Algorithmic Improvements in Regular Model Checking ⋆

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

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 is 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 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 of 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 of improvements to the techniques of [AJNd02], which reduces verification times in many examples by at least an order of magnitude. (* Was this accurate? *) We here briefly describe the most important improvements. They are illustrated on an example in the next section.

- *Bi-determinaztion*: 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 quotienting of T^* is equivalent to T^*. Our equivalences are 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, 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 trasitive 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^*.
- o*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 context 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 optimization, and tested them to compute the transitive closure of the transition relation of a number of parameterized mutual exclusion protocols from the literature. (* Now explain what were the improvements *)

– What is the average performance improvement on the standard examples?
– (* Say that we can do liveness, if this is correct. Say that it is different from the Israeli team results *)

Related Work. Regular model checking was advocated by Kesten et al. [KMM$^+$01], and implemented, in the the Mona [HJJ$^+$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, which present 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 present a simple example which we will use to illustrate our algorithm. In Section **??** we describe the algorithm for computing transitive closures. In Section **??**, we show soundness and completeness of the algorithm and present sufficient conditions for termination. Section **??** contains a description of an implementation of the algorithm and the result of applying it to a number of mutual exclusion protocols. Concluding remarks and directions for future research are given in Section **??**.

2 An Example

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

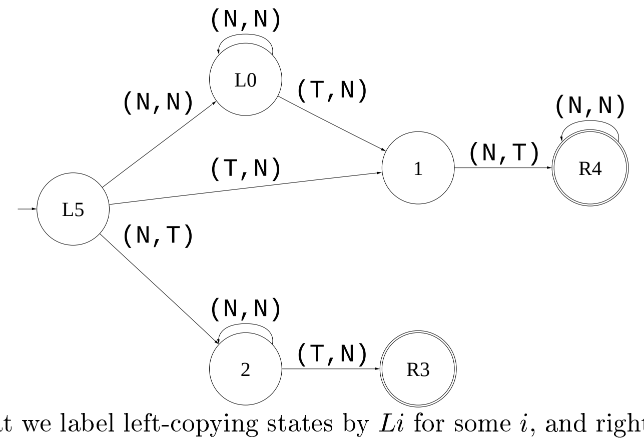
Let Σ be a finite alphabet of symbols. Let R be a regular relation on Σ, represented by a finite-state *transducer* $T = \langle Q, q_0, \longrightarrow, F\rangle$ where Q is the (finite) set of states, q_0 is the initial state, $\longrightarrow \subseteq Q \times (\Sigma \times \Sigma) \times Q$ is 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 are 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 $(N, N)(T, N)(N, T)(N, N)$ meaning 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 $q_1 q_2 \cdots q_m \stackrel{(a,a')}{\Longrightarrow} r_1 r_2 \cdots r_m$ iff there are $a_0, a_1, \ldots, a_m$ with $a = a_0$ and $a' = a_m$ such that $q_i \stackrel{(a_{i-1},a_i)}{\longrightarrow} r_i$ 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

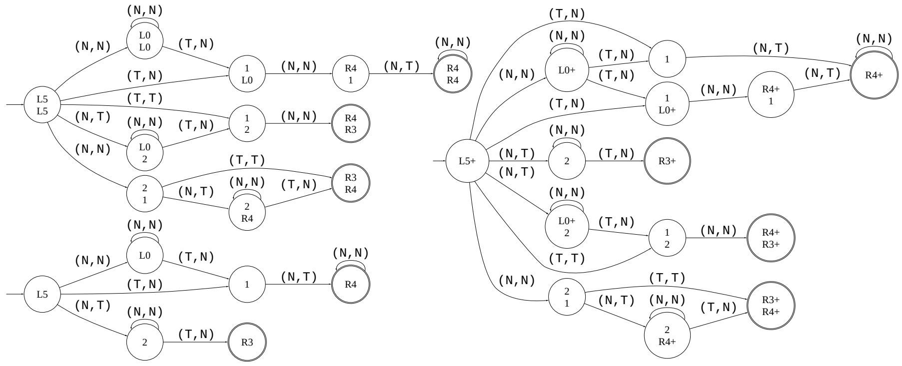
- $Q^*/\simeq$ is the set of equivalence classes of columns,
- $q_0^*/\simeq$ is the partitioning of q_0^* with respect to $\simeq$, where we assume that q_0^* is a union of equivalence classes.
- $\Longrightarrow_\simeq \subseteq ((Q^*/\simeq) \times (\Sigma \times \Sigma)) \times (Q^*/\simeq)$ is defined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

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

- $F^*/\simeq$ is the partitioning of F^* with respect to $\simeq$, where we assume that F^* is a union of equivalence classes.

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.



