RWTH Aachen

Department of Computer Science
Technical Report

On Compositional Failure Detection in Structured Transition Systems

Ingo Felscher and Wolfgang Thomas

ISSN 0935–3232 · Aachener Informatik-Berichte · AIB-2011-12

RWTH Aachen · Department of Computer Science · August 2011

On Compositional Failure Detection in Structured Transition Systems

Ingo Felscher and Wolfgang Thomas

Lehrstuhl für Informatik 7
RWTH Aachen University, Germany
Email: {felscher,thomas}@automata.rwth-aachen.de

Abstract. In model-checking, systems are often given as products. We propose an approach that is built on a preprocessing of specifications in terms of appropriate automata. This allows to incorporate information about the local behaviour and synchronization of the system components into the specification. We develop a framework of (partially) synchronized automaton products and a format of corresponding specification automata that allows for a compositional failure detection of linear regular properties (either for finite or for infinite behaviour). As a result we obtain an algorithm which separates the local and the non-local segments of system runs, resulting in improved complexity bounds in typical specifications.

Keywords: model-checking; finitely synchronized products; compositional failure detection.

1 Introduction

In model-checking we examine whether a given system, normally modelled as a transition system, satisfies a specification, modelled as a logic formula. The systems under investigation often arise as products composed of several components – again transition systems – that may interact with some or all other components and may also perform actions independently of the other components. The main problem in this scenario is the question of state space explosion, studied in a large body of literature, see e. g. [BK08].

The basic problem is to separate aspects of the specification that are local (to the components) from each other and from synchronizing features. This is a natural idea which is also familiar from the "composition method" of algorithmic model theory. The method allows to deduce the truth of a formula in a product from information about the truth value of formulas in the components. In model theory this approach was initiated in the pioneering paper [FV59] of Feferman and Vaught and further developed by numerous authors [CK90,GS98,Hod93,Mak04,Mos52,She75,Tho97a]. For more recent results, now in the field of model-checking, see [Fel08,FT09,Rab07,WT04]. The complexity of this compositional approach is excessive (in fact, non-elementary in the size of the given formula – even for modal logic and first-order logic), due to the large number of auxiliary formulas that have to be constructed. At least for first-order logic this effect is known to be unavoidable [DGKS07]. (Apart from this, the classical approach is restricted to (variants of) first-order logic. Already for modal logic extended by the logical operator EG and for first-order logic with regular reachability predicates the composition method fails [Rab07,WT04].)

In the present paper we offer a compositional analysis of reachability (motivated by failure detection) that may lead to a considerable reduction of the high complexity as known from the logical framework. Our approach relies on automata theoretic specifications. The present paper is an extended and corrected version of [FT11].

Another advantage of the insistence on automata as specification formalism is the avoidance of the initial conversion of a given logic formula to an automaton. In most cases (e. g., for temporal, first-order or monadic second-order logic), the costs of this conversion of formulas are exponential (or much more) in time complexity. (It is well-known that an *MSO(<)* formula can be translated into an equivalent automaton [Büc60,Elg61] and that the complexity of this translation is non-elementary [AHU74,Tho97b].)

In the present work we start with the description of undesired behaviour using a "complement specification", denoted $\overline{Spec}$. The given system is a partially synchronized product Sys with (binary, labeled) relations and (unary) predicates. We split $\overline{Spec}$ into parts which can be checked in the individual components. For purpose of exposition, we first consider the case of $\overline{Spec}$ where unary predicates are missing, and then treat the general case.

The general idea is to split the complement specification automaton into parts (called "local blocks") each of which has only labels and predicates from a fixed set of components. As result we then get the local blocks (as mentioned above, as specification automata which can be checked in the individual components) and a "global specification automaton" *Glob* which describes the possible concatenations of these local block automata. For this, information about the synchronization behaviour of a transition is used: the "synchronization profile". This profile specifies which of the components are synchronized via the transition's label. In the runs of *Glob*, sequences of transitions with the same synchronization profile are grouped together.

In the main result, first stated for the case without unary predicates, the question whether a product of transition systems and a given complement specification automaton have paths with a common labeling is reduced to the question whether a path in the global specification automaton exists such that the components and parts of the complement specification which are described by the local blocks of the global specification automaton have paths with common labeling. The presented result is limited to complement specifications with "bounded synchronization", i. e. specifications which can use only a bounded number of synchronization transitions of the same profile in a sequence. However, in Section 6 we add some remarks how this restriction can be avoided.

For the generalization of the result to specifications with predicates we first linearize the transition system Sys and the complement specification $\overline{Spec}$: We code the predicates of states in labeled self-loops of these states and thus dissolve, for example, the fulfillment of predicates $p_1, \neg p_2, p_3$ at a state s into the subsequent execution of self loops at s, labeled $p_1, \neg p_2, p_3$. If an action move c is executable at s, the corresponding c-transition may be taken after the mentioned self-loops.

The terminological complexity of a compositional framework as developed here is considerable – an unavoidable feature also known from the literature above. As a gain of this effort, we will show that the algorithm derived from this automaton composition method will only be double exponential ($2^{2^{r^2}}$) in the number of states r the largest component has. Further it will be exponential in the number of components and of states and predicates the complement specification has.

Of course, a drawback is the necessity of preprocessing when a logical specification is given. However, in many practical situations, when specifications are short, this preprocessing can often be done efficiently in spite of the exponential standard algorithms [Wol01]. In other cases, one might be able to use automata theoretic specifications directly.

The applicability of our method depends on an appropriate set-up of the specification automaton: It should offer as much as possible the potential to separate the various local and synchronized computations. Of course, in the worst case as represented by always fully synchronized transitions, the decomposition does not pay (since only blocks of length one are formed).

The paper is structured as follows: After this introduction we present in Sect. 2 technical preliminaries. For this, we show our notion of a synchronized product and the complement specification automaton. These definitions are then used in Sect. 3 in the main result for specifications without predicates. We further add a sketch how this result can be generalized to infinite behaviours captured by Büchi automata. In Sect. 4 we introduce definitions and notions needed for the generalization of the main result which is shown in Sect. 5. This generalization allows to capture specifications with unary predicates. We conclude the paper in Sect. 6 with a summary and some remarks on open problems.

2 Technical Preliminaries

In this section we introduce the basic definitions: In Sect. 2.1 we treat products of transition systems and in Sect. 2.2 the automaton models used for the complement specification and its transformation into the global automaton. In Sect. 2.3 we define transition profiles as in [KN01]. Based on these profiles we introduce in Sect. 2.4 the notion of an effect of an automaton $\mathcal{A}$ on an automaton $\mathcal{B}$ – the state changes of $\mathcal{A}$ that are possible by words in the language of $\mathcal{B}$.

2.1 Products of Transition Systems

A *transition system* is a labeled graph $K = (S, \{R_a \mid a \in \Sigma\}, \{P_v \mid v \in V\})$ with state set S, transition relations $R_a \subseteq S \times S$ and predicates $P_v \subset S$.

We introduce our notion of a synchronized product with asynchronous and synchronous behaviour: *Synchronized transitions* are transitions which are taken at the same time in a subset of the components – captured by the "synchronization profile" – and independently of the transitions of the other components. *Asynchronous transitions* are taken independently of the transitions in all other components. They can be seen as synchronized with a trivial synchronization profile (i. e. a synchronization profile which contains only one component index).

From now on, we use $[m]$ for $m \in \mathbb{N}$ as an abbreviation for the set $\{1, \ldots, m\}$.

Definition 1 (Synchronized product). *Let $[n]$ be a finite set of indices and Σ an alphabet of labels (for the transitions) and $V := \{v_1, \ldots, v_l\}$ a set of names*

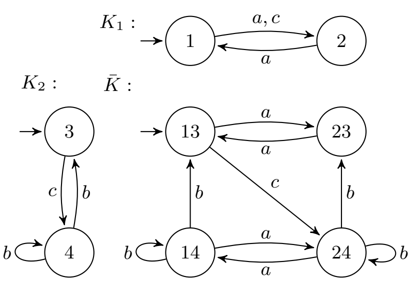

Fig. 1. Components K_1, K_2 and their product $\overline{K}$

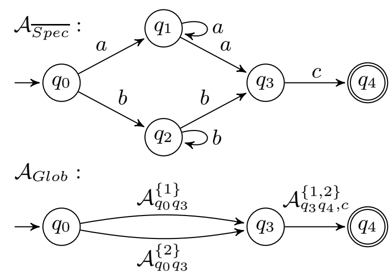

Fig. 2. Complement specification and Global automaton

for the unary predicates. For $i \in [n]$ let a component transition system K_i be of the form $K_i = (S_i, \{R^i_c \mid c \in \Sigma\}, \{P^i_v \mid v \in V\})$ as mentioned above.

A synchronization profile $sp(c)$ for $c \in \Sigma$ defines which components are synchronized via c-transitions and is formally defined as $sp : \Sigma \to Pot([n]), c \mapsto \{i \mid |R^i_c| \neq \emptyset\}$.
The synchronized product $\mathcal{A}_{Sys}$ of the components K_i is defined as the transition system $\overline{K} := (\bar{S}, \{\bar{R}_c \mid c \in \Sigma\}, \{\bar{P}_{v^i} \mid v \in V\})$ where

- *the state set $\bar{S}$ is the product of the component state sets: $\bar{S} := \prod_{i \in [n]} S_i$. (We write $\bar{s}[i]$ for the state of the i-th component of $\bar{s} \in \bar{S}$.)*
- *the synchronized transition relation $\bar{R}_c$ is defined by $(\bar{x}, \bar{y}) \in \bar{R}_c$ iff $\forall i \in sp(c)$: $(\bar{x}[i], \bar{y}[i]) \in R^i_c$ and $\forall j \in [n]$ with $j \notin sp(c)$: $\bar{x}[j] = \bar{y}[j]$.*
- *the predicate $\bar{P}_{v^i}$ is the set $\{\bar{s} \mid \bar{s}[i] \in P^i_v\}$.*

Example 1. In Fig. 1 we show a synchronized product $\overline{K}$ of two components K_1, K_2 with asynchronous a- and b-transitions in K_1, respectively K_2, and synchronized c-transitions with synchronization profile $\{1, 2\}$. For better readability, the state names of K_1, K_2 are chosen differently.

2.2 Automata

In this section we introduce the format of complement specification automata for a given synchronized product. They are used to express properties that lead to a failure in the product. Afterwards, we translate the complement specification automaton into a "global specification automaton". For this, the complement specification is split into parts that can be checked in the synchronization profiles.

