A Unifying Model Checking Approach for Safety Properties of Parameterized Systems *

Monika Maidl

LFCS, University of Edinburgh, The Kings's Buildings, Edinburgh EH9 3JZ
Email: monika@dcs.ed.ac.uk

Abstract. We present a model checking algorithm for safety properties that is applicable to parameterized systems and hence provides a unifying approach of model checking for parameterized systems. By analysing the conditions under which the proposed algorithm terminates, we obtain a characterisation of a subclass for which this problem is decidable. The known decidable subclasses, token rings and broadcast systems, fall in our subclass, while the main novel feature is that global guards (in form of unnested quantification over index variables) are allowed under certain circumstances. The approach is illustrated by the Bakery algorithm and the Illinois protocol.

1 Introduction

We present a model checking algorithm for safety properties that is applicable to parameterized systems. A parameterized system is a family of systems, one for each instantiation of the parameter, where an instantiation by n is the composition of n copies of the system, and the verification problem consists in checking whether all instantiations fulfil a given property. Model checking for parameterized systems has been shown to be undecidable in general [Suz88], so the problem can only be approached for subclasses or by semi-algorithmic methods. Solutions based on different technical frameworks have been proposed. Since our model checking algorithm is applicable to parameterized systems in general, it provides a unifying method for different subclasses.

By exploring under which restrictions the algorithm terminates, we obtain a characterisation of a class of parameterized systems for which model checking of safety properties is decidable. The restrictions concern the way the copies communicate with each other: The admissible forms of communication are the very restricted synchronisation of token rings (no information exchanged between neighbours except "I have/have not the token") and the anonymous synchronisation of broadcast protocols. Another form of communication, using values of variables in neighbouring copies in guards or assignments like in the Dining Philosophers example, has to be excluded. Under certain restrictions, we can however allow communication by global variables and by global guards,

* This work was supported supported by the European Commission (FETproject ADVANCE, contract No IST-1999-29082).

(expressed by universal quantification over index variables, which run over the instantiated copies). The latter is used e.g. in the Bakery algorithm, and in cache coherence protocols with a global condition considered in [Del00], e.g. the Illinois protocol, and extends the known subclasses of decidable systems. Our algorithm terminates for both the Bakery algorithm and the cache coherence protocols considered in [Del00], but the Bakery algorithm does not lie in the subclass we characterise because it contains integer variables, while we restrict considerations to system variables with finite domains. It is possible to prove termination of our algorithm for the examples of [Del00], but the proof has to take into account a special property of these protocols, which is described in the last section.

As specification logic, we consider a linear time logic built over state predicates that can contain index variables. We restrict ourselves to safety properties; liveness properties of broadcast protocols were shown to be undecidable in [EFM99], so decidability of model-checking certainly does not extend to liveness properties for the whole class we characterise.

Related Work. Model checking for parameterized systems has been addressed by many researchers. One line of research is concerned with restrictions that, when imposed on parameterized systems, make model checking decidable for safety properties or even full temporal logic. The systems considered there are either token rings, like in [EN95], or broadcast protocols, which were introduced in [EN98] and also considered in [EFM99]; [GS92] deals with a restricted form of broadcast protocols. Both types are subsumed by our subclass.

The approach for broadcast systems in [EN98] and [EFM99] falls under the paradigm of using well-ordered sets for the verification of infinite-state systems [ACJT96]. All sets that are considered are sets that are upward closed with respect to a well order. This means that all guards of transitions have to describe upward closed sets, which excludes certain global conditions, e.g. the one used in the cache coherence protocol that Delzanno considered in [Del00].

Regular model checking, advocated by e.g. [KMM$^+$97] and [BW98], is based on representing sets of states by regular languages. Termination is obtained by applying some form of acceleration in order to compute the transitive closure of the transition relation [BJNT00]. Emphasis lies not on detecting decidable classes but providing general, not necessarily exact methods to handle a large class of systems. In this context, handling global guards was considered in [ABJN99].

Overview. First we explain the types of parameterized systems we consider. The third section contains our model checking algorithm for ordinary systems, while in the fourth section it is adapted to parameterized systems, and the conditions under which the algorithm terminates are given.

2 Framework

As program notation we use concurrent state-based guarded-command systems; a system is hence of form $(V, C_1, \ldots, C_n, I)$, where V is a set of variables, C_i are components and I is a predicate describing the initial states. A component consists of a set of transitions, where guards and assignments are built over boolean or enumerative variables or integer terms. More precisely, the terms occurring on the right-hand side of assignments are terms of Presburger arithmetic, enumerative constants or formulas of Presburger arithmetic, depending on the sort of the left-hand side of the assignment, and guards are formulas of *Presburger arithmetic*. The restriction to Presburger arithmetic, i.e. to multiplication only with constants, guarantees decidability. A step is defined by choosing some component and one of its transitions with a guard that is satisfied in the current state, and performing all assignments of this transition simultaneously. As an example of the program notation, consider Table 1, the well-known Bakery algorithm, in a version for 2 components.

Table 1. Program text for the 2-component Bakery algorithm

V: $c_1, c_2 : \{T, W, C\}, \quad n_1, n_2 : \mathrm{NAT}$	
I: $c_1 = T \wedge c_2 = T \wedge n_1 = 0 \wedge n_2 = 0$	
Component 1:	Component 2:
$c_1 = T \longrightarrow \left\langle \begin{array}{l} c_1 := W \\ n_1 := max(n_1, n_2) + 1 \end{array} \right\rangle$	$c_2 = T \longrightarrow \left\langle \begin{array}{l} c_2 := W \\ n_2 := max(n_1, n_2) + 1 \end{array} \right\rangle$
$\begin{array}{l} c_1 = W \wedge \\ (n_2 = 0 \vee n_1 < n_2) \end{array} \longrightarrow \langle\, c_1 := C \,\rangle$	$\begin{array}{l} c_2 = W \wedge \\ (n_1 = 0 \vee n_2 < n_1) \end{array} \longrightarrow \langle\, c_2 := C \,\rangle$
$c_1 = C \longrightarrow \left\langle \begin{array}{l} c_1 := T \\ n_1 := 0 \end{array} \right\rangle$	$c_2 = C \longrightarrow \left\langle \begin{array}{l} c_2 := T \\ n_2 := 0 \end{array} \right\rangle$

2.1 Parameterized Systems

First we define a state language for parameterized systems.