(a) $T^{1,2}$: Part of the column transducer (b) $T^{1,2}$ under LR-Equivalence

Fig. 1. Part of column transducer under old equivalence

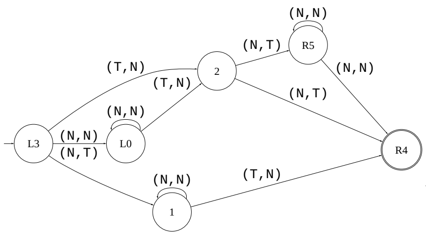
Bi-determinization In our previous work, the equivalence relation handled the case of repeating the same left-copying or right-copying state, but did not handle sequences of distinct left-copying states or distinct right-copying states. For example, in Figure 1(b), we note that the equivalence classes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

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 state will no common prefix. Analogously, columns with successive distinct right-copying states are not productive since two different right-copying state will no common suffix. In Figure 2(b), we show the result of biderministization of the token ring passing example.

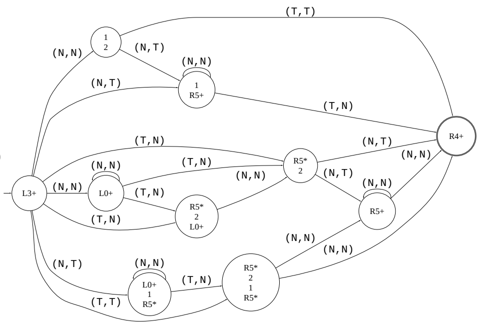
A new coarser equivalence In Section 4, we will derive a new equivalence which will allow us to obtain even coarser equivalence classes. The part of the column transducer for the bideterminized token ring system, consisting of states labeled by columns of length one or two, optimized with the new equivalence relation, is shown in shown in Figure 2(b). There are two new things 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, 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 higer-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 Principles

In this section, we describe a theoretical framework that allows to prove soundness of our constuction 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.*

Proof. By definition, simulations are preserved by adding reflexivity and transitivity. It is simple to see that whenever $x \ \leq_R \ y$, then for any columns w_1 and $w2$, it is the case that $w_1\ x\ w_2$ is simulated by $\ w_1\ y\ w_2$. Thus, a simulation is also preserved by extending it to a congruence. □

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

Two rewrite relations $\mapsto_L$ and $\mapsto_R$ have 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 is a z such that $x \mapsto_L z$ and $y \mapsto_R z$.

Proof. Symmetry and reflexivity are given by the definition. We have to check that $\simeq$ is transitive. Assume $x \simeq y \simeq z$. We let w and w' be columns such that $x \mapsto_L w$, $y \mapsto_R w$, $y \mapsto_L w'$, and $z \mapsto_R w'$. From the diamond property, we infer that there is a column w'' satisfying $w' \mapsto_R w''$ and $w \mapsto_L w''$. By transitivity of $\mapsto_L$ and $\mapsto_R$, we also have $x \mapsto_L w''$ and $z \mapsto_R w''$. We conclude that $x \simeq z$. □

Note that the definition is symmetric, i.e., if $x \simeq y$ then there is also a t such that $y \mapsto_L t$ and $x \mapsto_R t$.

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. We first claim that

– whenever $x \simeq y$ and $x \mapsto_R x'$,
  there exists y' such that $y \mapsto_R y'$ and $x' \preccurlyeq y'$.

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 $y \mapsto_R y'$. Since $\mapsto_L$ is included in $\preccurlyeq$, we have also $x' \preccurlyeq y'$.

Next, 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 the definition of covered, we can by induction for $i = 1, 2, \ldots, n$ find x'_i and y'_i such that $x'_{i-1} \xrightarrow{(a_i,b_i)} y'_i$ with $x_i \mapsto_R x'_i$ and $y'_i \preccurlyeq x'_i$, as follows:

– for $i = 1$ we take $y'_1 = y_1$,
– for $i = 2, \ldots, n$ 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 the definition of covered to infer that x'_i exists with $x_i \mapsto_R x'_i$ and $y'_i \preccurlyeq x'_i$.

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 $z_n = y'_n$
– 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 $y'_i \preccurlyeq x'_i \preccurlyeq z_i$.

We conclude that 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 coarser equivalence

In this section, we present the quotienting used in our implementation. It is derived from simulations pre-computed on the input transducer.

Let the input transducer T be the tuple $\langle Q, q_0, \longrightarrow, F \rangle$, where $Q_L \subset Q$ is the set of left-copying states and $Q_R \subset Q$ is the set of right-copying states. We shall assume that $Q_R \cap Q_L = \emptyset$.

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

Lemma 3. *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-coying states is never reachable in T^* since the left-copying part of T is deterministic. Similarly, a column with two distinct reght-copying states is never productive since the right-copying part is reverse-deterministic. □

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

Lemma 4. *The relation W_R is a forward simulation.*

Proof. We consider the pair $(q_R, q_R\ q_R) \in W_R$. We assume that q_R has a transition $q_R \xrightarrow{(a,b)} q_1$. Since q_R is right-copying, we have $a = b$ and $q_1 = q'_R$ is also right-copying. Then, $q_R\ q_R \xRightarrow{(a,a)} q'_R\ q'_R$ is a transition of T^*. Notice also that $(q'_R, q'_R\ q'_R) \in W_R$.

