Compositional Analysis of Deadlock-Freedom for Tree-Like Component Architectures

Mila Majster-Cederbaum
Department of Computer Science, Universität Mannheim, Germany
mcb@informatik.uni-mannheim.de

Moritz Martens
Department of Computer Science, Universität Mannheim, Germany
mmartens@informatik.uni-mannheim.de

ABSTRACT

We study architectural constraints for component systems in order to be able to guarantee safety-properties. Representing safety-properties, we investigate deadlock-freedom. We present a compositional and hence polynomial time condition for deadlock-freedom for a class of component-systems whose architecture is tree-like. The architectural constraints that are developed can be understood as a design pattern that helps to construct systems satisfying safety-properties on the one hand. On the other hand, they might help to draw attention to potentially critical situations in a design. To model component-systems we use the formalism of interaction systems as proposed by Sifakis et al. The ideas can be transferred to other formal models where subsystems are cooperating via synchronous communication.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal methods*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*Mechanical verification and specification techniques*

General Terms

Design, Theory, Verification

Keywords

architecture, component-based systems, deadlock-freedom, design patterns, interaction systems, compositionality

1. INTRODUCTION

Component-based design techniques are an important paradigm for mastering design complexity and enhancing reusability of distributed systems. In contrast to the object-oriented approach where subsystems interact by explicitly accessing operations of other subsystems in their code, components are designed to be as independent as possible from their context of use. Ideally a component may be deployed in any meaningful environment that can use the component's functionality. Thus a component may not invoke operations or data of other components. Instead a component provides ports which can be used to glue several components together using some kind of separate gluing mechanism. This view has lead some authors, e.g., [2, 11, 29], to consider a component as a black-box in the sense that only the input/output behavior at the ports of the component is described explicitly. However, if we want to make assertions about the global behavior of such a system, be it functional, temporal, or quantitative, knowledge about the components has to be provided and we have to abandon the black-box point of view of a component.

There have been different approaches to model the internal behavior of a component, e.g., Petri-nets [5], process algebra [1, 30] or channel-based methods [10]. One approach to investigate generic properties such as deadlock-freedom, progress, or liveness of component-systems is model-checking, where in general the global state space is analyzed. Other approaches require certain conditions on the glue-code and on the components to be composed to ensure desired properties [1, 3, 21]. A first approach to use architectural constraints to address the problem of deadlock-freedom can be found in [7, 8].

The formalism of *interaction systems* was proposed and discussed by Sifakis et al. in [17, 18, 20, 33]. We build on this formalism in order to study component-systems. Interaction systems have been implemented in the BIP- [6] as well as the PROMETHEUS-tool [14]. The model is used as a common formal platform in the EU project SPEEDS [4, 9].

The model is particularly well suited to model component-systems as it strictly separates the description of the components from the way they are glued together. Each component i has a *static* interface description, which is given by a set $\mathcal{A}_i$ of *ports*. The *dynamic behavior* of component i is modeled by a labeled transition system T_i where the labels are the elements of $\mathcal{A}_i$. Components are glued together via *connectors* which constitute an independent layer of description. A connector is a set of ports, such that each component participates with at most one port, and describes a cooperation among components. The same set of components can be glued together differently (i.e., with other connectors) for different applications. The behavior of the global system is fully determined by the static and dynamic description of the components and by the connectors.

Theoretical results [26, 27] show that deciding virtually any important property of interaction systems is PSPACE-complete. In order to deal with this situation one can conceive various strategies:

1. establishing conditions for general component-systems that ensure the desired properties and can be tested in polynomial time
2. abstraction
3. exploiting compositionality
4. restricting the architecture of the component-systems

In [16, 15, 24, 23, 25] we took the first approach for general interaction systems. First attempts to exploit compositionality can be found in [24, 21, 3, 1]. In this paper we consider a combination of the last two strategies and deal with deadlock-freedom which is an important property in itself. Moreover checking safety-properties can be reduced to checking deadlock-freedom [13].