Definition 1 (Index predicates). Index terms *are of form* $j + k$ *or* k*, where* j *is an index variable and* k *is an integer constant.*
Index predicates *are defined as follows:*

– Basic index predicates *are quantifier-free expressions over variables that can be indexed by index terms.*
– *We say that the index term* $j + k$ occurs *in the index predicate* p *if there is some variable* y *occurring in* p *in form* $y[j + k]$*.*
– *If* p *is a basic index predicate and* $j_1, \ldots, j_n$ *are index variables s.t. all index terms occurring in* p *that contain some* j_i *have constant* 0*, then*
$$\forall\, j_1 \,\ldots \forall\, j_n \,(a \rightarrow p) \text{ and } \exists\, j_1 \cdots \exists\, j_n \,(a \wedge p)$$
are index predicates, where a *is a conjunction of expressions of form* $j_i \neq j{+}k$ *for an index variable* j *different from* $j_1, \ldots, j_n$*, or* $j_i \neq j_{i'}$*.*
– *Index predicates are closed under boolean operations.*

The restriction that a quantified variable j can only occur in index terms without constants is necessary for termination of the model checking algorithm, more precisely to guarantee that no new index terms are generated by instantiating quantifiers.

Models of index predicates with respect to n consist of a valuation v for the occurring index variables in $\{0, \ldots, n-1\}$ and of a valuation s for the system variables, where for every indexed system variable y, s defines values for $y[0], \ldots, y[n-1]$. We write $s, v \models_n p$ if s and v form a model for p with respect to n.

A **parameterized system** $S = (V, C[i], I)$ differs from an ordinary system in that the transitions of $C[i]$ are parameterized by the index variable i.[1] Note that we only consider systems where domains of system variables are defined independently of the parameterization.

Accordingly, some variables of V are indexed, while the others act as global variables. The guards of an *parameterizable component* $C[i]$, are index predicates, but we do not allowed quantification over index variables on the right-hand side of assignments. This guarantees that the predicates generated during model checking remain in the class of indexed predicates. The only index variable appearing freely in transitions of $C[i]$ is i, and on the left-hand side of an assignment we only allow i as index term, without constant, which means that a copy can only modify its own variables. To simplify the presentation, we assume that quantified expressions in guards quantify over only one index variable. Our results however also hold for the general case. The initial predicate I is a closed index predicate.

A parameterized version of the Bakery algorithm, shown in Table 2, should illustrate the notion of parameterized system.

Table 2. Parameterized bakery algorithm

$V: \ c[i] : \{T, W, C\}, \ n[i] : \mathrm{INT}$
$I: \ \forall i \, (n[i] = 0 \wedge c[i] = T)$
$c[i] = T \longrightarrow \left\langle \begin{array}{l} c[i] := W \\ n[i] := (max_j n[j]) + 1 \end{array} \right\rangle$
$c[i] = W \wedge \forall j (j \neq i \rightarrow n[i] < n[j] \vee n[j] = 0) \longrightarrow \left\langle c[i] := C \right\rangle$
$c[i] = C \longrightarrow \left\langle \begin{array}{l} c[i] := T \\ n[i] := 0 \end{array} \right\rangle$

For a natural number n, the **instantiation $S[n]$ of a parameterized system** S is $(V[n], C[0], \ldots, C[n-1], I[n])$, where $V[n]$ is the set of ordinary variables

[1] The proposed approach is applicable to systems composed of different parameterizable components and of ordinary components.

of V together with, for every indexable variable y in V, $y[0], \ldots, y[n-1]$, and where for a natural number h, $C[h]$ is obtained by replacing i by h in all indexed expressions. All expressions are intended to be *modulo* n. This can be done on the syntactic level by replacing all terms and predicates X by $X \bmod n$ as follows: For a variable y and an index term $i+k$, $y[i+k] \bmod n$ is $y[(i+k) \bmod n]$; this is extended in the usual way to terms and basic predicates. For quantified predicates, we define

$\forall j\,(a \rightarrow p) \bmod n$ to be $\bigwedge_{0 \leq h < n}(a \bmod n \rightarrow p[j := h] \bmod n)$ and $\exists j\,(a \wedge p) \bmod n$ *to be* $\bigvee_{0 \leq h < n}(a \bmod n \wedge p[j := h] \bmod n)$.

The interpretation of $(i+k) \bmod n$ under a given valuation v is the natural one. Note that the instantiation of a parameterized system is an ordinary system since i is the only free index variable occurring in $C[i]$.

The following **decidability result** is crucial for the model checking procedure we present. For satisfiability of an index predicate p, all possibilities of the index terms in p to be equal/not equal to each other have to be considered. Such a choice, given by a set of equations and inequations, can be expressed in Presburger arithmetic, so it is decidable whether it can be fulfilled modulo some n. For any of these choices, it suffices to check satisfiability for the smallest such n.

Theorem 1. *Satisfiability of (open) index predicates is decidable.*

In a **specification logic** for parameterized systems, it is desirable to be able to quantify over index variables. For example, the property of mutual exclusion in the Bakery example can be formulated as: $\forall i_1 \forall i_2\,(i_1 \neq i_2 \rightarrow G\,(c[i_1] \neq C \vee c[j_2] \neq C))$. More generally, we consider formulas of the form: $\forall j_1 \cdots \forall j_n\,(a \rightarrow \phi)$, where ϕ is an LTL formula with index predicates as state formulas, where $j_1, \ldots, j_n$ are the free index variables occurring in ϕ and where a is a conjunction of inequalities $j_i \neq [i+]k$ for some index variable i.

Now we can formally state the **model-checking problem for parameterized systems:**

$$S \models \forall j_1 \cdots \forall j_m\,(a \rightarrow \phi) \quad \text{if for all } n,\ S[n], s \models_n \forall j_1 \cdots \forall j_m\,(a \rightarrow \phi) \bmod n.$$

2.2 Types of Parameterized Systems

The characterisation we give of parameterized systems for which model checking is decidable concerns mainly the communication between different copies. A possible way of communication is to read the value of a variable of a different copy. This is the case if on the right-hand side of an assignment or in a guard, an expression $y[i+k]$ occurs, where k is not zero, and hence the transition depends of the value of the variable y in the k-th neighbour. An example for this type of communication is the following, a transition from the Dining Philosophers algorithm.

$$c[i] = h \wedge pr[i] \wedge \neg pr[i-1] \wedge\ c[i-1] \neq e \wedge c[i+1] \neq e \longrightarrow \left\langle \begin{array}{l} c[i] := e \\ pr[i] := \text{false} \end{array} \right\rangle$$

While this general form of communication has to be excluded, other forms of communication, namely that of token rings and broadcast systems are unproblematic.

