References

[ABJN99] Parosh Aziz Abdulla, Ahmed Bouajjani, Bengt Jonsson, and Marcus Nilsson. Handling global conditions in parameterized system verification. In *Proc. 11^{th} Int. Conf. on Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 134–145, 1999.

[AJMd02] Parosh Aziz Abdulla, Bengt Jonsson, Pritha Mahata, and Julien d'Orso. Regular tree model checking. In *Proc. 14^{th} Int. Conf. on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, 2002.

[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.

[BCMD92] J.R. Burch, E.M. Clarke, K.L. McMillan, and D.L. Dill. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98:142–170, 1992.

[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.

[BMT01] A. Bouajjani, A. Muscholl, and T. Touili. Permutation rewriting and algorithmic verification. In *Proc. LICS' 01* 17^{th} *IEEE Int. Symp. on Logic in Computer Science*. IEEE, 2001.

[BT02] Ahmed Bouajjani and Tayssir Touili. Extrapolating Tree Transformations. In *Proc.* 14^{th} *Int. Conf. on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, 2002.

[DLS01] D. Dams, Y. Lakhnech, and M. Steffen. Iterating transducers. In G. Berry, H. Comon, and A. Finkel, editors, *Computer Aided Verification*, volume 2102 of *Lecture Notes in Computer Science*, 2001.

[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.

[Iba78] O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25:116–133, 1978.

[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.

[KB70] D.E. Knuth and P.B. Bendix. Simple word problems in universal algebra. In *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon press, 1970.

[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.

[Tou01] T. Touili. Regular Model Checking using Widening Techniques. *Electronic Notes in Theoretical Computer Science*, 50(4), 2001. Proc. Workshop on Verification of Parametrized Systems (VEPAS'01), Crete, July, 2001.

[WB98] Pierre Wolper and Bernard Boigelot. Verifying systems with infinite but regular state spaces. In *Proc. 10th Int. Conf. on Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 88–97, Vancouver, July 1998. Springer Verlag.