Proc. Estonian Acad. Sci. Phys. Math., 1998, **47**, 3, 216–228
https://doi.org/10.3176/phys.math.1998.3.06

CONTEXT-DEPENDENT MINIMIZATION OF STATE/EVENT SYSTEMS

Gerd BEHRMANN, Kåre KRISTOFFERSEN, and Kim LARSEN

Department of Computer Science, Aalborg University, Fredrik Bajers Vej 7 E1, DK-9220 Aalborg Øst, Denmark; e-mail: kgl@iesd.auc.dk

Received 19 January 1998, in revised form 23 February 1998

Abstract. This paper presents a technique for efficient checking of reachability properties of concurrent state/event systems. The technique improves the traditional algorithm for the forwards exploration of the global state space by the incremental construction of subsystems kept small using a *context-dependent minimization*. A tool has been implemented to verify state/event systems. Experimental results report on a feasible automatic verification of the correctness of Milner's scheduler – an often used benchmark – with 100 cells. This result dramatically improves the previous best results for this benchmark. Moreover, our technique has proved well applicable to industrial designs of sizes up to 400 concurrent state machines.

Key words: state/event systems, reachability analysis, compositionality, subsystems, context-dependent minimization.

1. INTRODUCTION

Model checking of finite–state concurrent systems suffers from the potential combinatorial explosion of the state space arising from parallel composition. This phenomenon is also known as the state–explosion problem. During the last decade, various techniques have been developed to avoid the state–explosion problem in verifying finite–state systems, either by *symbolic* representation of the state space using Binary Decision Diagrams (BDDs) [1], by the application of *partial order* methods [2,3], which suppresses unnecessary interleavings of transitions, or by the application of *abstractions* and *symmetries* [4–6].

Our approach is to use a *compositional* technique in model checking embedded reactive systems using a state/event model. Our technique initially considers only a few components in determining the satisfaction of the verification task and if necessary, increases the number of the considered components. After each addition

of a component, the resulting intermediate subsystem is kept small by a suitable *context-dependent minimization.*

In our setting, a state/event model is a fixed number of finite state component transition systems that have input events associated with the transitions. The model is synchronous: each input is reacted upon by all machines in the lock–step. Firing of a transition in an one-component transition system can be made conditional on the local states of other machines using guards. The model is complete with respect to the inputs: In any state and no matter what input event is received, a component can fire *some* transition labelled with this event, regardless of the state of the other components.

The state/event model is convenient for describing the control portion of embedded reactive systems, including smaller systems, such as cellular phones, hi–fi equipment, and cruise controls for cars, and large systems, such as train simulators, flight control systems, telephone and communication protocols. The model is used in the commercial tool visualSTATE^{TM} [7]. This tool assists in developing embedded reactive software by allowing the designer to construct a state/event model and analyze it by either simulating it or by running a consistency checker. The tool automatically generates the code for the hardware of the embedded system. In fact, the consistency checker is a verification tool that checks for a range of properties that any state/event model should have. Some of the checks must be passed for the generated code to be correct, for instance, it is crucial that the model is deterministic. Other checks are issued as warnings that might be design errors, such as transitions that can never fire. A benchmark test in the last section of the paper reports on a feasible automatic verification of the correctness of Milner's scheduler – an often used benchmark – with 100 cells. The best result published earlier on this benchmark is that a scheduler with 50 cells could be handled [8].

Closely related work. A recent paper by Nielsen et al. [9] presents a very efficient BDD-based technique to the verification of state/event systems. In their approach, minimization is based on a promising *propagation* technique that makes the verification of industrial designs possible with more than 1400 state machines.

Outline. In the following section, we define the notion of a state/event system and in Section 3, we present the types of consistency checks we consider in this paper. In Section 4, we define the composition of components and introduce the notion of a well-behaved subsystem. Then in Section 5, we define the forwards reachability properties in a global state graph and in a subsystem. In Section 6, we describe how to minimize a subsystem with respect to its context and illustrate it by an example. Finally, in Section 7, we report on strong experimental results obtained in the verification of the scheduler.

2. STATE/EVENT SYSTEMS

A state/event system consists of a fixed number of component transition systems that synchronize upon the reception of input events. A component transition system

is a finite state transition system, where each transition is labelled with an input event and a guard expressing the legal states of the other component transition systems for this transition to be executable. The component transition systems constituting a state/event system are built over a set of disjoint state–groups as described in the following preliminaries.