We restrict ourselves to complement specifications that are able to check only a bounded number of synchronized transitions of the same non-trivial synchronization profile in a sequence. Note that the definitions in this section do not treat unary predicates of a product yet. How to cope with the predicates will be shown in Sect. 4 and 5.

Let us recall usual finite automata to fix notation. A *(non-deterministic) finite automaton* is defined by $\mathcal{A} := (Q, \Sigma, \Delta, q_0, F)$ with finite state set Q, input alphabet Σ, transition relation $\Delta \subseteq Q \times \Sigma \times Q$, initial state $q_0 \in Q$ and final state set $F \subseteq Q$.

A *(bounded synchronized) complement specification automaton* is a finite automaton, compatible with the alphabet of the transitions of the product, which uses only fixed numbers of synchronization transitions in a sequence.

Definition 2. *A (bounded synchronized) complement specification automaton without predicates is a finite automaton $\mathcal{A}_{\overline{Spec}} = (Q, \Sigma, \Delta_{\overline{Spec}}, q_0, F)$ which is compatible with the action alphabet of the synchronized product and fulfills the following property: for every synchronization profile $sp(c), c \in \Sigma$ which is not a singleton $\{i\}, i \in I$ the restriction of $\mathcal{A}_{\overline{Spec}}$ to all transitions of this synchronization profile $sp(c)$ contains no loops.*

The *global specification automaton* of a complement specification combines subsequent transitions of the same synchronization profile. Such a combination will result in "super"-transitions labeled from a *local block alphabet*: the alphabet of all sub-automata $\mathcal{A}^I_{q,q'(,w)}$ of $\mathcal{A}_{\overline{Spec}}$ where $\mathcal{A}^I_{q,q'(,w)}$ contains only transitions from the components of I and q' is reachable by w from q. The letter $\mathcal{A}^I_{q,q'}$ with trivial set $I = \{i\}$ is used for a block of asynchronous transitions from the component index i and the letter $\mathcal{A}^I_{q,q',w}$ with non-trivial I is used for blocks of synchronized transitions of the component indices from I.

Definition 3 (Global specification automaton). *Given a complement specification automaton $\mathcal{A}_{\overline{Spec}} = (Q, \Sigma, \Delta_{\overline{Spec}}, q_0, F)$, let the global specification automaton of $\mathcal{A}_{\overline{Spec}}$ be $\mathcal{A}_{Glob} := (G, \Sigma_B, \Delta_{Glob}, q_0, F)$ where:*

1. *the state set G contains all states of Q such that in $\mathcal{A}_{\overline{Spec}}$ there are out-going and in-coming transitions that belong to different synchronization profiles: $G := \{q_0\} \cup F \cup \{q \in Q \mid \exists q_1, q_2 \in Q \exists (q_1, c, q), (q, d, q_2) \in \Delta_{\overline{Spec}}$ with $sp(c) \neq sp(d)\}$,*
2. *the set Σ_B is the local block alphabet of*
   (a) *the letters $\mathcal{A}^I_{qq'}$ with $q, q' \in G$ and $I = sp(c)$ is a singleton for a letter $c \in \Sigma$ of asynchronous transitions and*
   (b) *the letters $\mathcal{A}^I_{q,q',w}$ with $q, q' \in G$ and $I = sp(c)$ for a c-labeled synchronized transition and there exists a w-labelled path in $\mathcal{A}_{\overline{Spec}}$ from q to q' using only letters from the components in I,*
3. *the transition relation Δ_{Glob} is defined as the set $\{(q, t, q') \mid q, q' \in G$ such that there exists a path labelled by a word w from q to q' in $\mathcal{A}_{\overline{Spec}}$ containing only labels of the components of I. The letter t is $\mathcal{A}^I_{qq'}$ if I is a singleton and $\mathcal{A}^I_{q,q',w}$ otherwise.$\}$.*

For a given $z = t_1 \ldots t_u \in L(\mathcal{A}_{Glob})$ let the *projection of z to component i*, denoted by $z\restriction_i$, be the restriction of z to all $t_j = \mathcal{A}^{I_j}_{q_j,q'_j(,w)}$ with $i \in I_j$.

Example 2. Figure 2 shows a complement specification and its transformation into a global automaton for the product from Fig. 1. Each letter from the local block alphabet can be interpreted as an automaton $\mathcal{A}^I_{q,q'(,w)}$, e. g. the letter $\mathcal{A}^{\{1\}}_{q_0q_3}$ corresponds to the automaton $\mathcal{A}_{\overline{Spec}}$ with initial state q_0, final state set $\{q_3\}$, and transitions $(q_0, a, q_1), (q_1, a, q_1)$ and (q_1, a, q_3).

2.3 Transition Profiles

For an automaton $\mathcal{A}$ we now define transition profiles according to [KN01]. Informally, a transition profile describes a class of words such that the automaton acts on these words in the same way.

Definition 4 (Transition profile).
Given an automaton $\mathcal{A}$ with state set Q and states $p, q \in Q$. Let $p \xrightarrow{u} q$ iff there exists a u-labeled path in $\mathcal{A}$ which starts in p and ends in q. Further, let $p \overset{u}{\Rightarrow} q$ iff $p \xrightarrow{u} q$ and the u-labeled path visits at least one final state. The transition profile $t_{\mathcal{A}}[u]$ of a word $u \in \Sigma^*$ is defined as two lists of tuples: the list of all states (p, q) with $p \xrightarrow{u} q$ in $\mathcal{A}$ and the list of all states (p, q) with $p \overset{u}{\Rightarrow} q$ in $\mathcal{A}$. Let $TP_{\mathcal{A}}$ denote the set of all transition profiles of $\mathcal{A}$.

Let $|Q| = k$. Obviously, it holds that $|TP_{\mathcal{A}}| \leq 2^{k^2}$. Let $u \sim_{\mathcal{A}} v$ iff $\forall p, q \in Q : (p \xrightarrow{u} q \Leftrightarrow p \xrightarrow{v} q$ and $p \overset{u}{\Rightarrow} q \Leftrightarrow p \overset{v}{\Rightarrow} q)$. The relation $\sim_{\mathcal{A}}$ is an equivalence relation. For $u \in \Sigma^*$ let $[u]$ denote the *$\sim_{\mathcal{A}}$-equivalence class of u*. The transition profile $t_{\mathcal{A}}[u]$ characterizes the class $[u]$. Because $\mathcal{A}$ has finitely many states, there are only finitely many transition profiles and the index of $\sim_{\mathcal{A}}$ is finite.

Definition 5. *Let $t_1, t_2 \in TP_{\mathcal{A}}$ be transition profiles of $\mathcal{A}$ then $t_1 \circ t_2$ – called the concatenation of t_1 and t_2 – is defined as follows: $p \to q \in t_1 \circ t_2$ iff $\exists r \in Q$: $(p \to r \in t_1$ and $r \to q \in t_2)$ and $p \Rightarrow q \in t_1 \circ t_2$ iff $\exists r \in Q$: $(p \Rightarrow r \in t_1$ and $r \to q \in t_2)$ or $(p \to r \in t_1$ and $r \Rightarrow q \in t_2)$. Let $T_1, T_2 \subseteq TP_{\mathcal{A}}$ be subsets of transition profiles. The concatenation $T_1 \circ T_2$ is defined by $T_1 \circ T_2 = \{t_1 \circ t_2 \mid t_1 \in T_1, t_2 \in T_2\}$.*

2.4 Effect of an Automaton

For the proof of the complexity of the main theorem, we use a precalculation of the "effect" of an automaton $\mathcal{A}$ on a transition system (seen as an automaton $\mathcal{B}$). This effect describes which state changes are possible in $\mathcal{B}$ by the words in the language of $\mathcal{A}$.

We first introduce a concatenation of automata. The concatenation automaton $\mathcal{A}_1 \cdot \mathcal{A}_2$ is defined by the union of $\mathcal{A}_1$ and $\mathcal{A}_2$ with additional transitions (p, ε, q_0) for all final states p of $\mathcal{A}_1$ and where q_0 is the initial state of $\mathcal{A}_2$. The concatenation ε-NFA[1] recognizes the concatenation of the languages of the involved automata. Formally, $\mathcal{A}_1 \cdot \mathcal{A}_2$ is defined as follows.

Definition 6. *Let $\mathcal{A}_1 := (Q_1, \Sigma_1, \Delta_1, q_{01}, F_1)$ and $\mathcal{A}_2 := (Q_2, \Sigma_2, \Delta_2, q_{02}, F_2)$. We assume that Q_1 and Q_2 are disjoint, otherwise we use a renamed copy of the state set. The concatenation (ε)-NFA $\mathcal{A}_1 \cdot \mathcal{A}_2$ is defined as $(Q_1 \dot\cup Q_2, \Sigma_1 \cup \Sigma_2, \Delta, q_{01}, F_2)$ with $\Delta := \Delta_1 \dot\cup \Delta_2 \dot\cup \{(p, \varepsilon, q_{02}) \mid p \in F_1\}$.*

Based on the transition profiles of the last section we now introduce the notion of an effect of an automaton.

Definition 7 (Effect of an automaton). *Given two automata $\mathcal{A}, \mathcal{B}$ the effect of $\mathcal{A}$ on $\mathcal{B}$ is defined by the set of transition profiles of $\mathcal{B}$ that are chosen by the words in the language of $\mathcal{A}$, formally $\Theta(\mathcal{A}, \mathcal{B}) := \{t_{\mathcal{B}}[u] \mid u \in L(\mathcal{A})\}$.*

[1] An ε-NFA is a non-deterministic finite automaton (NFA) with ϵ-transitions and can be converted into an NFA which accepts the same language.