We consider tree-like interaction systems, i.e., the communication structure forms a tree. We present conditions that only refer to neighbors in the tree but ensure deadlock-freedom in the global system. A similar approach was taken in [7, 8]. For example, the basic idea of [8] is that whenever a component cooperates with an inner node of the tree then the behavior of this inner node should not be confined by the cooperation. This is a rather rigid restriction. We present a criterion that computes an over-approximation of the projection of the set containing all reachable global states to any subsystem consisting of two neighbors in the tree. Then we check a condition which basically ensures that no states which could be involved in a global deadlock are reachable in these subsystems.

The paper is structured as follows. In Sect. 2, we formally introduce interaction systems and the notion of deadlock-freedom. In Sect. 3, we present the tree-like architecture we consider. Section 4 contains an example. The main result and its application to the example can be found in Sect. 5. In Sect. 6, we shortly discuss the result and compare our criterion with other results. Section 7 concludes the paper.

2. INTERACTION SYSTEMS

We build on a model for component-based systems, called *interaction systems*, that was proposed by Sifakis et al. in [17, 18, 19, 32, 33]. Interface automata [12] and I/O-automata [22] can be considered as special cases of interaction systems.

Definition 1. An *interaction system* is defined by means of a tuple $Sys := (K, \{\mathcal{A}_i\}_{i\in K}, C, \{T_i\}_{i\in K})$. Here $K = \{1, \ldots, n\}$ is a finite set of *components*. Components are referred to as $i \in K$. The *ports* or *actions* of component i are given by the *port set* $\mathcal{A}_i$. A *connector* c describes a possible cooperation between several components and is a nonempty set of ports that contains at most one port of every component. All allowed cooperations are described by the *connector set* C which is a set of connectors such that every port of every component is contained in at least one connector and all connectors are maximal with respect to set-inclusion[1]. For $i \in K$ and $c \in C$ we put $i(c) = \mathcal{A}_i \cap c$. We say that i *participates* in c if $i(c) \neq \emptyset$. Finally $T_i = (Q_i, \mathcal{A}_i, \to_i, q_i^0)$ is a transition system describing the *local behavior* of $i \in K$ where $\to_i \subseteq Q_i \times \mathcal{A}_i \times Q_i$ and $q_i^0 \in Q_i$ is the *initial state*. We write $q_i \xrightarrow{a_i}_i q_i'$ instead of $(q_i, a_i, q_i') \in \to_i$ and define $en(q_i) := \{a_i | \exists q_i' \in Q_i : q_i \xrightarrow{a_i}_i q_i'\}$ the set of ports that are *enabled* in q_i.

The *global behavior* $T = (Q, C, \to, q^0)$ of Sys is obtained in a straightforward manner. The set of *(global) states* is given by $Q := \prod_{i\in K} Q_i$. States are denoted by tuples $q := (q_1, \ldots, q_n)$. The *(global) initial state* is $q^0 := (q_1^0, \ldots, q_n^0)$. The *transition relation* $\to \subseteq Q \times C \times Q$ for Sys is defined canonically where for any $c \in C$ and any $q, q' \in Q$ we have $q \xrightarrow{c} q'$ if and only if for all $i \in K$ we have $q_i \xrightarrow{i(c)}_i q_i'$ if $i(c) \neq \emptyset$ and $q_i' = q_i$ otherwise. A connector c is *enabled* in q if for all i that participate in c the port $i(c)$ is enabled in q_i.

Remark 1. For better readability we identified singleton sets with their element in the previous definition. We will do so throughout the whole paper.

We need a notion of subsystem of Sys with respect to a subset $K' \subseteq K$.

Definition 2. Let Sys be an interaction system as above and let $K' \subseteq K$ be a subset of components. The projection of Sys to K' is defined by

$$Sys\downarrow_{K'} := (K', \{\mathcal{A}_i\}_{i\in K'}, C[K'], \{T_i\}_{i\in K'})$$

where $C[K'] := maxel\{c\downarrow_{K'} | c \in C\}$ is the connector set. Here $c\downarrow_{K'} := \{a_j | a_j \in c \wedge j \in K'\}$, and $maxel$ is the operator that maps a set S of sets to the set of all sets contained in S that are maximal with respect to set inclusion.