Preliminaries. Assume n pairwise disjoint state–groups $St_1, \ldots, St_n$. The global state space is then given by $ST = St_1 \times \cdots \times St_n$. Assume also a fixed set of events $E = e_1, \ldots, e_m$. We have two operators for generalization and projection on the sets of states. The notation for these is as follows:

Notation 1. *Let* $I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}, g \subseteq ST$ *and* $h \subseteq St_{i_1} \times \ldots \times St_{i_k}$. *Then* $h^+ = \{(s_1, \ldots, s_n) \mid (s_{i_1}, \ldots, s_{i_k}) \in h\}$ *and* $\Pi_I(g) = \{(s_{i_1}, \ldots, s_{i_k}) \mid \exists (t_1, \ldots, t_n) \in g.\, t_{i_1} = s_{i_1} \wedge \ldots \wedge t_{i_k} = s_{i_k}\}$. *When* I *is a singleton* $\{i\}$, *we write* $\Pi_i(g)$ *for* $\Pi_{\{i\}}(g)$.

A component transition system over the state–group St_j is a labelled transition system where each transition is labelled with an event $e \in E$ and a guard $g \in ST$. Furthermore, there is a designated initial state $s_j^0 \in St_j$. For reasons of simplicity, we impose three conditions of *consistency, completeness* and *monolithicity* on the component transition systems. These do not form any restriction; in fact, any transition system over St_j, E and ST can be transformed to the form used here.

Definition 1 (Component transition system). *A component transition system over the state–group* St_j *is a labelled transition system* $T_j = \langle St_j, s_j^0, \rightarrow_j \rangle$, *where* s_j^0 *is the initial state and* $\longrightarrow_j \subseteq St_j \times E \times 2^{ST} \times St_j$ *such that*

i) Whenever $s \xrightarrow{e,g} s'$ *then* $\Pi_j(g) = St_j$.

ii) Let $e \in E$, *let* $s \in St_j$ *and let* $s_1, \ldots, s_k$ *be all e-derivatives of* s; *i.e., all* s_i *st.* $s \xrightarrow{e,g_i} s_i$ *for some* g_i. *Then* $\bigcup_i g_i = ST$.

iii) Let $e \in E$ *and let* $s, s' \in St_j$. *Whenever* $s \xrightarrow{e,g} s'$ *and* $s \xrightarrow{e,h} s'$ *then* $g = h$.

For the transition $s \xrightarrow{e,g} s'$ in a component transition system over the state–group St_j, the guard $g \subseteq ST$ denotes the set of global states in which this transition is enabled. Guards cannot be arbitrary, in fact, the three conditions above express which component transition systems are allowed.

ad i) Guards are consistent, i.e., the guard g on the transition $s \xrightarrow{e,g} s'$ does not put any restrictions on the current state of T_j. However, the transition as a whole, naturally requires T_j to be in the state s in order to be enabled.

ad ii) The behaviour of a component transition system is *complete* wrt. all events in all states, i.e., a component transition system can respond to any event e in

any of its states s, regardless of the current state of the remaining component transition systems.

When all components in a state/event system have this property, the semantics of the concurrent behaviour upon the reception of an event will be that of pure synchronization. The completeness condition is not a restriction, however. Suppose that a state s is incomplete with respect to event e, we can safely add to T_j the transition $s \xrightarrow{e,g^c} s$, where $g = \cup\{h | \exists h, s'.s \xrightarrow{e,h} s'\}$.

ad iii) Component transition systems are *monolithic*, i.e., for all pairs of states s, s' in a component transition system over the state–group St_j and for all events e, there will be at most one transition. This property can be obtained by substituting all pairs of transitions $s \xrightarrow{e,g} s'$ and $s \xrightarrow{e,h} s'$ with $s \xrightarrow{e,g \cup h} s'$.

Now in all the following, assume component transition systems over ST, i.e., let $T_1 = \langle St_1, s_1^0, \longrightarrow_1 \rangle, \ldots, T_n = \langle St_n, s_n^0, \longrightarrow_n \rangle$. The state/event system described by a set of component transition systems $T_1, \ldots, T_n$ will be denoted $(T_1 \mid \ldots \mid T_n)$. The semantic interpretation of parallel composition will be that of full synchronization upon the reception of events. The global transition system of a state/event system is then given as follows:

Definition 2 (Global transition system). *The global transition system induced by* $T_1, \ldots, T_n$ *is* $T = \langle ST, \bar{s}^0, \longrightarrow \rangle$, *where* $\bar{s}^0 = (s_1^0, \ldots, s_n^0)$ *and* $(s_1, \ldots, s_n) \xrightarrow{e,tt} (s_1', \ldots, s_n')$ *if* $s_i \xrightarrow{e,g_i} s_i'$ *with* $(s_1, \ldots, s_n) \in g_i$ *for all* i.

The size of the global transition system is potentially exponential in the number of its components, and hence, it is infeasible as an object of verification as mentioned in the Introduction. Instead, we will be considering the reduced versions of the global transition system in the form of subsystems. A subsystem is the *composition* of a subset of the components in a state/event system.