The effect of $\mathcal{A}$ on $\mathcal{B}$ is always a finite set of transition profiles, because there are only finitely many transition profiles of $\mathcal{A}$ at all. We give a sketch of an algorithm to compute the effect of $\mathcal{A} := (Q_{\mathcal{A}}, \Sigma, \Delta_{\mathcal{A}}, p_0, F_{\mathcal{A}})$ on $\mathcal{B} := (Q_{\mathcal{B}}, \Sigma, \Delta_{\mathcal{B}}, q_0, F_{\mathcal{B}})$. Let $\mathcal{C} := \mathcal{A} \times \mathcal{B}$ be the product automaton with final state set $F_{\mathcal{C}} = F_{\mathcal{A}} \times Q_{\mathcal{B}}$. Then $(p_1 \to p'_1, p_2 \to p'_2, \dots, p_l \to p'_l,\ p_{i_1} \Rightarrow p'_{i_1}, \dots, p_{i_m} \Rightarrow p'_{i_m})$ is in the effect of $\mathcal{A}$ on $\mathcal{B}$ for $l \leq |Q_{\mathcal{B}}|$, $p_i, p'_i \in Q_{\mathcal{B}}, p_i \neq p_j$ for $i \neq j$ and $i_k \in [l]$ for $k \in [m]$ iff there exists a word $w \in L(\mathcal{C})$ such that: for all $i \leq l$: $(q_0, p_i) \xrightarrow{w} (q, p'_i) \in \Delta^*_{\mathcal{C}}$ for any $q \in Q_{\mathcal{A}}$ and $\forall k \leq m : p_{i_k} \Rightarrow p'_{i_k}$ iff the path of w in $\mathcal{C}$ visits a state (q_f, p) with $q_f \in F_{\mathcal{A}}$ and there exists no larger $l' > l$ and $k' > k$ with the same conditions.

It suffices to check a finite number of words w (until we get a state repetition in $\mathcal{C}$) to determine the effect of $\mathcal{A}$ on $\mathcal{B}$. Given automata $\mathcal{A}_1, \mathcal{A}_2, \mathcal{B}$ the effect of the concatenation automaton $\mathcal{A}_1 \cdot \mathcal{A}_2$ on $\mathcal{B}$ is determined by concatenating the transition profiles of the effect of $\mathcal{A}_1$ on $\mathcal{B}$ and $\mathcal{A}_2$ on $\mathcal{B}$.

3 Composition: Simple Case

In this section we present the result that reduces the question whether a given synchronized product and given complement specification (with bounded synchronization) have common labeling sequences to checking whether the components of this product and certain parts of the complement specification have common labeling sequences.

Theorem 1. *For a given (bounded synchronized) complement specification automaton $\mathcal{A}_{\overline{Spec}}$ without predicates and any synchronized product $\mathcal{A}_{Sys}$ of components K_i for $i \in I$, compatible with $\mathcal{A}_{\overline{Spec}}$, we have:*

$$L(\mathcal{A}_{Sys}) \cap L(\mathcal{A}_{\overline{Spec}}) \neq \emptyset \Leftrightarrow \exists z \in L(\mathcal{A}_{Glob}) \text{ such that } \forall i \in I : \ L(z\restriction_i) \cap L(K_i) \neq \emptyset.$$

Let us mention that the length of the word z can be restricted. A complexity analysis is deferred to the treatment of the general case in Sect. 5.

Example 3. The complement specification $\mathcal{A}_{\overline{Spec}}$ from Fig. 2 expresses that a synchronized transition should never be taken after any component has taken more than two asynchronous transitions. Obviously, in the product from Fig. 1 there is a path which conflicts with this property, namely $(12, a, 13, a, 12, c, 24)$.

In $\mathcal{A}_{Glob}$ there exists a path with label $z = \mathcal{A}^{\{1\}}_{q_0,q_3}\mathcal{A}^{\{1,2\}}_{q_3,q_4,c}$ and for $z\restriction_1 = \mathcal{A}^{\{1\}}_{q_0,q_3}\mathcal{A}^{\{1,2\}}_{q_3,q_4,c}$ there exists the label sequence aac in K_1 and for $z\restriction_2 = \mathcal{A}^{1,2}_{q_3,q_4,c}$ there exists the label sequence c in K_2. These sequences lead together to the failure via aac to state 24 in the product.

We can generalize Theorem 1 to complement specifications given as Büchi automata. Thus, we can capture all linear time properties which can be converted into Büchi automata with a bounded synchronization. For the conversion standard techniques can be used.

Corollary 1. *For a given bounded synchronized complement specification Büchi automaton $\mathcal{B}_{\overline{Spec}}$ and any synchronized product $\mathcal{A}_{Sys}$ compatible with $\mathcal{A}_{\overline{Spec}}$: $L(\mathcal{A}_{Sys}) \cap L(\mathcal{B}_{\overline{Spec}}) \neq \emptyset$ holds iff there exists a word $z \in L(\mathcal{A}_{Glob})$ such that $\forall i \in [n]$: $L(z\restriction_i) \cap L(K_i) \neq \emptyset$, where z is*

- *either a finite word and at least one word $z\restriction_i$ ends with Büchi automaton as local block*
- *or an ω-word and all local blocks of $z\restriction_i$ are finite automata.*

4 Extension to Specifications with Predicates

In this section we discuss specifications with unary predicates. For this, we encode in Sect. 4.1 the predicates (respectively their negation) in the components (as well as in the product) as self-loop transitions. In Sect. 4.2 we fix the format we use for a complement specification automaton that is compatible with the actions and the predicates of a product and consider a small modification to improve the results later. For the complement specification we introduce in Sect. 4.3 a sequential projection which allows us to verify which predicates hold at a state in the product by checking that all (possibly negated) "predicate" transitions exist before taking a "normal" transition. Further, we analyse the transition structure of the complement specification to reduce (in the sequential projection) the number of checks of "predicate" transitions, if successive "normal" transitions belong to the same synchronization profiles. We continue in Sect. 4.4 with a formal definition of a sequential projection of a word: This projection translates a word with letters like $(a, 1, 1, 0)^T$ to words where the predicates are checked via the self loop transitions, e. g. $(a, 1, 1, 0)^T$ is translated to $uv\neg wa$. In Sect. 4.5 we conclude by splitting the sequential projection into the local blocks of a global automaton as in Sect. 3.

4.1 Modification of the Product

To store the predicates as self loop transitions, we modify the components by adding transition relations $R^i_v/R^i_{\neg v}$ with $(x, x) \in R^i_v/R^i_{\neg v}$ iff $x \in P_v/x \notin P_v$, and we modify the synchronized product by adding transition relations $\bar{R}_{(\neg)v^i}$ with $(\bar{x}, \bar{y}) \in \bar{R}_{(\neg)v^i}$ iff $\bar{x} = \bar{y}$ and $(\bar{x}[i], \bar{x}[i]) \in R^i_{(\neg)v}$.

Example 4. In Fig. 3 three component transition systems K_1, K_2, K_3 and their synchronized product $\bar{K}$ are shown. Again, the state names are chosen differently for the components: $S_1 := \{1, 2\}$, $S_2 := \{3, 4\}$ and $S_3 := \{5, 6\}$. We use different letters for the predicates and only show the corresponding self-loop transitions $(\neg)u, (\neg)v, (\neg)w$ in K_1, K_2 respectively K_3 (and not the predicates itselves). The labels a/b are used for asynchronous transitions of K_1/K_2, i. e. $sp(a) = \{1\}$ and $sp(b) = \{2\}$ and the labels c, d are used for synchronized transitions with synchronization profile $sp(c) = sp(d) = \{2, 3\}$.

To compare the paths of a synchronized product with the complement specification, we translate $\mathcal{A}_{Sys}$ in an expanded form $\mathcal{A}_{ESys}$, where the values of the predicates are added to the transition labels, e. g. we have $s \xrightarrow{(a,1,1,0)} s'$ in $\mathcal{A}_{ESys}$ iff $\mathcal{A}_{Sys}$ contains the transition $s \xrightarrow{a} s'$ and P_u, P_v hold at state s, whereas P_w does not. Formally, $\mathcal{A}_{ESys}$ of $\mathcal{A}_{Sys}$ is defined as $(\bar{S}, \bar{R})$ with $\bar{S}$ as in Definition 1 and for $c \in \Sigma$: $(s, (c, b^1_1, \ldots, b^n_l)^T, s') \in \bar{R}$ holds iff $(s, s') \in \bar{R}_c$ and ($b^i_j = 1$ iff $\bar{P}_{v^i_j}$ holds at state s).

Now, we introduce a complement specification automaton with predicates.

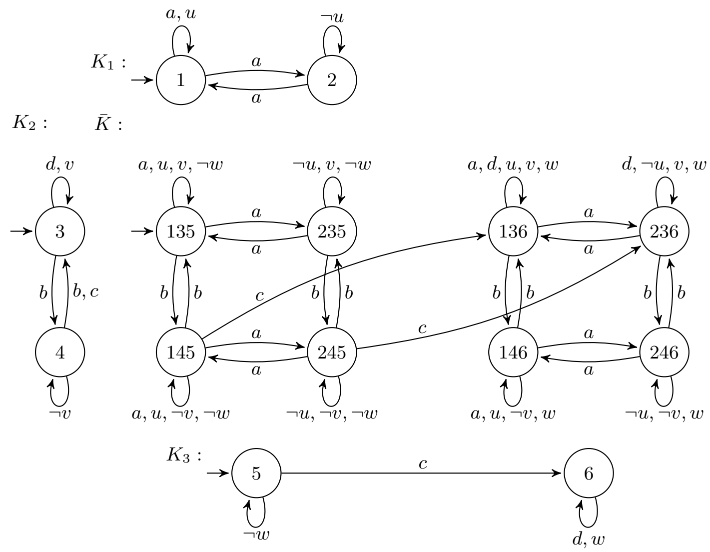

Fig. 3. Components K_1, K_2, K_3 and their synchronized product $\bar{K}$

4.2 Complement Specification with Predicates

Definition 8. *A (bounded synchronized) complement specification automaton with predicates for a synchronized product $\bar{K}$ is an automaton $\mathcal{A}_{\overline{Spec}}$, compatible with the action and predicate alphabet of $\bar{K}$. Formally, $\mathcal{A}_{\overline{Spec}} := (Q, \Sigma \times \mathbb{B}^{l \cdot n}, \Delta, q_0, F)$ with $l := |V|$. A transition has the form $(q, (c, B^1, \ldots, B^n)^T, q')$ with $c \in \Sigma$ and $B^i := (b_1^i, \ldots, b_l^i)$ specifies the truth values of the predicates $\bar{P}_{v^i}$ for $v \in V = \{v_1, \ldots, v_l\}$ at the state q. As in Definition 2 the boundedness condition has to be fulfilled: for every non-trivial synchronization profile $sp(c)$ of c-labeled transitions the restriction of $\mathcal{A}_{\overline{Spec}}$ to all transitions of this synchronization profile $sp(c)$ contains no loops.*

As preparation for the sequential projection of the complement specification and to reduce the number of checks of "predicate" transitions in it, we introduce the notion of *switching* transitions: for subsequent transitions we distinguish between transitions that use labels of the same synchronization profile as the transition before and those which switch to another component.