Next we present the notion of deadlock-freedom for interaction systems.

Definition 3. Let Sys be an interaction system. A state q is a *deadlock state* if no $c \in C$ is enabled in q. If there are no reachable (from q^0) deadlock states then Sys is called *deadlock-free*.

Deadlock-freedom is not only important from the system-design point of view where it is desirable that a system does not reach a state that cannot be left any more. Also, the violation of a safety-property can be modeled as a deadlock [13]. Thus a broad range of properties can be treated when we are able to handle deadlocks. Checking deadlock-freedom is difficult. It has been shown that deciding deadlock-freedom for interaction systems is PSPACE-complete [27]. As explained in the introduction there are various strategies to address this problem. Here we exploit compositionality and restrict the structure of the component-systems.

3. TREE-LIKE ARCHITECTURES

We consider a tree-like architecture for interaction systems. Tree-like component-systems arise naturally in many applications (see [7, 8, 31] for example).

Definition 4. Let Sys be an interaction system. Define a graph $G := (K, E)$ where the set of edges is given by $E = \{\{i, j\} \mid \exists c \in C \text{ such that } i \text{ and } j \text{ participate in } c\}$. Denote the set of neighbors of i by $nb(i)$. If G is a tree we say that Sys is *tree-like*.

[1]Note that even the subclass of interaction systems only allowing binary communication offers more flexibility than process calculi (e.g., as CCS) since one port may appear in various connectors.

Remark 2. For a tree-like system Sys the above definition implies $|c| \leq 2$ for all $c \in C$. For simplicity we assume that all communications are binary. The results can be easily adapted to architectures that also allow for connectors c with $|c| = 1$.[2]

The set of ports of component i that are used for communication with $j \in nb(i)$ is introduced as follows.

Definition 5. Let Sys be a tree-like interaction system, let $i \in K$ and $j \in nb(i)$ be components. Define $\mathit{comm}_i(j) := \{a_i \in \mathcal{A}_i | \exists c \in C$ such that j participates in c and $i(c) = a_i\}$ the set of ports of i that are used for communication with j.

We now define a requirement about exclusive communication. This requirement is rather technical. We need it to derive the reachability of (q_i, q_j) in $Sys \downarrow_{\{i,j\}}$ from the reachability of a global state q in Sys.

Definition 6. A tree-like interaction system Sys has *exclusive communication* if it satisfies the following condition for any components i and $j \neq k \in nb(i)$: $\mathit{comm}_i(j) \cap \mathit{comm}_i(k) = \emptyset$.

Remark 3. For any i that is a leaf in G the condition in the previous definition is trivially satisfied.

Whenever Sys has exclusive communication, for all $i \in K$ we get a partition of $\mathcal{A}_i$ into the sets $\mathit{comm}_i(j)$ where j ranges over the neighbors of i in G. The assumption that a tree-like interaction system Sys has exclusive communication is in fact not a restriction after all. It is possible to construct (in polynomial time) a system that exhibits the same behavior as Sys and has exclusive communication.

LEMMA 1. *Let Sys be a tree-like interaction system that does not have exclusive communication. Then there exists a tree-like system Sys' with $K' = K$ and $Q' = Q$ such that for any $q, \bar{q} \in Q$ we have:*

$$\exists c \in C \text{ such that } q \xrightarrow{c} \bar{q} \Leftrightarrow \exists c' \in C' \text{ such that } q \xrightarrow{c'} \bar{q}.$$

Sys' has exclusive communication.

There is no exponential blowup in the construction of Sys'.

The basic idea of the construction of Sys' is as follows. For all $i \in K$ we replace the port $a_i \in \mathcal{A}_i$ by a set of new ports a_i^j where j ranges over those components for which there is a connector c with $a_i \in c$ and $j(c) \neq \emptyset$. In T_i we substitute each edge labeled with a_i by a set of parallel edges, one for each a_i^j. The connector set C' is then obtained by replacing each $c = \{a_i, a_j\}$ by $c' = \{a_i^j, a_j^i\}$.