Consider the rule $(q_R, \epsilon) \in W_R$. Let $q_R \xRightarrow{(a,b)} q_1$ be a transition of T^*. Since q_R is right-copying, we have $a = b$ and $q_1 = q'_R \in Q_R$. Then, $\epsilon \xRightarrow{(a,a)} \epsilon$ is also a transition of T^*. We also observe that $(q'_R, \epsilon) \in W_R$. □

By applying lemma 1, we can conclude the following:

Corollary 1. *The rewrite relation $\mapsto_R$ is also a forward simulation.*

Let $\preccurlyeq$ be the maximal backward simulation on the states of T^*. This simulation will contain at least the following (by aa reasoning similar to that of lemma 4, these pairs form a simulation, and thus are included):

- $q_L \preccurlyeq \epsilon$ for any left-copying state q_L;
- $q_L \preccurlyeq q_L\ q_L$, for any left-copying state q_L.

For columns z_1, z_2, let $x \sim_{z_1,z_2} y$ denote that x and y simulate each other in the context of z_1 and z_2, i.e. $z_1\ x\ z_2 \preccurlyeq z_1\ y\ z_2$ and $z_1\ y\ z_2 \preccurlyeq z_1\ x\ z_2$. Note that $\sim_{z_1,z_2}$ is an equivalence relation for any z_1, z_2.

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 $x, y \in \{\epsilon\} \cup Q_L$ and $z_1, z_2 \in \{\epsilon\} \cup Q \setminus (Q_L \cup Q_R))$ such that $x \sim_{z_1,z_2} y$.

Lemma 5. *The rewite relations $\mapsto_L$ and $\mapsto_R$ satisfy the diamond property.*

Proof. The strategy is to prove that the diamond property holds for the generating rules of $\mapsto_L$ and $\mapsto_R$. By a simple induction on the number of atomic left and right rewrites, we get the result for the full rewrite relations.

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 sides of the rules of $\mapsto_L$ is in Q_L^* and $q_R \notin Q_L$, 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. □

We are now ready to apply theorem 1 and conclude that quoting with $\simeq$ is sound.

The following lemma characterizes $\simeq$.

Lemma 6. *For any non-copying states $q, q' \in Q \setminus (Q_L \cup Q_R)$, 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. $q_L^i \simeq q_L^j$, *for* $i, j > 0$.
2. $q_R^i \simeq q_R^j$, *for* $i, j > 0$.
3. *If* $x, y \in Q_L^*$ *and* $x \sim_{q,q'} y$, *then* $q\ x\ q' \simeq q\ y\ q'$.
4. *If* $x \simeq y$ *and* $x' \simeq y'$ *then* $x\ x' \simeq y\ y'$.
5. *If* $x\ q \simeq y\ q$ *and* $q\ x' \simeq q\ y'$, *then* $x\ q\ x' \simeq y\ q\ y'$.
6. $q_R\ q_L\ q_R \simeq q_R\ q_L\ q_R\ q_L\ q_R$.

□

Using the above Lemma, we will use a finer equivalence relation than $\simeq$, with equivalence classes of the form:

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

where

- $z_0 = z_n = \epsilon$ and for i with $0 < i < n$ each $z_i \in Q \setminus (Q_L \cup Q_R)$ is a non-copying state.

and where each e_i is of the form

$$f_0\ q_{R_1}^+\ f_1\ q_{R_2}^+ \cdots q_{R_m}^+\ f_m$$

where for all j with $0 \le j \le m$ we have for some left-copying state $q_L \in Q_L$,

- If $0 < j < m$ and $q_{R_j} = q_{R_{j+1}} = q_R$ then $f_i = q_L^+ (q_R^+\ q_L^+)^*$.
- If $0 < j < m$ and $q_{R_j} \neq q_{R_{j+1}}$, then $f_i = q_L^+$.
- f_0 is an equivalence class in $q_L^* / \sim_{z_i, \epsilon}$.
- f_m is an equivalence class in $q_L^* / \sim_{\epsilon, z_{i+1}}$.
- If $m = 0$, then $f_0 = f_m$ is an equivalence class in $q_L^* / \sim_{z_i, z_{i+1}}$.

5 Implementation

We have implemented the equivalences mentioned in this paper along with some other improvements. We have run several examples to measure the effectiveness of the equivalences, by comparing the number of states built. At the end of this section, we mention some more technical improvements.

In our previous work we have for each algorithm computed the transitive closures for individual *actions* representing one class of statements in the program. Now, we compute the transitive closure of the transducer representing the *entire program*. To be able to do this for some algorithms, we intersect the transducer with the set of reachable configurations before computing the transitive closure. The set of reachable configurations is obtained automatically with reachability analysis using the transitive closure of individual actions.

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

- **Old equivalence** This is the old equivalence used in our previous work.
- **Bi-determinization** Using 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. Szymanski was intersected with the invariant before the transitive closure while Bakery was not, thus the huge difference in the number of states. In fact, reachability with only the transitive closure of individual actions of Bakery does not terminate.

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 just looking at the label. For bidetermistic transducer, any label $L1L2$ and $R1R2$ can be declared dead because they are either not reachable (in the case of left-copying states) or productive (in the case of right-copying states).

Caching Many transitions are merged, and many transitions have the same edge. Thus, caching the result of composing edges provide 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.