3. CONSISTENCY CHECKS

The consistency checker in visualSTATE^{TM} performs five predefined types of checks each of which can be reduced to verifying reachability properties. Additionally, visualSTATE^{TM} is able to check for the absence of local deadlocks. The focus in this paper is on the checking of reachability properties only. Checking reachability properties means to determine for a set of global states g if a state in g can be reached from the initial state.

The ability of a state/event to execute all transitions is one of the checks performed by visualSTATE^{TM}. For each transition, it is checked if there exists a global state such that the guard g on the transition is satisfied. Also, visualSTATE^{TM} performs a check for "conflicting transitions", i.e., it checks whether two or more transitions can become enabled in the same local state, leading

to non-determinism. This can be reduced to questions of reachability by considering all pairs of guards g_1 and g_2 of transitions with the same source state $s \in St_j$ and input event e. These transitions must have different targets due to the monolithicity condition. A conflict can occur if a global state t is reachable such that $t_j = s$ and $t \in g_1 \cap g_2$.

In order to reduce the number of checks to be carried out, we perform an *implicational analysis* between the properties to be checked. If a property g_1 implies another property g_2, then clearly, if g_1 is reachable, so is g_2. Initially, we sort the guards in the ascending order of the size of their satisfying state space. Hence, the most specific guards are checked first, and for each new guard to be checked, we compare it to the already checked (and reachable) guards. If the new guard includes one of them, then we know that it is satisfiable. In our experiments, the reduction in the number of checks obtained using implicational analysis is between 40 and 90%.

4. COMPOSITION AND WELL-BEHAVED SUBSYSTEMS

Subsystems. In our compositional method, we will consider subsystems, i.e., the transition systems representing the concurrent behaviour of a subset of the components constituting a state/event system. In a subsystem, the guards on transitions will only concern the components in the context, as the internal constraints are already resolved during composition.

Definition 3 (Subsystem). *Let $T_1, \ldots, T_n$ be component transition systems. Let $I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$. Then we say that $T_I = \langle St_I, s_I^0, \longrightarrow_I \rangle$, where $s_I^0 = (s_{i_1}^0, \ldots, s_{i_k}^0)$ is the initial state and $St_I \subseteq \times_{i \in I} St_i$ and $\longrightarrow_I \subseteq St_I \times E \times 2^{ST} \times St_I$ is a subsystem with respect to I.*

Now, the composition of the two subsystems is defined in the following way.

Definition 4 (Composition). *Let $I, J \subseteq \{1, \ldots, n\}$ with $I \cap J = \emptyset$ and let $T_I = \langle St_I, s_I^0, \longrightarrow_I \rangle$ and $T_J = \langle St_J, s_J^0, \longrightarrow_J \rangle$ be subsystems wrt. I and J. Then the transition system $T_I \bullet T_J = \langle St_{IJ}, s_{IJ}^0, \longrightarrow_{IJ} \rangle$ is the subsystem wrt. $I \cup J$, where s_{IJ}^0 is the initial state obtained from s_I^0 and s_J^0 and $St_{IJ} = \Pi_{I \cup J}(St_I^+ \cap St_J^+)$ and whenever $\bar{s} \xrightarrow{e,g}_I \bar{s}'$ and $\bar{t} \xrightarrow{e,h}_J \bar{t}'$, where $\bar{s}, \bar{s}' \in St_I, \bar{t}, \bar{t}' \in St_J$ st. $\bar{t} \in \Pi_J(g)$ and $\bar{s} \in \Pi_I(h)$ then $\bar{s} \bullet \bar{t} \xrightarrow{e,k}_{IJ} \bar{s}' \bullet \bar{t}'$, where*

$$k \;=\; \left[\Pi_{(I \cup J)^c}(g) \cap \Pi_{(I \cup J)^c}(h)\right]^+$$

and $\bullet$ is defined as follows: $\bar{s} \bullet \bar{t} =^{defn} \Pi_{I \cup J}(\bar{s}^+ \cap \bar{t}^+)$. Note that the righthandside is a singleton, and hence $\bar{s} \bullet \bar{t}$ is a well-defined element.

Well-behaved subsystems. If a subsystem enjoys the three properties of consistency, completeness and monolithicity, we say that it is *well-behaved*. In fact, well-behaved subsystems can be seen as the generalization of component

transition systems, and as we will see in the following, composition of well-behaved subsystems preserves well-behavedness.