We call a transition with label $B = (c, b_1^1, b_2^1, \ldots, b_l^n)$ *switching with respect to a transition with label* $B' = (c', b_1'^1, \ldots, b_l'^n)$ if c has a synchronization profile different from c' ($sp(c) \neq sp(c')$) or if there exists at least one predicate valuation of the other components which does not coincide ($\exists j \in [l]$ with $b_j^k \neq b_j'^k$ for $k \notin sp(c)$). A transition t is called *switching* if there exists a predecessor t' such that t is switching with respect to t'. A transition is called *non-switching* if it is not switching with respect to all predecessors.

For a given complement specification automaton we use a small modification to improve the results later: we double each state s which has self loop transitions if all of these transitions are non-switching w.r.t. each other: Let t denote a transition $(s, (c, \bar{B}), s)$ which is non-switching w.r.t. all other transitions of the form $t' = (s, (c', \bar{B}'), s)$ of $\mathcal{A}_{\overline{Spec}}$. For all such transitions t we remove t itself, introduce a new state s' and add transitions $(s, (c, \bar{B}), s')$ and $(s', (c, \bar{B}), s')$. For

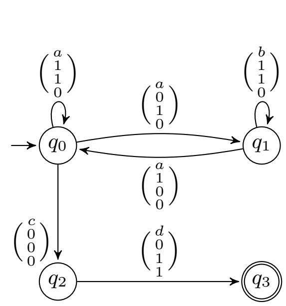

Fig. 4. Complement specification

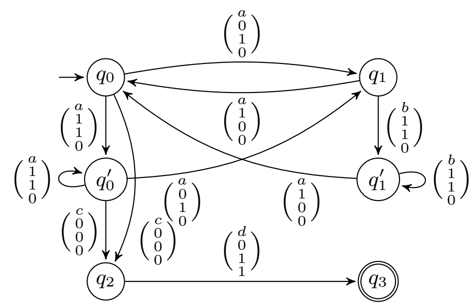

Fig. 5. Modified complement specification

all outgoing transitions $(s, (c_1, \bar{B}_1), s_1)$ with $s_1 \neq s$ and any c_1 we further add a transition $(s', (c_1, \bar{B}_1), s_1)$. Note that this modification does not change the language of the automaton, but ensures that after taking the transition from s to s' – meaning that we would take the self-loop transition in the original automaton once – we are now in a state which is only reachable by transitions in which the values of the predicates of the other components do not change until we leave this state. Thus, in a run which repeats the state s' all transitions from s' to s' can be checked in the same component/synchronization profile. Fig. 4 shows a complement specification automaton and Fig. 5 this modification.

4.3 Sequential Projection of a Complement Specification

The *sequential projection $\mathcal{A}_{Proj}$ of a complement specification* $\mathcal{A}_{\overline{Spec}} = (Q, \Sigma \times \mathbb{B}^{l \cdot n}, \Delta_{\overline{Spec}}, q_0, F)$ transfers the truth value of the predicates into "predicate transitions" which are checked before the "normal transitions". It is defined as the automaton $\mathcal{A}_{Proj} := (Q \cup R, \Sigma \cup (V \times [n]), \Delta_{Proj}, q_0, F)$ with $R := (\Sigma \times \mathbb{B}^{l \cdot n}) \times [l \cdot n] \times Q$. We explain the definition of the transition relation Δ_{Proj}: for a transition $t = (q, (c, B^1, \dots, B^n), q') \in \Delta_{\overline{Spec}}$ we check the predicates – corresponding to $B^1, \dots, B^n$ – one after the other and afterwards the label c of the transition t. Note that the order in which the predicates have to be checked can be chosen freely. For each synchronized transition with $c \in \Sigma$ we first verify that for all components different from $sp(c)$ there exist transitions for the predicates corresponding to the sets B^j of t before verifying this for the components of $sp(c)$ and before taking the c-labeled transition. If the transition is non-switching with respect to all predecessor transitions, we only check the predicates of the components of $sp(c)$ before taking the c-labeled transition.

To define Δ_{Proj} formally, we introduce for a transition to a state q and with label $B := (a, B^1, \dots, B^n)$, $B^i = (b^i_1, \dots b^i_l)$ and a fixed number $k \in \{0, l, 2 \cdot l, \dots, (n-1) \cdot l\}$ the abbreviation $(B, k, q) \overset{B^i}{\dashrightarrow} (B, k+l, q)$ to denote the sequence of transitions $(B, k, q) \xrightarrow{x^i_1} \cdots \xrightarrow{x^i_l} (B, k+l, q)$ where $x^i_j = v^i_j$ if $b^i_j = 1$ and $x^i_j = \neg v^i_j$ if $b^i_j = 0$. The transition relation Δ_{Proj} is then defined as follows: Let $t := (q, B, q') \in \Delta_{\overline{Spec}}$ be a synchronized transition with label $c \in \Sigma$ and synchronization profile $sp(c) = \{i_1, \dots, i_m\}$. If there exists a predecessor transition t' such that t is switching w.r.t. t', we add the transitions $q := (B, 0, q') \overset{B^1}{\dashrightarrow} \cdots \overset{B^{i_1 - 1}}{\dashrightarrow} (B, (i_1 - 1) \cdot l, q') \overset{B^{i_1+1}}{\dashrightarrow} \cdots \overset{B^n}{\dashrightarrow} (B, (n-m) \cdot l, q') \overset{B^{i_1}}{\dashrightarrow} \cdots \overset{B^{i_m}}{\dashrightarrow} (B, l \cdot n, q') \xrightarrow{a} q'$ to

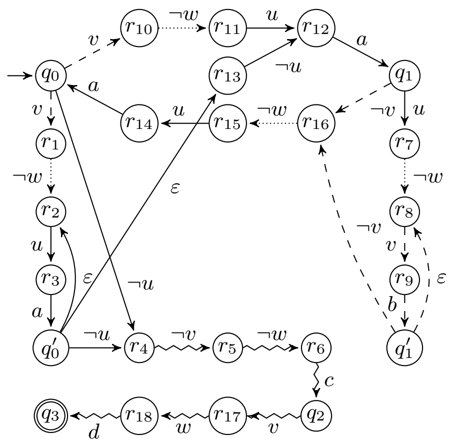

Fig. 6. Complement specification projection

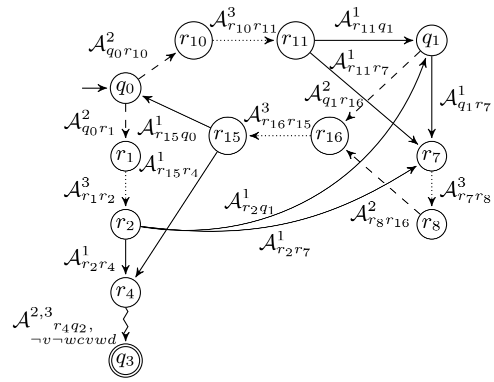

Fig. 7. Global specification

Δ_{Proj}. If t is non-switching for all predecessors $(q^*, (c', A^1, \ldots, A^n), q) \in \Delta_{\overline{Spec}}$, only the predicates of the components from $sp(c)$ have to be checked for t so only the transitions $q \xrightarrow{\varepsilon} (B, (n-m) \cdot l, q') \overset{B^{i_1}}{\dashrightarrow} \cdots \overset{B^{i_m}}{\dashrightarrow} (B, n \cdot l, q') \xrightarrow{x} q'$ are added to Δ_{Proj}.

Example 5. In Fig. 6 we see the sequential projection of the modified complement specification from Fig. 5. For readability the states that were added to the complement specification automaton are abbreviated with r_i $(1 \leq i \leq 16)$, where e. g. $r_{10} := ((a, 1, 1, 0)^T, 1, q_1)$ and $r_6 := ((c, 0, 0, 0)^T, 3, q_2)$. The transitions for the predicates P_u, P_v, P_w are labeled with u, v, w, respectively their negation. To indicate the assigned component, the transitions with labels of K_1, K_2, respectively K_3, are drawn as normal, dashed respectively dotted lines. The transitions of the synchronization profile $sp(c) = sp(d) = \{2, 3\}$ are drawn as zigzag lines.

4.4 Sequential Projection of a Word

To be able to compare a run in the complement specification automaton with a path in the synchronized product using the self-loop transitions for the predicates, we define a *sequential projection of a word w*. In this sequential projection the truth values $b^i_j = 1/b^i_j = 0$ for a letter $B = (c, b^1_1, \ldots, b^n_l)$ are tested by checking the existence of the self-loop transitions with the labels $v^i_j/\neg v^i_j$.

Let $B^k = (c_k, \ldots)$ and $B^{k+1} = (c_{k+1}, \ldots)$ be two subsequent letters in w where c_k, c_{k+1} have the same synchronization profile $sp(c_k) = sp(c_{k+1})$. Because of the structure of the product, the predicates of the other components $I \backslash sp(c_k)$ cannot change between subsequent transitions in the product with labels B^k, B^{k+1} and thus do not have to be checked again if they coincide in B^k and B^{k+1}. We define $seq(w)$ as a regular expression which allows that the coinciding predicates are not checked again but also that they are checked again. Further, we allow any possible order in which these predicates are tested. We now define $seq(w)$ formally.

Let $\Sigma' := \Sigma \times \mathbb{B}^{l \cdot n}$ and $w = B_1 \cdots B_{len(w)} \in \Sigma'^*$ be a word with length $len(w)$. Further let $\mathfrak{S}_t$ be the symmetric group with t elements. The *sequential* *projection* $seq(w)$ *of the word* w is defined by $seq(w) := seq(B_1) \cdots seq(B_{len(w)})$ with $seq(B_k)$ ($k \in [len(w)]$) for a letter $B_k = (c_k, b_1^1, \ldots, b_l^n)$ defined as follows:

If $k = 1$ or B_k is switching w.r.t. B_{k-1} then the regular expression $seq(B_k)$ is $\bigvee_{\sigma \in \mathfrak{S}_{n \cdot l}} z_{\sigma(1)} \cdots z_{\sigma(n \cdot l)} c_k$ where $z_{(m-1)\cdot l+j} = v_j^m$ if $b_j^m = 1$ and $z_{(m-1)\cdot l+j} = \neg v_j^m$ if $b_j^m = 0$ for $m \in [n]$. (This allows any sequence in which the predicates $(\neg)v_j^m$ are checked.)