COROLLARY 1. *Let Sys be a tree-like interaction system that does not have exclusive communication and let Sys' be defined as above.*

1. *A state q is reachable in Sys if and only if it is reachable in Sys'.*
2. *A state $q \in Q$ is a deadlock state of Sys if and only if it is a deadlock state of Sys'.*

The first part can be shown using Lemma 1 and induction on the length of a path from q^0 to q. The second part follows directly from the lemma. Thus whenever we investigate deadlock-freedom of a tree-like system Sys without exclusive communication we may construct and investigate Sys' instead.

[2]Connectors of size one can be interpreted as internal actions.

4. AN EXAMPLE

We present an example that shows how a banking system which is similar to the banking system described in [7] can be modeled using the concept of interaction systems. The example has been chosen as to be somewhat complex and realistic, but still simple enough to verify the results by hand. The system consists of the following components. There are m banks represented by components b_i for $i \in \{1, \ldots, m\}$. For each b_i there are $m_i \geq 1$ ATM components atm_j^i where $j \in \{1, \ldots, m_i\}$. Finally there is a clearing company cc. We get $K_{bank} := \left\{cc, b_i, atm_j^i | 1 \leq i \leq m, 1 \leq j \leq m_i\right\}$. Instead of listing the port-sets of the components explicitly we give the local behavior of each $k \in K_{bank}$ in Fig. 1. Each ATM can request money from its bank which then checks the correctness of the PIN delivered with the clearing company. Depending on the reply of the clearing company, the ATM may disburse money or cancel the transaction. We abstract from the actual values of the PIN and the money on the accounts, since we only want to model the communication between the components. Keeping track of the values could be done by introducing local variables for the components for example. Alternatively the different values could be incorporated into the states and actions of the components. The following connectors describe the

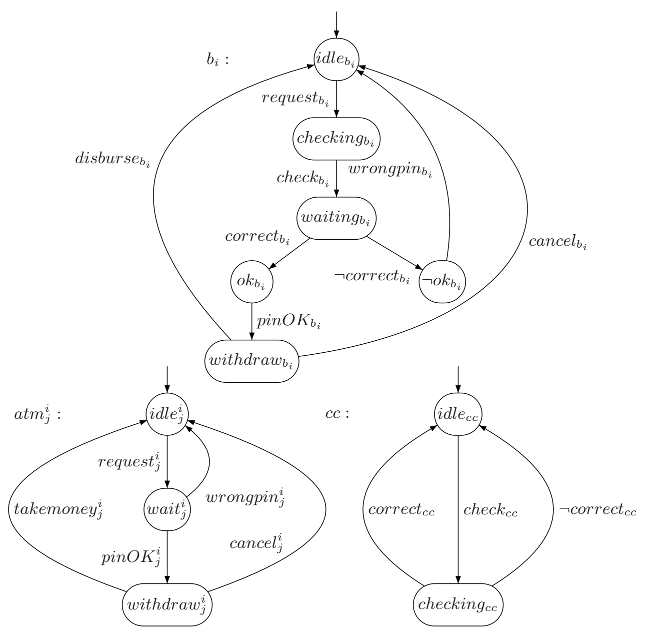

Figure 1: The local behavior of the components of Sys_{bank}

allowed interactions between the components. Each bank b_i communicates with the clearing company by means of the following set of connectors: $C_i := \{\{check_{b_i}, check_{cc}\}, \{\neg correct_{b_i}, \neg correct_{cc}\}, \{correct_{b_i}, correct_{cc}\}\}$. For every ATM atm_j^i the following set of connectors allow for it to synchronize with its bank: $C_j^i := \left\{\left\{wrongpin_{b_i}, wrongpin_j^i\right\}, \left\{req_{b_i}, request_j^i\right\}, \left\{cancel_{b_i}, cancel_j^i\right\}, \left\{pinOK_{b_i}, pinOK_j^i\right\}, \left\{disburse_{b_i}, takemoney_j^i\right\}\right\}$ (for limitations of space we set $req_{b_i} := request_{b_i}$). Then we define $C_{bank} := \bigcup_{i=1}^m C_i \cup \bigcup_{i=1,j=1}^{m,m_i} C_j^i$. We have now completely specified $Sys_{bank} := (K_{bank}, \{\mathcal{A}_i\}_{i \in K_{bank}}, C_{bank}, \{T_i\}_{i \in K_{bank}})$. It is tree-like but does not have exclusive communication. For example there are actions of cc that are used for communication with various banks. Instead we investigate Sys'_{bank} from Lemma 1.