Both token rings and broadcast systems have a *control-state*, i.e. there is a special variable c of enumerative sort $\mathcal{D}_{control}$ such that the guard of every transition is of form $c[i] = d \wedge p$ for some index predicate p, and contains an assignment $c[i] := d'$.

A **token ring** ([EN98]) is a control-state component $C[i]$ for which some transitions are marked by *send* and some by *rec*. In all transitions of $C[i]$, only i occurs as index term, so a copy can only read the value of its own variables. With the execution of a *rec* transition, the copy acquires the token, while with a *send* transition, the token is passed to the right neighbour. We require that *send* and *rec* transitions alternate along every path through $C[i]$. Token rings are not executed in a completely asynchronous fashion as the general parameterized systems we consider here: To execute *send* or *rec* transitions, neighbouring copies have to synchronise: For any instantiation with n and some $0 \leq h < n$, a transition in $C[h]$ marked by *rec* can only be executed in parallel with a *send* transition in $C[h-1]$ (or $C[n-1]$ if $h = 0$ resp.), and vice versa. By $\mathcal{D}_{token}$ we denote the set of control states in a token ring in which the copy "has the token", i.e. which are reachable by a sequence transitions such that the last marked transition was a *rec*-transition. The initial state is required to be uniform in all indices except 0, i.e. is of form $P(0) \wedge \forall j\,(j \neq 0 \rightarrow Q(j, 0))$, where P and Q are quantifier-free expression containing only O and j, O as index terms respectively. Furthermore, we assume that in the initial state, only process 0 has the token.

A **broadcast component** ([EN98]) is again a control-state component $C[i]$, and communication between copies is only possible in form of synchronisation. But the copy to synchronise with is not determined but can be any other copy that can execute a matching transition. Moreover, a broadcast synchronisation is possible: In such a synchronisation step, all copies execute a transition. More formally, let $\Sigma = \Sigma_{rv} \cup \Sigma_{bc}$ be an *action alphabet*, where Σ_{rv} and Σ_{bc} are disjoint finite sets. The transitions can be marked by $a!$ or $a?$ for $a \in \Sigma_{rv}$, or by $a!!$ or $a??$ for $a \in \Sigma_{bc}$. The semantics of broadcast systems is as follows: A transition marked by $a!$ in some component can only be executed simultaneously with a transition marked by $a?$ in some other component and vice versa. The broadcast actions $a!!$ can only be executed simultaneously with an action marked by $a??$ in all other components, and an action marked by $a??$ can only be executed in such a situation. We assume that guards of ??-transitions do not contain quantification. (In [EN98,EFM99], the (stronger) assumption is made that for every control state d and every $a \in in\Sigma_{bc}$, some $a??$-action is executable.)

The cache coherence protocol used as example in [Del00] can be modelled as broadcast protocol as shown in Table 3, and provides an example of a broadcast protocol with global guards (in broadcast transitions).

Table 3. Illinois protocol

$V: \ c[i] : \{I, E, D, S\}$
$I: \ \forall i\,(c[i] = I)$
$(rd1??), (rd2??), (w??), (rp??) \ c[i] = I \longrightarrow \Big\langle\, c[i] = I \,\Big\rangle$
$(rd2!!) \ c[i] = I \wedge \exists j\,(c[j] = S \vee c[j] = E) \longrightarrow \Big\langle\, c[i] := S \,\Big\rangle$
$(rd1!!) \ c[i] = I \wedge \forall j\,(c[j] = I) \longrightarrow \Big\langle\, c[i] = E \,\Big\rangle$
$(rp!!) \ c[i] = I \longrightarrow \Big\langle\, c[i] = W \,\Big\rangle$
$(rd3!) \ c[i] = I \longrightarrow \Big\langle\, c[i] = S \,\Big\rangle$
$(w??), (rd1??), (rd2??) \ c[i] = D \longrightarrow \Big\langle\, c[i] = D \,\Big\rangle$
$(rd3?) \ c[i] = D \longrightarrow \Big\langle\, c[i] = S \,\Big\rangle$
$(rp??) \ c[i] = D \longrightarrow \Big\langle\, c[i] = I \,\Big\rangle$
$c[i] = D \longrightarrow \Big\langle\, c[i] = I \,\Big\rangle$
$(rd1??), (rd2??) \ c[i] = S \longrightarrow \Big\langle\, c[i] = S \,\Big\rangle$
$(w!!) \ c[i] = S \longrightarrow \Big\langle\, c[i] := D \,\Big\rangle$
$(rp??), (w??) \ c[i] = S \longrightarrow \Big\langle\, c[i] := I \,\Big\rangle$
$c[i] = S \longrightarrow \Big\langle\, c[i] := I \,\Big\rangle$
$(w??), (rd1??) \ c[i] = E \longrightarrow \Big\langle\, c[i] := E \,\Big\rangle$
$(rd2??) \ c[i] = E \longrightarrow \Big\langle\, c[i] := S \,\Big\rangle$
$c[i] = E \longrightarrow \Big\langle\, c[i] := D \,\Big\rangle$
$c[i] = E \longrightarrow \Big\langle\, c[i] := I \,\Big\rangle$

3 Model Checking by Tree Construction

Model checking of safety linear temporal logic formulas can be restricted to model checking of formulas of form $EF\,p$ for some state predicate p by using an automaton that accepts the bad prefixes for a safety linear temporal logic formula ([KV99]). The fixed point approach for verifying that S satisfies $EF\,q$ uses the fact that the set of states satisfying $EF\,q$ is the least fixed point of the functional: $F(X) = \{s \mid s \models q\} \cup \widetilde{wp}.S.X$.[2] We carry out the fixpoint computation on predicates, i.e. we compute a state predicate which characterises the least fixed point of F, starting with the empty predicate false. The necessary ingredients needed to do this are:

– Decidability of implication and satisfiability for state predicates. The former is needed to decide whether a fixed point has been reached, and the latter for deciding whether an initial state satisfies an expression.

[2] The set $\widetilde{wp}.S.X$ consists of all states that have at least on successor in X.