If B_k is non-switching w.r.t. B_{k-1} ($k \neq 1$) and $sp(c_{k-1}) = sp(c_k) = \{i_1, \ldots, i_f\}$ then in $seq(B_k)$ either the predicates of the components from $sp(c_k)$ are checked in any sequence or all predicates are checked in any sequence. For $\sigma \in \mathfrak{S}_{f \cdot l}$ let $\iota : [f \cdot l] \to I$ be a function that assigns the first l numbers to i_1, the next l numbers to i_2 and so on. The function is defined as $\iota(t) = i_j$ if $(j-1) \cdot l < \sigma(t) \leq j \cdot l$. Further let $\gamma : [f \cdot l] \to [n \cdot l]$ be defined as $\gamma(t) = (\iota(t) - 1) \cdot l + (\sigma(t) \bmod l)$ then $seq(B_k) := \bigvee_{\sigma \in \mathfrak{S}_{f \cdot l}} z_{\gamma(1)} \cdots \ z_{\gamma(f \cdot l)} c_k \vee \bigvee_{\sigma \in \mathfrak{S}_{n \cdot l}} z_{\sigma(1)} \cdots z_{\sigma(n \cdot l)} c_k$.

Example 6. Let $w = (a, 1, 1, 0)^T (a, 0, 1, 0)^T$. We have $uv\neg wa\neg ua \in L(seq(w))$ (where $v, \neg w$ are not checked again), but also e. g. $v\neg wua\neg uv\neg wa \in L(seq(w))$ (where all transitions are checked again and we use an other sequence in which the predicate transitions are checked).

The following lemma shows the connection of the paths in the synchronized product and the paths of the expanded synchronized product:

Lemma 1. *Let* $\mathcal{A}_{Sys}$ *be a synchronized product and* $\mathcal{A}_{ESys}$ *the expanded synchronized product of* $\mathcal{A}_{Sys}$ *and* $y \in \Sigma' := \Sigma \times \mathbb{B}^{l \cdot n}$ *then* $y \in L(\mathcal{A}_{ESys})$ *holds iff* $L(seq(y)) \cap L(\mathcal{A}_{Sys}) \neq \emptyset$.

Proof. If $y \in L(\mathcal{A}_{ESys})$ holds then there exists a path $s_0 \xrightarrow{B_1} \ldots \xrightarrow{B_m} s_m$ with $y = B_1 \ldots B_m$ and at s_k the predicate $P_{v_j}^i$ holds iff $b_j^i = 1$ of $B_k = (c, b_1^1, \ldots, b_l^n)$. We conclude that in $\mathcal{A}_{Sys}$ there exist self-loop transitions at s_k labeled by $(\neg)v_1^1, \ldots, (\neg)v_l^n$ according to $b_j^i = 1(/0)$ and thus $L(seq(y)) \subseteq L(\mathcal{A}_{Sys})$.
If $L(seq(y)) \cap L(\mathcal{A}_{Sys}) \neq \emptyset$ with $y = B_1 \ldots B_m$ and $B_k = (c, b_1^1, \ldots, b_l^n)$ then for B_k there exists a path from s_{k-1} by a word w_k to s_k that checks all predicates at s_{k-1} and afterwards uses a c-transition to s_k (or it checks only the predicates of component K_i if a transition of the same component was taken before and the other predicates did not change). We conclude that in $\mathcal{A}_{ESys}$ there exists a transition with label B_k from s_{k-1} to s_k. By the concatenation of the path $y \in \mathcal{A}_{ESys}$ follows.

4.5 Global Specification

The *global specification automaton* $\mathcal{A}_{Glob} := (G, \Sigma_B, \Delta_{Glob}, q_0, F)$ for a sequential projection $\mathcal{A}_{Proj} = (Q \cup R, \Sigma, \Delta_{Proj}, q_0, F)$ of a (bounded synchronized) complement specification automaton is defined as in Definition 3, but the state set G is the union of the sets

1. $\{q_0\} \cup F$
2. $\{q \in Q \mid \exists r_1, r_2 \in R\ \exists (r_1, c, q), (q, v^i, r_2) \in \Delta_{Proj} : c \in \Sigma \wedge i \notin sp(c)\}$
3. $\{r = (c, B^1, \ldots, B^n, k, q) \in R \mid \exists r_1, r_2 \in Q \cup R\ \exists (r_1, v^i, r), (r, w^j, r_2) \in \Delta_{Proj} : i \neq j \wedge (i \notin sp(c) \vee j \notin sp(c))\}$

For a local block $\mathcal{A}^I_{r,r',w}$ and $i \in I$ let $\mathcal{A}^I_{r,r',w}\restriction_i := \mathcal{B}^i_{r,r'}$ where $\mathcal{B}^i_{r,r'}$ is the automaton which results from $\mathcal{A}^I_{r,r',w}$ if we replace all transitions from component indices different from i by ε-transitions. For a local block $\mathcal{A}^I_{r,r'}$ with singleton $I = \{i\}$ we define $\mathcal{A}^I_{r,r'}\restriction_i$ to be $\mathcal{A}_{r,r'}$ itself. For a given $z = t_1 \dots t_u \in L(\mathcal{A}_I)$ we define the *projection of z to component i*, denoted by $z\restriction_i$, as $t^i_1 \dots t^i_{len(i)}$ with $len(i)$ maximal such that for $j \in [len(i)] : t^i_j = \mathcal{A}^{I_j}_{r_j,r'_j(,w)}\restriction_i$ (for states $r_j, r'_j \in G$) and $i \in I_j$ in the order in which the $\mathcal{A}^{I_j}_{r_j,r'_j(,w)}$ appear in z.

Example 7. In Fig. 7 we show the global specification automaton of the sequential projection from Fig. 6. The local blocks are defined as in Sect. 3, e. g. $\mathcal{A}^{\{1\}}_{r_{11},r_7} = (\{r_{11}, r_{12}, q_1\}, \{u, a\}, \Delta, r_{11}, \{q_1\})$ where Δ contains only the transitions (r_{11}, u, r_{12}) and (r_{12}, a, q_1).

5 Composition: General Case

With the preliminaries from the last section we generalize Theorem 1 to specifications which can also check the predicates of a product. Further, we give an upper bound for the induced algorithm.

Theorem 2. *For a given (bounded synchronized) complement specification automaton $\mathcal{A}_{\overline{Spec}}$ and any synchronized product $\mathcal{A}_{ESys}$ of components K_i for $i \in [n]$, compatible with $\mathcal{A}_{\overline{Spec}}$ we have:*

$$L(\mathcal{A}_{ESys}) \cap L(\mathcal{A}_{\overline{Spec}}) \neq \emptyset \Leftrightarrow \exists z \in L(\mathcal{A}_{Glob}) \text{ such that } \forall i \in [n] :\ L(z\restriction_i) \cap L(K_i) \neq \emptyset.$$

The size of $\mathcal{A}_{Glob}$ is quadratic in the size of $\mathcal{A}_{\overline{Spec}}$ and linear in the number of predicates and components. The length of z is exponential in the maximal number of states a component has. The tests whether $L(z\restriction_i) \cap L(K_i) \neq \emptyset$ need a precalculation which is exponential in the number of components, predicates and states of the complement specification, and in the number of states the largest component has.

Example 8. In $\mathcal{A}_{ESys}$ (which is $\mathcal{A}_{Sys}$ from Fig. 3 with the predicate valuations of the current state on the outgoing transitions) and in $\mathcal{A}_{\overline{Spec}}$ from Fig. 5 there exist the paths $\pi_{\overline{Spec}} = (q_0, q'_0, q_1, q'_1, q_0, q_2, q_3)$ and $\pi_{ESys} = (s_{135}, s_{235}, s_{135}, s_{145}, s_{245}, s_{236}, s_{236})$ labeled with $(a, 1, 1, 0)^T (a, 0, 1, 0)^T (b, 1, 1, 0)^T (a, 1, 0, 0)^T (c, 0, 0, 0)^T (d, 0, 1, 1)^T$. In $\mathcal{A}_{Sys}$ there exists a path π_{Sys} with the same state sequence like π_{ESys}, but with each state repeated four times and the label sequence $v\neg wua\varepsilon\neg uau\ \neg wvb\neg v\neg wua\neg u\neg v\neg wcvwd$. From the path $\pi_{\overline{Spec}}$ we get a path $\pi_{Proj} = (q_0, r_1, r_2, r_3, q'_0, r_{13}, r_{12}, q_1, r_7, r_8, r_9, q'_1, r_{16}, r_{15}, r_{14}, q_0, r_4, r_5, r_6, q_2, r_{17}, r_{18}, q_3)$ in $\mathcal{A}_{Proj}$ of Fig. 6 with the same label sequence.

From π_{Proj} we get $\pi_{Glob} = (q_0, r_1, r_2, r_7, r_8, r_{16}, r_{15}, r_4, q_3)$ in $\mathcal{A}_{Glob}$ from Fig. 7 for $z = \mathcal{A}^{\{2\}}_{q_0,r_1} \mathcal{A}^{\{3\}}_{r_1,r_2} \mathcal{A}^{\{1\}}_{r_2,r_7} \mathcal{A}^{\{3\}}_{r_7,r_8} \mathcal{A}^{\{2\}}_{r_8,r_{16}} \mathcal{A}^{\{3\}}_{r_{16},r_{15}}\ \mathcal{A}^{\{1\}}_{r_{15},r_4} \mathcal{A}^{\{2,3\}}_{r_4,q_3,\neg v\neg wcvwd}$. Thus, for $i \in \{1, 2, 3\}$ there exist a word in $L(z\restriction_i) \cap L(K_i)$, e. g. for $i = 2$: $z\restriction_2 = \mathcal{A}^{\{2\}}_{q_0,r_1} \cdot \mathcal{A}^{\{2\}}_{r_8,r_{16}} \cdot (\mathcal{A}^{\{2,3\}}_{r_4,q_3,\neg v\neg wcvwd}\restriction_2)$ and $v \cdot vb\neg v \cdot \neg v\varepsilon cv\varepsilon d \in L(z\restriction_2) \cap L(K_2)$ with the path $\pi_2 = (3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3)$ in K_2.

We show correctness, completeness and complexity bounds.