Definition 5 (Well-behaved subsystem). *Let $T_1 \ldots, T_n$ be component transition systems and let $I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$. Then a subsystem wrt. I, $T_I = \langle St_I, s_I^0, \longrightarrow_I \rangle$ is said to be a well-behaved subsystem if the following holds:*

i) Whenever $(s_{i_1}, \ldots, s_{i_k}) \xrightarrow{e,g} (s'_{i_1}, \ldots, s'_{i_k})$, then $\Pi_I(g) = St_I$.

ii) Let $e \in E$, let $(s_{i_1}, \ldots, s_{i_k}) \in St_I$ and let $(s^1_{i_1}, \ldots, s^1_{i_k}), \ldots, (s^l_{i_1}, \ldots, s^l_{i_k})$ be all e-derivatives of $(s_{i_1}, \ldots, s_{i_k})$; i.e. $(s_{i_1}, \ldots, s_{i_k}) \xrightarrow{e,g_1} (s^1_{i_1}, \ldots, s^1_{i_k}), \ldots, (s_{i_1}, \ldots, s_{i_k}) \xrightarrow{e,g_l} (s^l_{i_1}, \ldots, s^l_{i_k})$, then $\bigcup_{i=1}^{l} g_i = ST$.

iii) Let $e \in E$ and let $(s_{i_1}, \ldots, s_{i_k}), (s'_{i_1}, \ldots, s'_{i_k}) \in St_I$. Whenever $(s_{i_1}, \ldots, s_{i_k}) \xrightarrow{e,g} (s'_{i_1}, \ldots, s'_{i_k})$ and $(s_{i_1}, \ldots, s_{i_k}) \xrightarrow{e,h} (s'_{i_1}, \ldots, s'_{i_k})$, then $g = h$.

The composition of two well-behaved subsystems is also a well-behaved subsystem as stated in the following theorem:

Theorem 1. *Let $I, J \subseteq \{1, \ldots, n\}$ s.t. $I \cap J = \emptyset$ and let T_I and T_J be well-behaved subsystems wrt. I and J, then $T_I \bullet T_J$ is also a well-behaved subsystem (wrt. $I \cup J$).*

Theorem 2. *Let T_j be a component transition system with respect to the state group St_j. Then T_j is also a well-behaved subsystem with respect to $\{j\}$.*

Theorem 3. *Let $T_1, \ldots, T_n$ be component transition systems and let T be the global transition system induced by these. Then T is a well-behaved subsystem with respect to $\{1, \ldots, n\}$.*

We see that well-behaved subsystems preserve the three properties of *consistency*, *completeness* and *monolithicity* from the component transition systems they are composed from. This leaves us with well-behaved subsystems as the basic building blocks in our compositional verification procedure. In the rest of the paper, we will be using the term subsystem for a well-behaved subsystem, i.e., a subsystem is well-behaved unless something else is mentioned directly.

Obviously, given a subsystem T_I wrt. some $I \subseteq \{1, \ldots, n\}$, it will only be possible to determine reachability of local states in the components $\{T_i \mid i \in I\}$. Fortunately, almost all properties we need to check are this kind of *local* properties, and hence, by a careful selection of component transition systems, it will suffice to consider a subsystem instead of the global transition system. Moreover, as the intermediate subsystems can be kept small by the use of a suitable context dependent minimization and propagation and pruning, state–explosion can be avoided. The results displayed in the last section of the paper justify this claim.

5. REACHABILITY ANALYSIS

The sets of reachable states in a state event/system are those states that can be reached by following the transition relation starting from the initial state. This is formalized in the following definition:

Definition 6 (Reachability). *Let $T_1, \ldots, T_n$ be component transition systems and let $T = \langle ST, \bar{s}^0, \longrightarrow \rangle$ be the global transition system induced by these. Further, let $g \subseteq ST$. We say that g is reachable in T if there exists a sequence of transitions $\bar{s}_0 \xrightarrow{e_1,tt} \bar{s}_1 \cdots \bar{s}_{n-1} \xrightarrow{e_n,tt} \bar{s}_n$ such that $\bar{s}_0 = \bar{s}^0$ and $\bar{s}_n \in g$.*

Lemma 1. *Let $I, J \subseteq \{1, \ldots, n\}$ such that $I \cap J = \emptyset$ and let T_I, T_J be well-behaved subsystems with respect to I and J. Further, let $g \subseteq ST$ and let $s^I \in St_I$ and $s^J \in St_J$. Then $\bar{s}^I \in \Pi_I(g)$ and $\bar{s}^J \in \Pi_J(g)$ if and only if $\bar{s}^I \bullet \bar{s}^J \in \Pi_{I \cup J}(g)$.*

The composition preserves reachability properties as stated in the following theorem.

Theorem 4. *Let T_I and T_J be subsystems and let $g \in ST$. Then g is reachable in $(T_I \mid T_J \mid \Pi_{j \notin I \cup J} T_j)$ if and only if g is reachable in $(T_I \bullet T_J \mid \Pi_{j \notin I \cup J} T_j)$.*

Mostly we will be checking reachability of the property that at most depends on a subset of the total set of component transition systems constituting a state/event system.