5. COMPOSITIONAL TESTING OF DEADLOCK-FREEDOM IN TREE-LIKE INTERACTION SYSTEMS

This section contains the sufficient criterion for deadlock-freedom of tree-like interaction systems. From now on we assume that for all $i \in K$ and all $q_i \in Q_i$ we have $\mathit{en}(q_i) \neq \emptyset$, i.e., all local states of all components offer at least one port. Before stating the criterion we will first motivate and define step-by-step the concepts needed. These concepts will be merged in Proposition 1. Exemplarily we will illustrate the notions and the result by means of Sys'_{bank} defined in the previous section.

Our goal is to infer deadlock-freedom from conditions that can be checked on subsystems of the form $Sys\downarrow_{\{i,j\}}$ where $\{i,j\} \in E$. In each of these subsystems we want to detect those states that might contribute to a reachable global deadlock. To be able to pinpoint these states, for every $i \in K$ and every local state q_i we distinguish those neighbors of i in G, that are possible communication partners for q_i from those that are not.

Definition 7. Let Sys be a tree-like interaction system that has exclusive communication. For $i \in K$ and $q_i \in Q_i$ we denote the set of components one of which is needed for cooperation by i in state q_i by

$$need(q_i) := \{j \in nb(i) \,|\, \mathit{en}(q_i) \cap \mathit{comm}_i(j) \neq \emptyset\}.$$

Note that in order for i to be able to participate in a connector in q_i it suffices if one of the components in $need(q_i)$ enables one of the required actions. Further note that $i \notin need(q_i)$, and $need(q_i) \neq \emptyset$ because of our assumptions $|c| = 2$ and $\mathit{en}(q_i) \neq \emptyset$. In Sys'_{bank} from Sect. 4 all states q_{cc} satisfy $need(q_{cc}) = \{b_1, \ldots, b_m\}$ and for any $q_{atm^i_j}$ we have $need(q_{atm^i_j}) = \{b_i\}$ since b_i is the only neighbor of atm^i_j. On the other hand for b_i we get $need(checking_{b_i}) = \{cc\}$ whereas $need(idle_{b_i}) = \{atm^i_1, \ldots, atm^i_{m_i}\}$, for example.

Going back to $Sys\downarrow_{\{i,j\}}$ it is clear that any reachable state (q_i, q_j), where a connector $c = \{a_i, a_j\} \in C[\{i,j\}]$ is enabled, can never be part of a global deadlock state because the same connector c will be enabled in any global state whose entries for components i and j coincide with (q_i, q_j). Further we may conclude that a state (q_i, q_j) where at least one of the two components is not a possible communication partner for the other (i.e., $j \notin need(q_i)$ or vice versa) is also not problematic at least from the point of view of components i and j. This does not mean that this kind of a state will never be part of a global deadlock state, but if it is we will have to detect this global deadlock by looking at a different pair of components. Conversely whenever there is a reachable state (q_i, q_j) where $i \in need(q_j)$ and $j \in need(q_i)$ but no connector from $C[\{i,j\}]$ involving both components is enabled we know that this state can possibly be part of a reachable global deadlock. This means that a sufficient condition for deadlock-freedom that only investigates the subsystems $Sys\downarrow_{\{i,j\}}$ for $\{i,j\} \in E$ must detect these kinds of states. First we want to state more precisely when a local state of component j is problematic with respect to $q_i \in Q_i$.