Proof (Correctness).
Given $L(\mathcal{A}_{ESys}) \cap L(\mathcal{A}_{\overline{Spec}}) \neq \emptyset$ then there exists a word y with $y \in L(\mathcal{A}_{\overline{Spec}})$ and $L(seq(y)) \subseteq L(\mathcal{A}_{Sys})$ by the proof of Lemma 1. Because of the construction of $\mathcal{A}_{Proj}$ from $\mathcal{A}_{\overline{Spec}}$, it follows that $L(seq(y)) \cap L(\mathcal{A}_{Proj}) \neq \emptyset$. Thus, there exists a word $v \in L(\mathcal{A}_{Proj}) \cap L(\mathcal{A}_{Sys})$. Let π_{Proj} be a path induced by v in $\mathcal{A}_{Proj}$. We group the subsequent path segments of π_{Proj} which use transitions of the same component respectively with the same synchronization profile. The path π^{Proj} is decomposable into path segments $\pi_1^{Proj}, \pi_2^{Proj}, \ldots, \pi_u^{Proj}$ for $u \leq len(\pi^{Proj})$ such that π_j^{Proj} uses only transitions with labels of exactly one synchronization profile $sp(c)$ for $c \in \Sigma$. Let I_j denote these components $sp(c)$ of the j-th path segment. We assume that $I_j \neq I_{j+1}$ for $j \in [u-1]$, otherwise the corresponding path segments could be joined to guarantee this property.
Because of the definition of $\mathcal{A}_{Glob}$ there exists a path π_{Glob} in $\mathcal{A}_{Glob}$ with label sequence $z = \mathcal{A}^{I_1}_{r_1,r'_1(,w)} \cdots \mathcal{A}^{I_u}_{r_u,r'_u(,w)}$ where r_j, r'_j are the first respectively the last state of the path segment π_j^{Proj} and in $\mathcal{A}^{I_j}_{r_j,r'_j}$ the path π_j^{Proj} leads from r_j to r'_j. Let v_j be the label of the path segment π_j^{Proj}.
Let π_{Sys} be a path induced by v in $\mathcal{A}_{Sys}$. Then, there exists a decomposition of π_{Sys} into path segments $\pi_1^{Sys}, \ldots, \pi_u^{Sys}$ with the labels $v_1, \ldots, v_u$. Because v_j uses only labels of I_j the state "names" in π_j^{Sys} differ only in the components I_j. For all $i \in I_j$ let $v_j{\upharpoonright}_i$ be the restriction of v_j to the labels of the component K_i, then $v_j{\upharpoonright}_i \in L(\mathcal{A}^{I_j}_{r_j,r'_j}{\upharpoonright}_i)$ holds and for all $i \in I_j$ there exists a path in K_i from a state s_j^i to a state s'^i_j labeled by $v_j{\upharpoonright}_i$. Because of the bounded synchronization condition, for all $i \in I_j$ the restriction of $v_j{\upharpoonright}_i$ to the labels of the "normal" transitions is the same word. Note that $s'^i_j = s^i_k$ holds for $j < k$ if there is no path segment between π_j^{Sys} and π_k^{Sys} with associated components I with $i \in I$, i.e. there exists no s^i_m with $j < m < k$.
We define $v[i]$ to be the restriction of v to all segments with associated component i, i.e. $v[i] := v_{j_1}{\upharpoonright}_i \cdot v_{j_2}{\upharpoonright}_i \cdots v_{j_z}{\upharpoonright}_i$ where $j_1 < \cdots < j_z$, $i \in I_{j_1}, \ldots, I_{j_z}$ and $\forall j' \notin \{j_1, \ldots, j_z\} : i \notin I_{j'}$. It follows that for every component K_i of $\mathcal{A}_{Sys}$ there exists a path in K_i labeled by $v[i]$ and $v[i] \in L(z{\upharpoonright}_i)$. Thus, $L(z{\upharpoonright}_i) \cap L(K_i) \neq \emptyset$.

Proof (Completeness).
It is given that there exists a $z \in L(\mathcal{A}_{Glob})$ such that for all $i \in I$ there exist words v^i with $v^i \in L(z{\upharpoonright}_i) \cap L(K_i)$. Because of $v^i \in L(z{\upharpoonright}_i)$ the word v^i is decomposable into words $v^i_1 \cdots v^i_{len(i)}$ with $v^i_j \in L(t^i_j)$ and $t^i_j = \mathcal{A}^{I_j}_{r_j,r'_j,w}{\upharpoonright}_i$, respectively $t^i_j = \mathcal{A}^{I_j}_{r_j,r'_j}{\upharpoonright}_i$ for a local block $\mathcal{A}^{I_j}_{r_j,r'_j(,w)}$ of $\mathcal{A}_{Glob}$ that appears in z. From $v^i \in L(K_i)$ it follows that in the component K_i there exists a path $s^i_0 \overset{v^i_1}{\rightsquigarrow} s^i_1 \rightsquigarrow \cdots \overset{v^i_{len(i)}}{\rightsquigarrow} s^i_{len(i)}$ for states $s^i_j \in S_i$.
From the construction of $\mathcal{A}_{Glob}$ out of $\mathcal{A}_{Proj}$ we conclude that there exists a path labeled by w in $\mathcal{A}_{Proj}$ whereas the word w is defined as the result of the concatenation of words which are constructed out of the v^i_j in the order as the t^i_j appear in z. Formally we define $w := w_1 \ldots w_u$ with w_k as follows. For $t_k = \mathcal{A}^I_{r,r'}$ with singleton $I = \{i\}$ for an $i \in [n]$ there exists a path labeled with a word w_k. For $t_k = \mathcal{A}^I_{r,r',w_k}$ with synchronization profile $I = \{i_1, \ldots, i_c\}$ there exist $j_1, \ldots, j_c$ such that $t^{i_d}_{j_d} = \mathcal{A}^I_{r,r',w_k}{\upharpoonright}_{i_d}$ for all $d \in [c]$. Let $v^{i_1}_{j_1}, \ldots, v^{i_c}_{j_c}$ be the words

corresponding to $t_{j_d}^{i_d}$ ($d \in [c]$). We conclude that by definition of $\mathcal{A}^I_{r,r',w_k}\!\restriction_{i_d}$ ($d \in [c]$) and by the bounded synchronization condition there exists an interleaving of the words $v_{j_1}^{i_1}, \dots, v_{j_c}^{i_c}$ which admits a w_k-labeled path $\mathcal{A}^I_{r,r'}$, because for every component index the "same" w_k is chosen[2]. Thus, $w \in \mathcal{A}_{Proj}$ holds because of the definition of $\mathcal{A}_{Glob}$ from $\mathcal{A}_{Proj}$.
It remains to show that $w = w_1 \dots w_u \in L(\mathcal{A}_{Sys})$. Let $k \in \{0, \dots, u-1\}$ and $\bar{s}_0 := (s_0^1, \dots, s_0^n)$ be the initial state of the synchronized product. For a synchronization profile $I_{k+1} = \{i_1, \dots, i_c\}$ there exists a path from $\bar{s}_k$ to $\bar{s}_{k+1}$ by w_{k+1} because for all $d \in [c]$ there exist paths from $\bar{s}_k[i_d]$ to a state $s_{i_d} =: \bar{s}_{k+1}[i_d]$ by $v_{j_d}^{i_d}$ and the $v_{j_d}^{i_d}$ coincide in the non-predicate transitions, and for all other $i' \notin I_{k+1}$: $\bar{s}_{k+1}[i'] = s_k[i']$ holds. Thus by concatenating these segments, in $\mathcal{A}_{Sys}$ there exists a path from $\bar{s}_0$ to $\bar{s}_u$ by w. We conclude $w \in L(\mathcal{A}_{Sys}) \cap L(\mathcal{A}_{Proj})$. Because of the construction of $\mathcal{A}_{Proj}$ from $\mathcal{A}_{\overline{Spec}}$ there exists a word y with $w \in L(seq(y))$ and $y \in L(\mathcal{A}_{\overline{Spec}})$ and $w \in L(\mathcal{A}_{Sys})$. By Lemma 1 $y \in L(\mathcal{A}_{ESys})$ holds. Thus, $L(\mathcal{A}_{ESys}) \cap L(\mathcal{A}_{\overline{Spec}}) \neq \emptyset$.

Theorem 2 induces the following algorithm for checking whether $L(\mathcal{A}_{ESys}) \cap L(\mathcal{A}_{\overline{Spec}}) \neq \emptyset$ holds:

1. Construct $\mathcal{A}_{Proj}$ from $\mathcal{A}_{\overline{Spec}}$.
2. Construct $\mathcal{A}_{Glob}$ from $\mathcal{A}_{Proj}$.
3. For all local blocks $\mathcal{A}^{\{i\}}_{r,r'}/\mathcal{A}^I_{r,r',w}$ of $\mathcal{A}_{Glob}$ calculate the effect of $\mathcal{A}^{\{i\}}_{r,r'}/\mathcal{A}^I_{r,r',w}\!\restriction_i$ on K_i.
4. Check for all words $z \in L(\mathcal{A}_{Glob})$ up to a maximal length that $\forall i \in I$: $L(z\!\restriction_i) \cap L(K_i) \neq \emptyset$ by concatenating the local blocks in $z\!\restriction_i$.

We now justify the complexity claims of Theorem 2, by giving more precise complexity bounds for the induced algorithm. For a synchronized product we use the following parameters:

- n: the number of components
- l: the number of predicates
- r: the largest number of states of the components (the maximum over all n_i)

For a complement specification we use the parameters:

- s: the number of states
- w_{syn}: the number of synchronizing words for non-trivial synchronization profiles (Formally, w_{syn} is defined as $\sum_{I \subseteq [n], I \neq \{i\}} w^I_{syn}$ where w^I_{syn} is the (finite) number of words in the complement specification restricted to the synchronisation profile I.)

We show that the size of $\mathcal{A}_{Glob}$ is $\leq s^2 \cdot l \cdot n + s$ and the length of the word z can be restricted to $(s^2 + w_{syn}) \cdot l \cdot n \cdot 2^{r^2}$. Further, the complexity of the precalculation is at most $(s^2 + w_{syn}) \cdot l \cdot n \cdot [s^2 \cdot l]^{r \cdot s^2 (l \cdot n + 1)}$.

Proof (Complexity of the Algorithm).