– For a given state predicate p, a state predicate representing $\widetilde{wp}.S.p$ must be computable.

For a program S and a given state predicate p, a predicate $\widetilde{wp}.S.p$ that represents $\widetilde{wp}.S.\{s \mid s \models p\}$ is easy to define: Let $Tr(C)$ be the set of all transitions of component C, where a transition is of form $g \longrightarrow < v_0 := t_0, \ldots, v_k := t_k >$, then $\widetilde{wp}.S.p = \bigvee_{C \text{ comp. of } S} = \bigvee_{C \text{ comp. of } S} \bigvee_{g \longrightarrow < v_0 := t_0, \ldots, v_k := t_k > \in Tr(C)} (g \wedge p[v_0 := t_0, \ldots, v_k := t_k])$, where $[v_0 := t_0, \ldots, v_k := t_k]$ denotes simultaneous substitution.

The specific feature of our algorithm is that the fixed point computation is represented in form of a tree over $\mathbb{N}$, i.e. as set of finite sequences over $\mathbb{N}$ that is closed under prefix-formation: The root is denoted by ϵ, and e. g. 00 is the first successor of the first successor of the root. Note that the algorithm works directly on the program notation.

Definition 2 (Proof tree construction). *Let q be a predicate of the form $\bigwedge_i p_i$ for literals p_i. The proof tree $\mathcal{T}(EF\,q, S)$ is a tree over $\mathbb{N}$ with labelling $l : \mathcal{T}(EF\,q, S) \longrightarrow \{ \bigwedge_i p_i \mid p_i$ literals $\}$ such that*

– $l(\epsilon) = q$ *and*
– $\bigvee_j l(xj) \Leftrightarrow \widetilde{wp}.S.l(x)$.[3]

A node $x \in \mathcal{T}(EF\,q, S)$ is a leaf if one of the following conditions holds:
(i) $l(x)$ is not satisfiable (unsuccessful leaf);
(ii) $I \wedge l(x)$ is satisfiable (where I is the initial predicate of the system), i.e. there is an initial state of S satisfying $l(x)$ (successful leaf);
(iii) there is a node $y \in \mathcal{T}(EF\,q, S)$ such that $|x| > |y|$ and $l(x) \Rightarrow l(y)$ (unsuccessful leaf).

The tree $\mathcal{T}(EF\,q, S)$ is *successful* if it has a successful leaf. The following theorem states the correctness of the algorithm.

Theorem 2. *(a) $S \models AG\,(\neg q)$ if and only if $\mathcal{T}(EF\,q, S)$ is not successful.*
(b) If all sorts of variables of S are finite, then the construction of $\mathcal{T}(EF\,q, S)$ terminates.

To illustrate the tree construction, the first steps of the proof tree construction for the 2-component Bakery algorithm are shown in Figure 1. For the root, $\widetilde{wp}.S.(c_1 = C \wedge c_2 = C)$ is $(c_1 = W \wedge (n_1 < n_2 \vee n_2 = 0) \wedge c_2 = C) \vee (c_1 = C \wedge (n_2 < n_1 \vee n_1 = 0) \wedge c_2 = W)$, and by transforming this into disjunctive normal form, the four successors of the root node are obtained.

An advantage of using a tree structure to represent the least fixed point is that the predicates labelling the nodes are relatively small and automatically only elements of the frontier set are considered for the next iteration step. This distinguishes the approach from other approaches to use Presburger arithmetic for representing sets of states during the fixed point iteration [BGP97] and makes it easier to check the Conditions (i), (ii) and (iii). Producing counterexamples is easy since the paths in the tree form potential refutation sequences.

[3] It is irrelevant for the correctness which representation in disjunctive normal form is chosen.

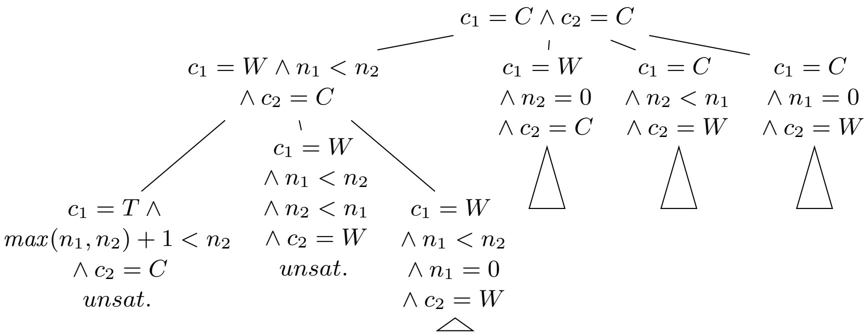

Fig. 1. Proof tree for the 2-component Bakery algorithm

4 Model Checking for Parameterized Systems

4.1 Adaption of the Tree Construction to Parameterized Systems

In order to apply the tree construction to parameterized systems, a definition of $\widetilde{wp}.S.p$ for a parameterized system S has to be provided. For an ordinary parameterized system S and a given instantiation with n, and for an index predicate p without free index variables, $\widetilde{wp}.S[n].p$ can be define as on page 8, by using a disjunction over all components, i.e. instantiated copies of $C[i]$. This easily generalises to token rings or broadcast systems by considering all possibilities of synchronisation.

If p contains free index variables, it is possible to represent $\bigvee_{0 \le h < n} \widetilde{wp}.C[h].p$ uniformly in n by an index predicate that contains the free index variables that occur in p. Such a representation can be obtained by computing $\widetilde{wp}$ with respect to copies $C[t]$, instantiated by index terms t that occur in p. We can define $\widetilde{wp}.C[t].p$ by considering index terms as constants and use for example the definition on page 8. This is however only correct if for all index terms t and t' in p, p implies $t \neq t'$. So it is necessary to consider all possibilities of dividing the index terms in p in those that are equal to t and those that are not equal to t, and to compute $\widetilde{wp}.C[t].p$ for each of those possibilities. To make this precise, we need some notation. Let $t_1, \ldots, t_n$ be the free index terms (i.e. index terms $i + k$ such that i is free) occurring in p. For an ordinary parameterized system, we define

- $G(p) \stackrel{\text{def}}{=} \{Equ(t_{i_1}, \ldots, t_{i_k}) \mid \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}\}$, where
- $Equ(t_{i_1}, \ldots, t_{i_k}) \stackrel{\text{def}}{=} t_{i_1} = t_{i_2} \wedge \cdots \wedge t_{i_1} = t_{i_k} \wedge \bigwedge_{r \in \{1,\ldots,n\} \setminus \{i_1,\ldots,i_k\}} t_{i_1} \neq t_r$, [4],
- $Equ(\emptyset) = \bigwedge_{1 \le r \le n} i^* \neq t_r$, where i^* is a new index variable.