Definition 7. *Let $g \subseteq ST$ be a guard and let $I \subseteq \{1, \ldots, n\}$. We say that g is a property at most depending on I if $[\Pi_I(g)]^+ = g$.*

5.1. Sub-Reachability

It is possible to reason about reachability properties by considering only a subsystem. Let T_I be a subsystem and let $g \subseteq ST$ be a guard at most depending on I. Let $R_0(g) = g$ and let

$$\begin{aligned} R_i(g) \;=\; & \{\bar{s} \in St_I \mid \exists \bar{s}_1, \ldots, \bar{s}_k \in R_{i-1}(g), g_1, \ldots, g_k \in ST, e_1, \ldots, e_k \in E \\ & \text{s.t } \bar{s} \xrightarrow{e_i,g_i} \bar{s}_i, i = 1, \ldots, k, \text{ and } [\cup_{i=1}^{k} g_i]^+ = ST\} \cup R_{i-1}(g). \end{aligned}$$

We compute the minimum set of states containing g and closed under the application of $R_i(g)$. This set of states has the property that regardless of the state of the context there exists a sequence of transitions leading to a g-state. Consider the following definition of sub-reachability.

Definition 8 (Sub-reachability). *Let $T_1, \ldots, T_n$ be component transition systems, let $I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ and let T_I be a subsystem wrt. I. Further, let $g \subseteq ST$ be a property at most depending on I. Then if $(s^0_{i_1}, \ldots, s^0_{i_k}) \in R_i(g)$ we say that g is sub-reachable in T_I.*

In other words, a set of states g is sub-reachable in a subsystem T_I if one of the states in g can be reached in T_I regardless of the component transition systems not being part of T_I. The following theorems state when exactly we may infer answers to reachability questions from considering only subsystems:

Theorem 5. *Let $T_1, \ldots, T_n$ be component transition systems and let T be the global transition system induced by these. Let $I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ and let T_I be a subsystem wrt. I. Further, let $g \subseteq ST$ be a property at most depending on I. Then g is reachable in T if g is sub-reachable in T_I.*

Theorem 6. *Let $T_1, \ldots, T_n$ be component transition systems and let T be the global transition system induced by these. Let $T_{\{1,\ldots,n\}}$ be a subsystem obtained by composing $T_1, \ldots, T_n$. Let $I \subseteq \{1, \ldots, n\}$ and let $g \subseteq ST$ be a property at most depending on I. Then g is not reachable in T if g is not sub-reachable in $T_{\{1,\ldots,n\}}$.*

5.2. Transformation

When used directly as presented above, the notion of sub-reachability does not make reachability checking as efficient as possible in the sense that the state space is not kept small in all steps of the model checking procedure. More precisely, model checking is only improved if after a series of compositions it happens that the check for sub-reachability has a positive answer. In all those steps of the algorithm where the check for sub-reachability has a negative answer, the intermediate state space is left unchanged before the next composition takes place. To obtain an efficient model checking tool, it is vital that after each composition step we make a compression of the intermediate state space. This can be done, based on the notion of sub-reachability as follows:

Let T_I be a subsystem, let g be a guard at most depending on I and let $\bar{s} \in St_I$. We will say that g is sub-reachable from $\bar{s}$ or that g *propagates to* $\bar{s}$ if $\bar{s} \in R_i(g)$. After each composition step, the efficient model checker will compute the set of states that g propagates to and store only one single state as a representative for them all.

We will attempt to answer the reachability problem of a property g by checking for sub-reachability in a subsystem T_I. In case the reachability question cannot be answered, we provide extra information by extending T_I to a subsystem T_J with $I \subseteq J$. The extension will be done in a clever way by letting a dependency analysis determine which component to add next. Hence, in the worst case situation we end up constructing $T_{\{1,\ldots,n\}}$. However, as our experiments show, it almost always suffices to consider rather small subsets of the total number of component transition systems, and thus we succeed in keeping model checking feasible.

6. CONTEXT DEPENDENT MINIMIZATION

In the previous sections, we have presented a method for analyzing reachability properties based on a compositional approach. That means the component transition

systems constituting the state/event system under investigation are gradually composed in the order that a dependency analysis finds most appropriate. After each addition of a machine, the resulting intermediate subsystem is examined for forwards reachability.

The idea behind context-dependent minimization is to keep these intermediate systems small by suitable minimizations. More specifically, the technique is based on minimization with respect to a bisimulation [10,11]. Obviously, to ensure correctness of the method, it is important that the states of the subsystem, which are distinguished by some guard occurring in the components not yet included, are kept distinct. Hence, the relevant bisimulation is context-dependent in the sense of [12–14].