1. Let t be the number of transitions of $\mathcal{A}_{Spec}$. The automaton $\mathcal{A}_{Proj}$ has $s_{Proj} \leq t \cdot l \cdot n + s \leq s^2 \cdot l \cdot n + s$ states and $t_{Proj} \leq t \cdot (l \cdot n + 1) \leq s^2 \cdot (l \cdot n + 1)$ transitions, because for each of the t transitions in $\mathcal{A}_{\overline{Spec}}$ there are at most $l \cdot n$ additional states and transitions in $\mathcal{A}_{Proj}$.

[2] "Same" means here that the words coincide if we omit the predicate transitions.

2. Let t_{sw} be the number of switching transitions. Because $\mathcal{A}_{Glob}$ compresses the check of transitions of the same synchronization profile into a local block, $\mathcal{A}_{Glob}$ has $s_{Glob} \leq t_{sw} \cdot l \cdot n \leq s^2 \cdot l \cdot n$ states and there are at most $t_{Glob} \leq (t_{sw} + w_{syn}) \cdot l \cdot n \leq (s^2 + w_{syn}) \cdot l \cdot n$ local blocks.
3. For each local block $\mathcal{A}^I_{r,r'(,w)}\lceil_i$ and each component K_i, we have to calculate the effect of $\mathcal{A}^I_{r,r'(,w)}\lceil_i$ on K_i. For this, first, we have to build the product automaton of $\mathcal{A}^I_{r,r'(,w)}\lceil_i$ and K_i (see the remark after Definition 7). The size of this product is at most $t_{Proj} \cdot r \leq s^2 \cdot (l \cdot n + 1) \cdot r$. Then, we have to check all words in the product up to the size of the product, i. e. at most $[s^2 \cdot l]^{r \cdot t_{Proj}}$ words. Thus, in total, we get a cost of $t_{Glob} \cdot [s^2 \cdot l]^{r \cdot t_{Proj}} \leq (s^2 + w_{syn}) \cdot l \cdot n \cdot [s^2 \cdot l]^{r \cdot s^2 (l \cdot n+1)}$ for this precalculation.
4. Let TP_i be the set of all transition profiles of K_i, $tp_i := |TP_i|$ the number of transition profiles for the i-th component and $tp := \max_{i \in I} tp_i$ the number of transition profiles of the largest component. Note that each automaton $\mathcal{A}^I_{r,r'(,w)}\lceil_i$ selects a subset of TP_i – the effect of $\mathcal{A}^I_{r,r'(,w)}\lceil_i$ on K_i.
For the estimation of the length of z let us note that on the one hand, the number t_{Glob} of transitions of $\mathcal{A}_{Glob}$ is not sufficient because $\mathcal{A}_{Glob}$ might have only one state with a loop ($t_{Glob} = 1$) and the effect of this loop might contain only one transition profile, e. g. $\tau = (1 \to 2, \ldots, n_{i-1} \to n_i)$. We could get other transition profiles like $(1 \to 3, \ldots, n_{i-1} \to 1)$ by repeating τ more than $t_{Glob} = 1$ times. On the other hand, tp is also not sufficient for the length of z because t_{Glob} might be much larger than tp, but for all but one letter the effect on K_i might be the identity $(1 \to 1, \ldots, n_i \to n_i)$ and the other effect might be $(1 \to 2, \ldots, n_{i-1} \to n_i)$. Thus, we could get other transition profiles with a length $> tp$ of z.
For $i \in I$ let RTP_i be the set of sets of transition profiles reachable by any word $z\lceil_i$ for $z \in L(\mathcal{A}_{Glob})$. We will now show that the length of z can be restricted to at most $tp \cdot t_{Glob} + 1$ by proving that RTP_i can already be reached by all words up to length $tp \cdot t_{Glob} + 1$.
As shown above, it could be the case that the effect in K_i of a loop in $\mathcal{A}_{Glob}$ contains only one transition profile τ different from the identity. The largest loop without repeating more than one state has to be of size $\leq t_{Glob}$, so after at least $t_{Glob} + 1$ letters we can be sure that we either have already reached RTP_i or the effect of this word is a transition profile different from t. By repeating any loops of size $\leq t_{Glob}$ at most tp_i times we can reach RTP_i. Thus, the size of all $z\lceil_i$ can be bounded by $t_{Glob} \cdot tp$. We conclude that the maximal length of the word z is $t_{Glob} \cdot tp \leq ((s^2 + w_{syn}) \cdot l \cdot n) \cdot (2^{r^2})$.
For each of these words z the effect of $z\lceil_i$ on K_i has to be calculated. Because the effect of the concatenation of local blocks on K_i is determined by the concatenation of the effects, we can simply concatenate the precalculated effects. Thus, we have to check at most $(n \cdot t_{Glob})^{t_{Glob} \cdot tp} \leq ((s^2 + w_{syn}) \cdot l \cdot n^2)^{((s^2+w_{syn}) \cdot l \cdot n) \cdot (2^{r^2})}$ words.

The generalization to Büchi automata as in Sect. 3 also works in the case with predicates. The complexity differs only by a constant factor.

The proof idea for the generalization to Büchi automata is the following: Consider an infinite path in the projection of the complement specification automaton and its decomposition into path segments π_j^{Proj} which use only labels of exactly one synchronization profile (and subsequent segments cannot be joined while remaining this property). Then there are two cases: either the last path segment is infinite or there are infinitely many (finite) path segments with different synchronization profiles. This amounts to the two cases for the word z in the global specification: either z is finite and the last local block is a Büchi automaton or z is an infinite word of the form $z = uv^\omega$ for finite u, v and all local blocks are finite automata.

Note that the generalization to Büchi automata only adds a factor 2 to the length of the word z. For the first case, where a finite z is considered we can use the same estimation as in the proof of Theorem 1. We want to remark, that for the last local block of z we have to calculate the effect of a Büchi automaton on a finite automaton. However, this can be done in an analogous way to the case of a finite automaton. For the second case where z is infinite of the form $z = uv^\omega$, we can use the known estimation for both subwords u and v. Thus, together we get a complexity which is twice the complexity of the known algorithm.

6 Further Results and Conclusion

We have presented a compositional approach for reducing failure detection in a product of transition systems to the components, working in an automata theoretic rather than a logical framework. The method allows us to reduce the question whether a product of transition systems and a given complement specification automaton (with bounded synchronization) have paths with a common labeling to the question whether a path in the global specification automaton exists such that the components and parts of the complement specification which are described by the local blocks of the global specification automaton have paths with common labeling. The composition method uses information about the transitions in the product – their synchronization profiles – to split the complement specification automaton into parts. Further, we have shown that the complexity of the induced algorithm is at most double exponential in the number of states of the largest component and exponential in the number of components, in the number of states and predicates the complement specification has. We want to remark that the restriction to bounded synchronization can be removed: Consider a part of the specification with a fixed synchronization profile and unbounded synchronization (i. e. this part contains a loop). This part cannot be checked as a single local block, because there is no guarantee that in all components of the synchronization profile the same word is chosen. Therefore, we have to use local blocks for every letter of this part. Of course, in general, we have to pay in complexity for these parts of unbounded synchronization. However, as we estimated the complexity by s^2 in Section 5 the worst case complexity does not change.

The results in this paper complement research on synchronized state/event systems [BVF+99] in which the descriptional framework is modal logic, and where model-checking is done by a reduction of the product index set while transforming the given specification (formula). As another related paper we mention [CK96] where a different set-up for specifying synchronization is used (via "interface processes").

We mention that the present technique can be improved further in appropriate scenarios: E. g., one could use the fact that in the complement specification successive transitions of the same component must have the same valuation of the predicates of the other components, to reduce the number of transitions by deleting transitions where this is not the case. A second improvement would be to duplicate states with incoming transitions of different components and thereby to split the different paths. However, one would have to ensure that this procedure does terminate by considering the decomposition of the complement specification automaton into strongly connected components and aborting the procedure if we reach the same state of a loop again.

Let us mention two possible generalizations: First, one should aim at a composition theorem directly for the language inclusion problem $L(\mathrm{Sys}) \subseteq L(\mathrm{Spec})$, because it is more natural to specify the desired behaviour of a system than the behaviour of the complement. Second, one should get a deeper understanding of the technique by looking at how the decomposition of the complement specification automaton can be translated to a decomposition of a logical formula. Therefore, one could consider e. g. a variation of linear time temporal logic (LTL) with additional information about the components, respectively the synchronization profile on parts of the formula. Linear time temporal logic is here a better candidate than classical first-order logic.

References

AHU74. Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison Wesley, Reading, Massachusetts, 1974.

BK08. Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking.* MIT Press, 2008.

Büc60. J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundladen der Mathematik*, 6:66–92, 1960.

BVF+99. Nicky O. Bodentien, Jacob Vestergaard, Jakob Friis, Kåre J. Kristoffersen, and Kim G. Larsen. Verification of state/event systems by quotienting. *BRICS*, RS-99-41, December 1999. *Nordic Workshop in Programming Theory*, Uppsala, Sweden, October 6–8, 1999.

CK90. Chen Chung Chang and H. J. Keisler. *Model Theory.* North Holland, Amsterdam, 1990.

CK96. Shing Chi Cheung and Jeff Kramer. Context constraints for compositional reachability analysis. *ACM Trans. Softw. Eng. Methodol.*, 5:334–377, October 1996.

DGKS07. Anuj Dawar, Martin Grohe, Stephan Kreutzer, and Nicole Schweikardt. Model theory makes formulas large. In *ICALP'07: 34th Int. Colloquium on Automata, Languages and Programming*, volume 4596 of *Lecture Notes in Computer Science*, pages 913–924. Springer, 2007.

Elg61. C.C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–52, 1961.

Fel08. Ingo Felscher. The compositional method and regular reachability. *Electronic Notes in Theoretical Computer Science*, 223:103–117, December 2008.

FT09. Ingo Felscher and Wolfgang Thomas. Compositionality and reachability with conditions on path lengths. *Int. Journal of Foundations of Computer Science*, 20(5):851–868, May 2009.

FT11. Ingo Felscher and Wolfgang Thomas. Compositional failure detection in structured transition systems. *Lecture Notes of Computer Science*, 6807:130–141, 2011.

FV59. S. Feferman and R. Vaught. The first-order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.

GS98. D.M. Gabbay and V.B. Shehtman. Products of modal logics, part 1. *Logic Journal of IGPL*, 6(1):73–146, 1998.

Hod93. Wilfrid Hodges. *Model theory*, volume 42 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, Cambridge, 1993.