For an element $g \in G(p)$, let $rep(g)$ be t_{j_1} or i^* respectively. For token rings, in addition sets of index terms that equal $rep(g) + 1$ or $rep(g) - 1$ have to be

[4] Note that $i + k = i + k'$ can be satisfiable in some instantiation for even if $k \neq k'$.

selected, and for broadcast systems, a set of index terms that are interpreted as the index of the component that is chosen for synchronisation has to be chosen.

For an element $g \in G(q)$, the index predicate $q\langle g\rangle$ is the result of replacing all index terms that are required by g to be equal to $rep(g)$, by $rep(g)$.

So for a parameterized system S, let

$$\widetilde{wp}.S.q \stackrel{\text{def}}{=} \bigvee_{g\in G(q)} (g \wedge \widetilde{wp}.C[rep(g)].q\langle g\rangle).$$

If guards of S contain quantified expressions, labels in the tree can contain quantified expressions as well. We can assume that in this case, labels are of form

$$P(t_1,\ldots,t_l) \wedge \forall j\, (j \neq t_1,\ldots,t_l \to Q),$$

where P is conjunction of literals that only contain index terms in $t_1,\ldots,t_l$. This form of labels can be obtained as follows:

Whenever a universally quantified expression appears in a successor label (as part of a guard), it is instantiated for all free index terms that occur in the label.

Furthermore, an existentially quantified expression that appears in a label, can be removed by introducing new index variables as follows: A label

$$P(t_1,\ldots,t_l) \wedge \exists j\, (a \wedge E) \wedge \forall j\, (j \neq t_1,\ldots,t_l \to Q)$$

is replaced by the set of labels

$$\begin{aligned}\{P(t_1,\ldots,t_l) \wedge E[j := t_i] \wedge \forall j\, (j \neq t_1,\ldots,t_l \to Q) \mid 1 \leq i \leq l\}&\\ \cup \{P(t_1,\ldots,t_l) \wedge E[j := i'] \wedge \forall j\, (j \neq t_1,\ldots,t_l \to Q)\}&\end{aligned}$$

where i' is a new index variable.

The definition of $\langle g\rangle$ has to be adapted to labels with universally quantified expressions: In case $rep(g)$ is the new variable i^*,

$$\forall j\, (j \neq t_1,\ldots,t_l \to Q)\langle g\rangle = Q[j := i^*]\langle g\rangle \wedge \forall j\, (j \neq t_1,\ldots,t_l, i^* \to Q\langle g\rangle).$$

Otherwise,

$$\forall j\, (j \neq t_1,\ldots,t_l \to Q)\langle g\rangle = \forall j\, (j \neq t_1,\ldots,t_l \to Q\langle g\rangle).$$

The new variable i^* has to be introduced to represent the choice that a copy $C[h]$ takes a step, where none of the free index terms in the label are interpreted as h. So after this transition, $Q[j := h]$ has to hold. Note that if a label L does not contain a universally quantified expression, then introducing the new variable i^* in some $g \in G(p)$ can be omitted since for quantifier-free L, $\widetilde{wp}.C[i^*].L\langle g\rangle$ is equivalent to L.

The following lemma states that $\widetilde{wp}.S[n].p$ is correctly represented by the definition above. Note that $p[\{j := v(j) \mid j \text{ free index variable in } p\}] \bmod n$ does not contain free index variables and can hence be considered as ordinary predicate over the system variables of $S[n]$.

Lemma 1. *For all n, states s and valuations v with respect to n: There is a valuation w of the index variables introduced in $G(q)$ s.t. $s, v+w \models_n \widetilde{wp}.S.q$ iff there is a successor s' of s such that $s', v \models_n p \bmod n$.*

So we can now define the adaption of the proof tree construction for parameterized systems.

Definition 3 (Proof tree construction for parameterized systems). *Let p be an index predicate. $\mathcal{B}^{para}(EF\, p, S)$ is a tree over $\mathbb{N}$ with labelling $\bigwedge_i p_i$, where p_i is either a quantifier-free index predicate which is a literal, or a quantified index predicate.*

- $l(\epsilon) = p$ *and*
- $\bigvee_j l(xj) = \widetilde{wp}.S.l(x)$.

A node $x \in \mathcal{B}^{para}(EF\, p, S)$ is a leaf if one of the following holds:
(i) $l(x)$ is not satisfiable (unsuccessful leaf);
(ii) $I \wedge l(x)$ is satisfiable (successful leaf);
(iii) there is a node $y \in \mathcal{B}^{para}(EF\, q, S)$, s.t. $|x| > |y|$ and

- *$l(y)$ does not contain a quantified expression, and $\exists\, l(x) \to \exists\, l(y)$ holds, where $\exists$ denotes quantification over all free index variables, or*
- *there is a renaming σ of the free index variables in $l(x)$ such that $l(x)[\{i := \sigma(i) \mid i\}] \to l(y)$ holds.*

The second part of Condition (iii) is a special instance of $\exists\, l(x) \to \exists\, l(y)$. By Theorem 1, both requirements are decidable.

4.2 Parameterized Systems for which Model Checking of Safety Properties is Decidable

By exploring under which conditions the proof tree construction terminates, we obtain a characterisation of parameterized systems for which model checking of safety properties is decidable.

Termination can only be obtained by showing that Condition (iii) holds eventually along a path of the proof tree. Requirements for this are that domains of system variables are finite, which we assume, and furthermore that for index variables j, only bounded many index terms of form $j+k$ occur in labels of the proof tree

The latter requirement does not hold for parameterized systems $C[i]$ in which expressions of form $i+k$ for $k \neq 0$ occur in guards or right-hand sides of assignments, as in the example on page 5, because such transitions can cause the introduction of index terms $i+k$ for unbounded many k during the tree construction. In this situation, there can be infinitely many node labels that do not imply each other. In fact, the coding of a Turing machine in [Suz88] is in terms of systems that use (exclusively) this form of communication.

In what follows, we restrict ourselves to systems that have the property that for every index variable j, only bounded many index terms of form $j + k$ occur in the labels of the tree.