6.1. Context-Dependent Bisimulation

In the following, we provide a sketch of the minimization technique: Let $T_1, \ldots, T_n$ be the machines constituting the considered system. Now assume that we have composed the components T_i for $i \in I$, where $I \subseteq \{1, \ldots, N\}$, and let $T_I = (St_I, s_I^0, T_I)$ be the subsystem representing this composition. Now assume further that G is the set of guards on T_I occurring in the remaining machines. Then two states s, t of T_I are said to be the bisimilar modulo G, $s =_G t$, iff s and t satisfy the same guards of G and

1. Whenever $s \xrightarrow{e,g} s'$, then $t \xrightarrow{e,h} t'$ for some h, t' st. $g \subseteq h$ and $s' =_G t'$.

2. Whenever $t \xrightarrow{e,g} t'$, then $s \xrightarrow{e,h} s'$ for some h, s' st. $g \subseteq h$ and $t' =_G s'$.

Using a variant of the partition algorithm in [15] (alternatively [16]), one may obtain a minimized version of the subsystem T_I, $T_{I_{min}}^G$. In the minimized version, the set of states consists of exactly one representative from each of the $=_G$-equivalence classes of St_I, and transitions are obtained in the following way: For each pair of $=_G$-equivalence classes b and b' and each event e, the transitions $s_1 \xrightarrow{e,g_1} s_1, \ldots, s_k \xrightarrow{e,g_k} s'_k$, where $s_1, \ldots, s_k \in b$ and $s'_1, \ldots, s'_k \in b'$ are substituted by one transition from b to b' labelled with e and the union of g_i for $i = 1, \ldots, k$. The correctness of this minimization effort follows from the following theorem stating that the appropriate context-dependent minimization of a subsystem preserves reachability properties:

Theorem 7. *Let $T_1, \ldots, T_n$ be component transition systems and let $g \subseteq ST$. Let $I \subseteq \{1, \ldots, N\}$ and let T_I be the subsystem obtained by composing the components $\{T_i | i \in I\}$. Further, let $G = \{\Pi_I(h) \,|\, \exists j \in \bar{I}. \exists s, s' \in St_j s.t. s \xrightarrow{e,h} s'\} \cup \Pi_I(g)$. Then g is reachable in $(T_I \,|\, \Pi_{j \notin I} T_j)$ iff g is reachable in $\left(T_{I_{min}}^G \,|\, \Pi_{j \notin I} T_j\right)$.*

6.2. Minimization Example

Assume a state/event system with three components; T_1 builds over the state group $\{p_1, p_2\}$, T_2 builds over $\{q_1, q_2\}$ and T_3 builds over $\{U_1, U_2, U_3\}$. T_1 and T_2 are displayed in Fig. 1. T_3 is not presented. In this figure, an alternative and more compact notation for guards on transitions is used. It means that a transition $s \xrightarrow{e,tt} s'$ in a component transition system over the state–group St_j is enabled in all global states $(s_1, \ldots, s_n)$, where $s_j = s$. Similarly, a transition $s \xrightarrow{e,U_1} s'$ in a component transition system over the state–group St_j is enabled in all global states $(s_1, \ldots, s_n)$, where $s_j = s$ and $s_l = U_1$ if U_1 belongs to the state–group St_l. Assuming the current verification task demands us to compose all three systems, we chose to initially compose T_1 and T_2 and get the subsystem T_{12}, which is a four state transition system (Fig. 1). Before composing T_{12} with T_3, we will attempt to minimize T_{12} with respect to its context, T_3. Now assume that the only guard on a transition in T_3 different from tt is on the form p_1. We will then have to initially split the state space of T_{12} into two parts $\{(p_1, q_1), (p_1, q_2)\}$ and $\{(p_2, q_1), (p_2, q_2)\}$. By