KN01. Bakhadyr Khoussainov and Anil Nerode. *Automata Theory and its Applications*. Progress in Computer Science and Applied Logic (PCS), Vol. 21. Birkhäuser Boston, 2001.

Mak04. Johann A. Makowsky. Algorithmic uses of the Feferman-Vaught theorem. *Annals of Pure and Applied Logic*, 126(1-3):159–213, 2004.

Mos52. Andrzej Mostowski. On direct products of theories. *The Journal of Symbolic Logic*, 17(1):1–31, 1952.

Rab07. Alexander Rabinovich. On compositionality and its limitations. *ACM Transactions on Computational Logic*, 8(1), January 2007.

She75. Saharon Shelah. The monadic theory of order. *The Annals of Mathematics*, 102(3):379–419, 1975.

Tho97a. Wolfgang Thomas. Ehrenfeucht games, the composition method, and the monadic theory of ordinal words. In Jan Mycielski, Grzegorz Rozenberg, and Arto Salomaa, editors, *Structures in Logic and Computer Science, A Selection of Essays in Honor of A. Ehrenfeucht*, volume 1261 of *LNCS*, pages 118–143. Springer–Verlag, 1997.

Tho97b. Wolfgang Thomas. Languages, automata and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages, vol. 3 beyond words*, volume 3, pages 389–455. Springer–Verlag, New York, NY, USA, 1997.

Wol01. Pierre Wolper. Constructing automata from temporal logic formulas: A tutorial. In Ed Brinksma, Holger Hermanns, and Joost-Pieter Katoen, editors, *Lec. on Form. Meth. and Performance Analysis*, volume 2090 of *LNCS*, pages 261–277. Springer Berlin/Heidelberg, 2001.

WT04. Stefan Wöhrle and Wolfgang Thomas. Model checking synchronized products of infinite transition systems. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, LNCS, pages 2–11, Washington, DC, USA, 2004. IEEE Computer Society.

Aachener Informatik-Berichte

This list contains all technical reports published during the past three years. A complete list of reports dating back to 1987 is available from `http://aib.informatik.rwth-aachen.de/`**. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email:** `biblio@informatik.rwth-aachen.de`

2008-01 * Fachgruppe Informatik: Jahresbericht 2007

2008-02 Henrik Bohnenkamp, Marielle Stoelinga: Quantitative Testing

2008-03 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, Harald Zankl: Maximal Termination

2008-04 Uwe Naumann, Jan Riehme: Sensitivity Analysis in Sisyphe with the AD-Enabled NAGWare Fortran Compiler

2008-05 Frank G. Radmacher: An Automata Theoretic Approach to the Theory of Rational Tree Relations

2008-06 Uwe Naumann, Laurent Hascoet, Chris Hill, Paul Hovland, Jan Riehme, Jean Utke: A Framework for Proving Correctness of Adjoint Message Passing Programs

2008-07 Alexander Nyßen, Horst Lichter: The MeDUSA Reference Manual, Second Edition

2008-08 George B. Mertzios, Stavros D. Nikolopoulos: The λ-cluster Problem on Parameterized Interval Graphs

2008-09 George B. Mertzios, Walter Unger: An optimal algorithm for the k-fixed-endpoint path cover on proper interval graphs

2008-10 George B. Mertzios, Walter Unger: Preemptive Scheduling of Equal-Length Jobs in Polynomial Time

2008-11 George B. Mertzios: Fast Convergence of Routing Games with Splittable Flows

2008-12 Joost-Pieter Katoen, Daniel Klink, Martin Leucker, Verena Wolf: Abstraction for stochastic systems by Erlang's method of stages

2008-13 Beatriz Alarcón, Fabian Emmes, Carsten Fuhs, Jürgen Giesl, Raúl Gutiérrez, Salvador Lucas, Peter Schneider-Kamp, René Thiemann: Improving Context-Sensitive Dependency Pairs

2008-14 Bastian Schlich: Model Checking of Software for Microcontrollers

2008-15 Joachim Kneis, Alexander Langer, Peter Rossmanith: A New Algorithm for Finding Trees with Many Leaves

2008-16 Hendrik vom Lehn, Elias Weingärtner and Klaus Wehrle: Comparing recent network simulators: A performance evaluation study

2008-17 Peter Schneider-Kamp: Static Termination Analysis for Prolog using Term Rewriting and SAT Solving

2008-18 Falk Salewski: Empirical Evaluations of Safety-Critical Embedded Systems

2008-19 Dirk Wilking: Empirical Studies for the Application of Agile Methods to Embedded Systems

2009-02 Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Quantitative Model Checking of Continuous-Time Markov Chains Against Timed Automata Specifications

2009-03 Alexander Nyßen: Model-Based Construction of Embedded Real-Time Software - A Methodology for Small Devices
2009-04 Daniel Klünder: Entwurf eingebetteter Software mit abstrakten Zustandsmaschinen und Business Object Notation
2009-05 George B. Mertzios, Ignasi Sau, Shmuel Zaks: A New Intersection Model and Improved Algorithms for Tolerance Graphs
2009-06 George B. Mertzios, Ignasi Sau, Shmuel Zaks: The Recognition of Tolerance and Bounded Tolerance Graphs is NP-complete
2009-07 Joachim Kneis, Alexander Langer, Peter Rossmanith: Derandomizing Non-uniform Color-Coding I
2009-08 Joachim Kneis, Alexander Langer: Satellites and Mirrors for Solving Independent Set on Sparse Graphs
2009-09 Michael Nett: Implementation of an Automated Proof for an Algorithm Solving the Maximum Independent Set Problem
2009-10 Felix Reidl, Fernando Sánchez Villaamil: Automatic Verification of the Correctness of the Upper Bound of a Maximum Independent Set Algorithm
2009-11 Kyriaki Ioannidou, George B. Mertzios, Stavros D. Nikolopoulos: The Longest Path Problem is Polynomial on Interval Graphs
2009-12 Martin Neuhäußer, Lijun Zhang: Time-Bounded Reachability in Continuous-Time Markov Decision Processes
2009-13 Martin Zimmermann: Time-optimal Winning Strategies for Poset Games
2009-14 Ralf Huuck, Gerwin Klein, Bastian Schlich (eds.): Doctoral Symposium on Systems Software Verification (DS SSV'09)
2009-15 Joost-Pieter Katoen, Daniel Klink, Martin Neuhäußer: Compositional Abstraction for Stochastic Systems
2009-16 George B. Mertzios, Derek G. Corneil: Vertex Splitting and the Recognition of Trapezoid Graphs
2009-17 Carsten Kern: Learning Communicating and Nondeterministic Automata
2009-18 Paul Hänsch, Michaela Slaats, Wolfgang Thomas: Parametrized Regular Infinite Games and Higher-Order Pushdown Strategies
2010-02 Daniel Neider, Christof Löding: Learning Visibly One-Counter Automata in Polynomial Time
2010-03 Holger Krahn: MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering
2010-04 René Wörzberger: Management dynamischer Geschäftsprozesse auf Basis statischer Prozessmanagementsysteme
2010-05 Daniel Retkowitz: Softwareunterstützung für adaptive eHome-Systeme
2010-06 Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Computing maximum reachability probabilities in Markovian timed automata
2010-07 George B. Mertzios: A New Intersection Model for Multitolerance Graphs, Hierarchy, and Efficient Algorithms
2010-08 Carsten Otto, Marc Brockschmidt, Christian von Essen, Jürgen Giesl: Automated Termination Analysis of Java Bytecode by Term Rewriting
2010-09 George B. Mertzios, Shmuel Zaks: The Structure of the Intersection of Tolerance and Cocomparability Graphs

2010-10 Peter Schneider-Kamp, Jürgen Giesl, Thomas Ströder, Alexander Serebrenik, René Thiemann: Automated Termination Analysis for Logic Programs with Cut

2010-11 Martin Zimmermann: Parametric LTL Games

2010-12 Thomas Ströder, Peter Schneider-Kamp, Jürgen Giesl: Dependency Triples for Improving Termination Analysis of Logic Programs with Cut

2010-13 Ashraf Armoush: Design Patterns for Safety-Critical Embedded Systems

2010-14 Michael Codish, Carsten Fuhs, Jürgen Giesl, Peter Schneider-Kamp: Lazy Abstraction for Size-Change Termination

2010-15 Marc Brockschmidt, Carsten Otto, Christian von Essen, Jürgen Giesl: Termination Graphs for Java Bytecode

2010-16 Christian Berger: Automating Acceptance Tests for Sensor- and Actuator-based Systems on the Example of Autonomous Vehicles

2010-17 Hans Grönniger: Systemmodell-basierte Definition objektbasierter Modellierungssprachen mit semantischen Variationspunkten

2010-18 Ibrahim Armaç: Personalisierte eHomes: Mobilität, Privatsphäre und Sicherheit

2010-19 Felix Reidl: Experimental Evaluation of an Independent Set Algorithm

2010-20 Wladimir Fridman, Christof Löding, Martin Zimmermann: Degrees of Lookahead in Context-free Infinite Games

2011-02 Marc Brockschmidt, Carsten Otto, Jürgen Giesl: Modular Termination Proofs of Recursive Java Bytecode Programs by Term Rewriting

2011-03 Lars Noschinski, Fabian Emmes, Jürgen Giesl: A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems

2011-04 Christina Jansen, Jonathan Heinen, Joost-Pieter Katoen, Thomas Noll: A Local Greibach Normal Form for Hyperedge Replacement Grammars

2011-07 Shahar Maoz, Jan Oliver Ringert, Bernhard Rumpe: An Operational Semantics for Activity Diagrams using SMV

2011-08 Thomas Ströder, Fabian Emmes, Peter Schneider-Kamp, Jürgen Giesl, Carsten Fuhs: A Linear Operational Semantics for Termination and Complexity Analysis of ISO Prolog

2011-11 Nils Jansen, Erika Ábrahám, Jens Katelaan, Ralf Wimmer, Joost-Pieter Katoen, Bernd Becker: Hierarchical Counterexamples for Discrete-Time Markov Chains

2011-13 Michael Förster, Uwe Naumann, Jean Utke: Toward Adjoint OpenMP

2011-14 Daniel Neider, Roman Rabinovich, Martin Zimmermann: Solving Muller Games via Safety Games

2011-16 Niloofar Safiran, Uwe Naumann: Toward Adjoint OpenFOAM

2011-18 Kamal Barakat: Introducing Timers to pi-Calculus

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.