Although token rings do use communication with neighbours, it is possible to stop the tree construction at labels $j + k - 2$, where is k is minimal such that $i + k$ occurs in some guard or the property to verify.

Furthermore, the construction can be stopped at labels that have more than one token, i.e. contain literals of form $c[t] = d$ and $c[t'] = d'$ for $d, d' \in \mathcal{D}_{token}$, where c is the control-state variable of $C[i]$

Application of these two new termination conditions cuts off successful nodes only if the smaller tree is already successful:

Let $k_1, \ldots, k_r$ be all constants (in linear ascending order) such that an index term $j + k_i$ occurs in the root label.

Since we are only considering system variables with finite domain, we can assume that all domains except that of the control-state variable c are boolean, and hence assume that a label consists, besides the quantified expression, of expressions of form:

$$(\star)\ P^{-1}(i + k_r + 1) \wedge P(i + k_1, \ldots, i + k_r) \wedge P^1(i + k_1 - 1) \wedge \cdots P^m(i + k_1 - m)$$

for every free index variable i, where $P^h(t)$ are conjunctions of literals that contain only the the index term t, and $P(t_1, \ldots, t_n)$ is a conjunction of literals only containing $t_1, \ldots, t_n$ as index terms. We can ignore labels with index terms $i + k_r + k$ for $k > 1$ since it can be observed that such labels necessarily contain several tokens.

Furthermore, it can be observed that for $1 \leq h < m$, P^h contains a literal $c[i+k_1-h] = d$ such that $d \notin \mathcal{D}_{token}$, and that P^m contains $c[i+k_1-m] = d$ for some $d \in \mathcal{D}_{token}$. In the computation of $\widetilde{wp}$ for a label L, synchronising actions have only to be considered with respect to $C[i + k]$ and $C[i + k - 1]$, where k is the smallest constant k' for which $i + k'$ occurs L, since for synchronising actions involving $C[i + k']$ for other index terms $i + k'$, labels containing more than one token are produced. This implies that the proof tree also contains a label

$$(\star\star)\ P^{-1}(i + k_r + 1) \wedge P(i + k_1, \ldots, i + k_r) \wedge P^m(i + k_1 - 1).$$

Under the restrictions on initial predicates for token rings, it follows that if some initial state satisfies $(\star)$ then there is also an initial state satisfying $(\star\star)$.

So after restricting to systems that have the property that for every index variable j, only bounded many index terms of form $j + k$ occur in the labels of the tree, the following types of parameterized systems are left for consideration, which we call *systems with bounded index terms.*

- Interleaving systems for which the index terms used in guards do not contain constants. These systems can communicate by global variables and quantified guards.
- Token rings; communication can be by synchronized send and receive actions as well as by global variables and quantified guards.

– Broadcast protocols; communication can be by handshake actions or broadcast actions and by global variables and quantified guards.

For systems with bounded index terms that do not contain universal quantification in guards, Condition (iii) holds eventually along each path. So we obtain the following result, that extends the known result of [EN98,EFM99,EN95].

Theorem 3. *Model checking of $\forall\, j_1, \ldots, j_m\, (a \to G\, q)$, where q is a quantifier-free index predicate, is decidable for systems with bounded index terms that contain only existential quantification in guards.*

Next we consider the case that universally quantified expressions occur in guards. As explained above, if quantified expressions occur in labels, the representation of the successor labels requires introduction of new index variables, and while existential quantifiers can be removed, this is not true for universal quantifiers. So if universal quantifiers occur, labels are of form $P(t_1, \ldots, t_l) \wedge \forall\, j\, (j \neq t_1, \ldots, t_l \to Q)$, where there is no bound on the number of different index variables that can occur as $t_1, \ldots, t_l$. For such labels, Condition (iii) does not need to hold eventually, so termination of the tree construction is not guaranteed.

But it can be observed that for systems for which guards do not contain existential quantification or global variables, introduction of new index variables is only necessary for a bounded number, more precisely the number n of free and existentially quantified index terms in the initial predicate of the system. This is because if for some instantiation there is a path from an initial state to the root labe,l then this path also exists for an instantiation with $n + k$ copies, where k is the number of index terms occurring in the property:

Assume s, v is a model that satisfies the initial predicate and some label L, and let s', v' be the result of projecting s, v' to the component indices h that are used to interpret free or existentially quantified variables in the initial predicate or the label L. Then a state satisfying the root label is reachable from s' as well, since without existentially quantified guard and without global variables, the same transitions can be executed as the ones that lead from s to a state satisfying the root label.

When only finitely many new index variables are introduced, again Condition (iii) eventually holds along a path, so we obtain:

Theorem 4. *For parameterized systems with bounded index terms (but not broadcast protocols) that do not contain global variables and do not contain existential quantification in guards, model checking of $\forall\, j_1, \ldots, j_m\, (a \to G\, q)$ is decidable, where q is an index predicate not containing global variables.*

We have to exclude broadcast protocols because handshake synchronization implicitely expresses existential quantification.

The Bakery algorithm provides an example of a system to which Theorem 4 can be applied: It is not necessary to introduce new index variables during the tree construction for the Bakery example. Figure 2 displays part of this tree construction, the full tree contains 23 nodes.

Generalising expressions We now turn to the case that universal quantification in guards is combined with existential quantification, global variables or handshake communication in broadcast protocols. This requires to generalise labels to sets of labels.

Since we prove termination only for systems where system variables have finite domain, we can restrict considerations to systems that have only boolean variables. This means that we can assume that quantified expressions occurring in guards are of form $\forall j\, (j \neq i \to Q(j))$ or $\forall j\, (Q(j))$, i.e. the quantified expression contains only j as free index variable.

It then follows that labels L of a proof tree are of form

$$P^1(I_1) \wedge \cdots \wedge P^n(I_n) \wedge \forall j\, (j \neq I_1, \ldots, I_n \to Q(j))$$

where I_j are disjoint sets of index variables, Q is a quantified expression that only contains j as free index variable and every P^h is a conjunction of literals that contain the same index variable exclusively. $P^h(I_j)$ stands for the conjunctions of all instantiations of P^h with some $i \in I_j$.