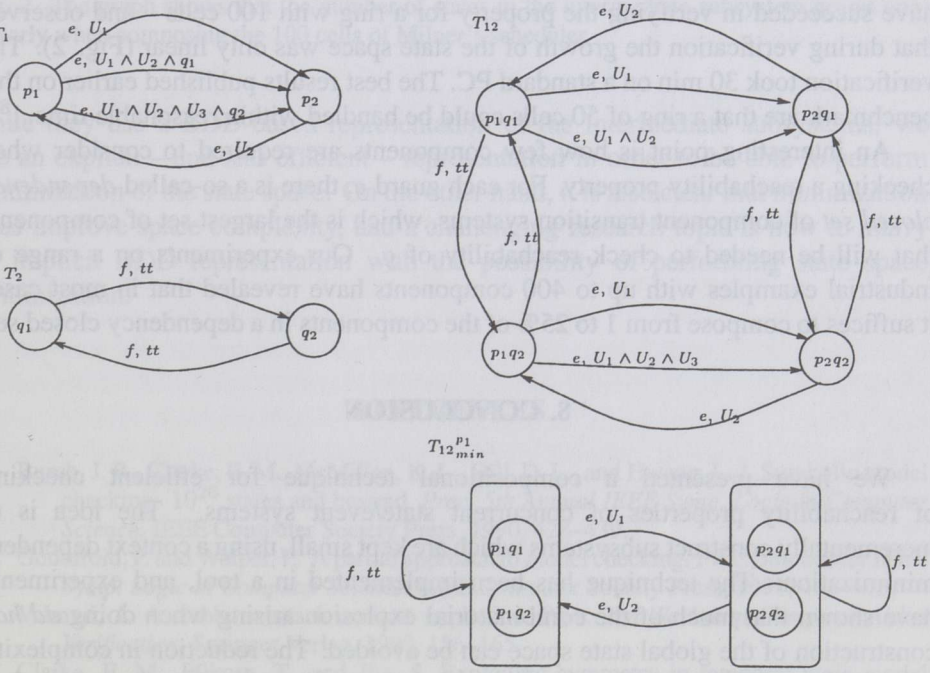

Fig. 1. Two-component transition systems T_1 and T_2 (left) are composed into the subsystem T_{12}(right). T_{12} is then minimized with respect to the guard p_1. The resulting transition system is $T_{12}{}^{l_1=p_1}_{min}$ (bottom).

a further examination of the transitions of T_{12}, we find that (p_1, q_1) and (p_1, q_2) can match each other's transitions. The same applies to the pairs (p_2, q_1) and (p_2, q_2). Hence, we get the minimized version $T_{12}{}^{p_1}_{min}$, which is a two-state transition system (Fig. 1).

7. EXPERIMENTAL RESULTS

A successful application of our tool on Milner's scheduler [10], which is a scalable benchmark, has been carried out. Milner's scheduler consists of a ring of n cells and a token circulating in this ring. When the ith cell receives the token, it can start its job and then terminate or pass on the token to cell the numbered $i+1$ in either order. The specification is that the jobs are started in sequence: $1, 2, \ldots, n, 1, \ldots$. We model the system as a state/event system with one component transition system for each cell. The specification is represented as an additional testing automaton in the style of [17]. The testing automaton will enter a certain *bad* state if ever the jobs are not started in the right sequence. Hence, the verification amounts to checking that the testing environment never enters this bad state. Proving this naturally requires a composition of all $n+1$ components of the system, and thus the state space should explode. However, when our context-dependent minimization is applied to the intermediate subsystem after each composition step explosion is avoided. We have succeeded in verifying the property for a ring with 100 cells – and observed that during verification the growth of the state space was only linear (Fig. 2). The verification took 30 min on a standard PC. The best results published earlier on this benchmark are that a ring of 50 cells could be handled within reasonable time [8].

An interesting point is how few components are required to consider when checking a reachability property. For each guard g, there is a so-called *dependency closed set* of component transition systems, which is the largest set of components that will be needed to check reachability of g. Our experiments on a range of industrial examples with up to 400 components have revealed that in most cases it suffices to compose from 1 to 25% of the components in a dependency closed set.

8. CONCLUSION

We have presented a compositional technique for efficient checking of reachability properties of concurrent state/event systems. The idea is to incrementally construct subsystems which are kept small, using a context dependent minimization. The technique has been implemented in a tool, and experiments have shown that much of the combinatorial explosion arising when doing *ad hoc* construction of the global state space can be avoided. The reduction in complexity results from our context-dependent minimization and from the fact that most often only a small subset of the components in a system needs to be considered.

It is obvious that our implementation is generally not as efficient as the tool presented in [9]. Both approaches utilize the same compositional principle but

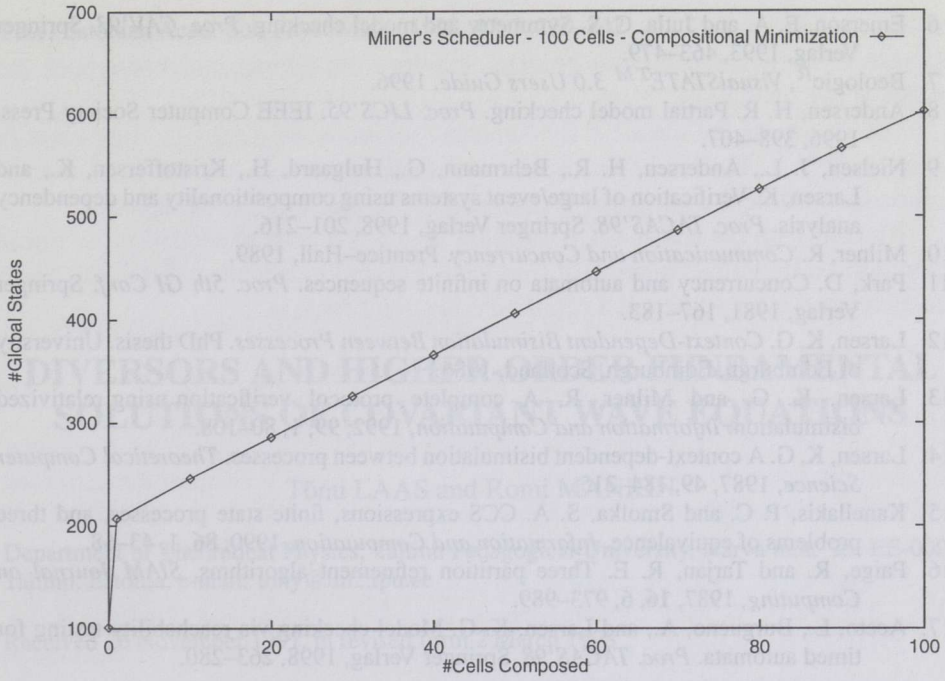

Fig. 2. The graph shows that the number of states in the intermediate subsystem grows only linearly when composing the 100 cells of Milner's scheduler.

while they use a BDD-based representation of the intermediate subsystems, we use an explicit – and less efficient – representation in order to be able to perform minimization of the state space. On the other hand, it is also clear that minimization *does* improve space complexity, and a challenging research topic is how to marry an implicit BDD representation with the possibility of performing state space minimization.

REFERENCES

1. Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L., and Hwang, L. J. Symbolic model checking: 10^{20} states and beyond. *Proc. 5th Annual IEEE Symp. Logic in Computer Science.* IEEE Computer Society Press, 1990, 428–439.
2. Godefroid, P. and Wolper, P. A partial approach to model checking. *Proc. 6th Annual IEEE Symp. Logic in Computer Science.* IEEE Computer Society Press, 1991, 406–415.
3. Valmari, A. A stubborn attack on state explosion. *Proc. 2nd Workshop Computer Aided Verification.* Springer Verlag, 1990, 156–165.
4. Clarke, E. M., Filkorn, T., and Jha, S. Exploiting symmetry in temporal logic model checking. *Proc. CAV'93.* Springer Verlag, 1993, 450–462.
5. Clarke, E. M., Grümberg, O., and Long, D. E. Model checking and abstraction. *Conf. Record of 19th Annual ACM SIGPLAN-SIGACT Symp. Principles of Programming Languages.* Alberquerque, New Mexico, 1992, 342–354.

6. Emerson, E. A. and Jutla, C. S. Symmetry and model checking. *Proc. CAV'93*. Springer Verlag, 1993, 463–479.
7. Beologic^{R}, *VisualSTATE*^{TM} *3.0 Users Guide*. 1996.
8. Andersen, H. R. Partial model checking. *Proc. LICS'95*. IEEE Computer Society Press, 1996, 398–407.
9. Nielsen, J. L., Andersen, H. R., Behrmann, G., Hulgaard, H., Kristoffersen, K., and Larsen, K. Verification of large/event systems using compositionality and dependency analysis. *Proc. TACAS'98*. Springer Verlag, 1998, 201–216.
10. Milner, R. *Communication and Concurrency*. Prentice–Hall, 1989.
11. Park, D. Concurrency and automata on infinite sequences. *Proc. 5th GI Conf.* Springer Verlag, 1981, 167–183.
12. Larsen, K. G. *Context-Dependent Bisimulation Between Processes*. PhD thesis. University of Edinburgh, Edinburgh, Scotland, 1986.
13. Larsen, K. G. and Milner, R. A complete protocol verification using relativized bisimulation. *Information and Computation*, 1992, **99**, 1, 80–108.
14. Larsen, K. G. A context-dependent bisimulation between processes. *Theoretical Computer Science*, 1987, **49**, 184–215.
15. Kanellakis, P. C. and Smolka, S. A. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 1990, **86**, 1, 43–68.
16. Paige, R. and Tarjan, R. E. Three partition refinement algorithms. *SIAM Journal on Computing*, 1987, **16**, 6, 973–989.
17. Aceto, L., Burgueno, A., and Larsen, K. G. Model checking via reachability testing for timed automata. *Proc. TACAS'98*. Springer Verlag, 1998, 263–280.

SÜSTEEMIDE OLEK–SÜNDMUS KONTEKSTIST SÕLTUV MINIMEERIMINE

Gerd BEHRMANN, Kåre KRISTOFFERSEN ja Kim LARSEN

On esitatud efektiivne meetod süsteemide olek–sündmus saavutatavusomaduste tuvastamiseks. Meetod muudab paremaks globaalses olekuruumis pärisuunalise otsimise traditsioonilised algoritmid. Selleks konstrueeritakse järk-järgult alamsüsteemid, kasutades nn. kontekstist sõltuvat minimeerimist. Süsteemide olek–sündmus automaatseks verifitseerimiseks on realiseeritud vahend, mille efektiivsust kinnitavad Milneri plaanuri verifitseerimisel saadud katseandmed 100 sõlmega juhu jaoks. Meetod osutus tunduvalt paremaks seni saavutatud parimast tulemusest. Kirjeldatud meetod on sobiv ka tööstusliku mastaabiga projekteerimisülesannete puhul (kuni 400 paralleelset olekumasinat).