A label

$$L' = P^1(I'_1) \wedge \cdots \wedge P^n(I'_n) \wedge \forall j\, (j \neq I'_1, \ldots, I'_n \to Q(j))$$

is an *extension* of a label

$$L = P^1(I_1) \wedge \cdots \wedge P^n(I_n) \wedge \forall j\, (j \neq I_1, \ldots, I_n \to Q(j))$$

if $|I_j| \leq |I'_j|$ for all j. Let $X(L, L')$ be the set containing $|I'_j \setminus I_j|$ many expressions P_j for each j.

The **generalisation** of L' extending L is

$$P^1(I_1) \wedge \cdots \wedge P^n(I_n) \wedge (X(L, L'))^{\geq 0} \wedge \forall j\, (j \neq I_1, \ldots, I_n \to Q(j)),$$

standing for the set of set of expressions that are obtained by adding $X(L, L')$ some number of times to the expression.

The proof tree construction has to be adapted for labels that contain generalisations: Not only quantified expressions, but also generalised expression X^n have to be instantiated by the $\langle g \rangle$-operation.

Note that a label of form

$$L = P^1(I_1) \wedge \cdots \wedge P^n(I_n) \wedge (X_1)^{m_1} \wedge \cdots \wedge (X_k)^{m_k} \wedge QE$$

such that $P^{i_1}, \ldots, P^{i_l} = X_h$ for some $i_1 \in I_1, \ldots, i_l \in I_l$ can be transformed to

$$P^1(I'_1) \wedge \cdots \wedge P^n(I'_n) \wedge (X_1)^{m_1} \wedge \cdots \wedge (X_h)^{m_l+1} \wedge \cdots \wedge (X_k)^{m_k} \wedge QE,$$

where $I'_j = I_j \setminus \{i_1, \ldots, i_l\}$.

So for all labels L in the proof tree, we can assume that the expressions $P^1, \ldots, P^n$ do not form a superset of any generalised expression X^n of L. This implies that for all generalised expressions X and X' in a label, X is not an extension of X', which means that there are only finitely many possibilities for extensions.

Termination is guaranteed for systems with bounded index terms since if $m_j \geq m'_j$ for all j,

$$L = P^1(I_1) \wedge \cdots \wedge P^n(I_n) \wedge (X_1)^{m_1} \wedge \cdots \wedge (X_k)^{m_k} \wedge QE$$

implies

$$L = P^1(I_1) \wedge \cdots \wedge P^n(I_n) \wedge (X_1)^{m'_1} \wedge \cdots \wedge (X_k)^{m'_k} \wedge QE.$$

So we obtain:

Theorem 5. *When applying all possible generalisations during the tree construction for parameterized systems with bounded index terms, the construction terminates.*

However, introducing generalisations is only correct, i.e. does not lead to over-approximation (false negatives), if all elements of a generalisation would occur in the full tree constructed without generalisations. This is the case if between L and L', no universally quantified expressions were introduced by the $\widetilde{wp}$ computation, and the system does not contain broadcast actions.

If there is only one quantified expressions $\forall j\, (j \neq i \rightarrow Q(j))$ (or $\forall j\, (Q(j))$ resp.) occurring in the guards of the system, then generalisation can be restricted to the case that between L and L', no universally quantified expressions were introduced by the $\widetilde{wp}$ computation, without loosing termination:

Along a path where $\forall j\, (j \neq i \rightarrow Q(j))$ is introduced infinitely often by the $\widetilde{wp}$ computation, Condition (iii) eventually holds, since after introduction of $\forall j\, (j \neq i \rightarrow Q(j))$, for all expressions P^h in a node except one, $P^h(i) \rightarrow Q(i)$. So if a node in which $\forall j\, (j \neq i \rightarrow Q(j))$ was newly introduced is extending a previously constructed node, Condition (iii) holds.

So when restricting to systems that only contain one quantified expression, applying generalisation in safe cases suffices to guarantee termination:

Theorem 6. *For parameterized systems with bounded index terms that contain at most one universally quantified expression in guards (which can have several occurrences), models checking of $\forall j_1, \ldots, j_m\, (a \rightarrow G\, q)$ is decidable, where q is quantifier-free. For broadcast protocols, in addition it is required that only handshake actions occur.*

By using generalization only in cases where it does not produce an over-approximation, i.e. if in between the labels N_1 and N_2 for which generalisation is introduced, no universal quantification is contained in an instantiated guard and no $\tilde{wp}$-computation is with respect to a broadcast transition, it can be shown that $\forall j_1, \ldots, j_m\, (a \rightarrow G\, q)$ is decidable for the examples considered by Delzanno in [Del00]. These examples, among them the Illinois protocol have the following characteristic: If there is a universally quantified expression $\forall j\, (j \neq i \rightarrow (c[i] = d_1 \vee \cdots \vee c[i] = d_m))$ in some guard, where $d_1, \ldots, d_m$ are control locations, then for every control location $d \neq d_1, \ldots, d_m$ there is a local, unsynchronised transition from d to some $d_1, \ldots, d_m$. This means that every label $N =$

$$P^1(I_1) \wedge \cdots \wedge P^n(I_n) \wedge \forall\, j\,(j \neq i \rightarrow (c[i] = d_1 \vee \cdots \vee c[i] = d_m))$$

can be generalized to $N' =$

$$P^1(I_1) \wedge \cdots \wedge P^n(I_n) \wedge (d_1')^{\leq 0} \wedge \cdots \wedge (d_k')^{\leq 0} \wedge \forall\, j\,(j \neq i \rightarrow (c[i] = d_1 \vee \cdots \vee c[i] = d_m)),$$

where $d_1', \ldots, d_k', d_1, \ldots, d_m$ are all control locations of the protocol. But existential quantification over N' is equivalent to just $P^1(I_1) \wedge \cdots \wedge P^n(I_n)$.

In the tree construction for the Illinois protocol, shown in Figure 3 for the property $\forall\, j_1, j_2\,(j_1 \neq j_2 \rightarrow G\,(c[j_1] \neq E \vee c[j_2] \neq E))$, it is however not necessary to use generalisation in order to obtain termination: The nodes that contain universal quantification are unsatisfiable. The three successor labels of the root arise as follows: The leftmost successor is $\widetilde{wp}$ with respect to an $rd1!!$-action of component $C[j_1]$, the next successor is with respect to an $rd1!!$-action of component $C[j_2]$, and the third successor corresponding to an $w!!$-action of component $C[j_3]$ for a new index variable j_3.

5 Conclusion

We introduced an algorithm for model checking safety properties that is applicable to a range of parameterised systems. Analysing under which conditions on parameterised systems this algorithm terminates lead to an extensions of known decidability results, and to a characterisation of a class for which termination can be forced by allowing overapproximations. A severe restriction of the systems we considered is that quantification can only be restricted by inequalities: so guards that contain expressions $\forall\, j\,(j < i \rightarrow ..)$ are excluded. Systems containing such guards are for example explored in [ABJN99]. How our approach can be extended to such guards is clearly a topic for further research.

References

[ABJN99] Abdulla, P. A., Bouajjani, A., Jonsson, B. and Nilsson, M. *Handling global conditions in parameterized system verification.* In: *Proc. 11th Intl. Conf. on Computer Aided Verification.* 1999, LNCS 1633.

[ACJT96] Abdulla, P., Cerans, K., Jonsson, B. and Tsay, Y.-K. *General decidability theorems for infinite-state systems.* In: *Proc. 11th Symp. Logic in Computer Science.* 1996.

[BGP97] Bultan, T., Gerber, R. and Pugh, W. *Symbolic model checking of infinite state systems using Presburger arithmetic.* In: *Proc. 9th Intl. Conf. on Computer Aided Verification.* 1997, LNCS 1254.

[BJNT00] Bouajjani, A., Jonsson, B., Nilsson, M. and Touili, T. *Regular model checking.* In: *Proc. 12th Intl. Conf. on Computer Aided Verification.* 2000, LNCS 1855.

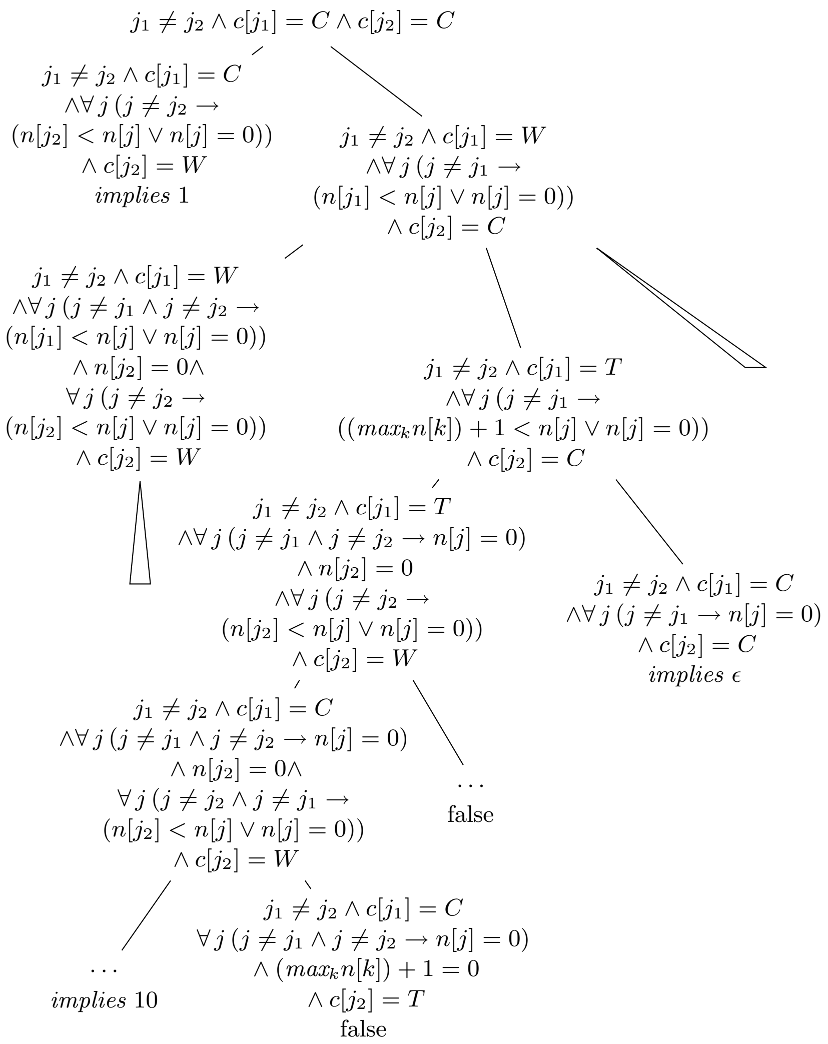

Fig. 2. Proof tree for the parameterized bakery algorithm

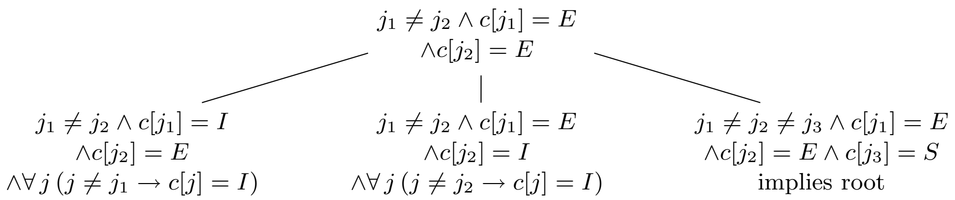

Fig. 3. Tree construction for the Illinois protocol

[BW98] Boigelot, B. and Wolper, P. *Verifying systems with infinite but regular state space*. In: *Proc. 10th Intl. Conf. on Computer Aided Verification*. 1998.

[Del00] Delzanno, G. *Automatic verification of parameterized cache coherence protocols*. In: *12th Intl. Conf. on Computer Aided Verification*. 2000, LNCS 1855.

[EFM99] Esparza, J., Finkel, A. and Mayr, R. *On the verification of broadcast protocols*. In: *Proc. 14th Symp. Logic in Computer Science*. 1999.

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

[EN98] Emerson, E. A. and Namjoshi, K. S. *On model checking for non-deterministic infinite-state systems*. In: *Proc. 13th Symp. Logic in Computer Science*. 1998.

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

[KMM+97] Kesten, Y., Maler, O., Marcus, M., Pnueli, A. and Shahar, E. *Symbolic model checking with rich assertional languages*. In: *Proc. 9th Intl. Conf. on Computer Aided Verification*. 1997, LNCS 1254.

[KV99] Kupferman, O. and Vardi, M. Y. *Model checking of safety properties*. In: *Proc. 11th Intl. Conf. on Computer Aided Verification*. 1999, LNCS 1633.

[Suz88] Suzuki, I. *Proving properties of a ring of finite state machines*. Information Processing Letters, 28: 213–214, 1988.