Definition 8. Let Sys be a tree-like interaction system that has exclusive communication. For $i \in K$, $q_i \in Q_i$, and $j \in need(q_i)$ we define the set of states of j that are *problematic with respect to* q_i:

$$\begin{aligned} PS_j(q_i) := \{q_j | i \in need(q_j) \text{ and } (q_i, q_j) \text{ is} \\ \text{reachable in } Sys\downarrow_{\{i,j\}}, \text{ but no } c \in C[\{i,j\}] \\ \text{involving } i \text{ and } j \text{ is enabled in } (q_i, q_j)\} \end{aligned}$$

As motivated above a state q_j is problematic with respect to q_i if in these states both components are ready to cooperate with each other but no cooperation is possible because the suitable ports are not available. Note that this characterization is symmetric. The same holds for the above formal definition and thus we have $q_j \in PS_j(q_i)$ if and only if $q_i \in PS_i(q_j)$. When we consider the state $idle_{cc}$ and some b_i in Sys'_{bank} we see that $PS_{b_i}(idle_{cc}) = \{waiting_{b_i}\}$ because the only other state of b_i for which cc is a possible communication partner is $checking_{b_i}$. But for this state the connector $\{check_{cc}, check_{b_i}\}$ is possible.

Given a component i, a global state $q = (q_1, \ldots, q_n)$ such that $q_j \in PS_j(q_i)$ for all $j \in need(q_i)$ might be a reachable deadlock state. This is clear, because all possible communication partners of i do not offer any action that can be used for cooperation by component i. If such a state is globally reachable, then its projection must be reachable consistently in all subsystems that consist of i and some $j \in need(q_i)$. This suggests the following strategy. For q_i and $j \in need(q_i)$ given, we determine the set of actions of i that can be used to enter a state (q_i, q_j) where q_j is problematic with respect to q_i. Then we compare these actions to the corresponding sets of actions when q_i is observed in $Sys\downarrow_{\{i,k\}}$ for the other components $k \in need(q_i)$. If the intersection of these sets of actions is empty we know that q_i can never be part of a reachable global state where all possible communication partners of q_i are in a state where they want to cooperate with i but are not able to do so. The idea of Corollary 2 on the following page is to ensure this for the local states of all components. The proposition we state requires a more general condition. By considering subsystems $Sys\downarrow_{\{i,j\}}$ with $j \in need(q_i)$ for a given q_i we are not able to confine the availability of those actions of i that are used for communication with components that are not in $need(q_i)$. Thus whenever we reach a state (q'_i, q_j) in $Sys\downarrow_{\{i,j\}}$ where q_i is reachable from q'_i in T_i without performing any port that is for communication with a component from $need(q_i)$ we must assume that (q_i, q_j) is also reachable. This leads to the following notion of backward-search.

Definition 9. Let Sys be a tree-like interaction system that has exclusive communication, and let $i \in K$ a component. For $q_i \in Q_i$ we define

$$\begin{aligned} BWS(q_i) := \{q'_i | \exists q'_i \xrightarrow{a^0_i} \ldots \xrightarrow{a^m_i} q_i \text{ in } T_i, \text{ where} \\ \forall l : a^l_i \notin \bigcup_{k \in need(q_i)} \mathit{comm}_i(k)\} \end{aligned}$$

the set of states of i from which q_i can be reached by only performing actions that belong to components that are not in $need(q_i)$.

For $Q'_i \subseteq Q_i$ we set $BWS(Q'_i) := \bigcup_{q_i \in Q'_i} BWS(q_i)$.

All banks are in $need(idle_{cc})$ and thus $BWS(idle_{cc}) = \{idle_{cc}\}$. In a similar way we obtain $BWS(PS_{b_i}(idle_{cc})) =$

$BWS\left(waiting_{b_i}\right) = \left\{waiting_{b_i}\right\}$. On the other hand for b_i we get $BWS\left(ok_{b_i}\right) = \left\{ok_{b_i}, waiting_{b_i}, checking_{b_i}\right\}$. For q_i and component $j \in need\left(q_i\right)$ as above we are now able to state precisely which actions of i are problematic with respect to q_i and j in the sense that in $Sys\downarrow_{\{i,j\}}$ they can lead to a state from $\{q_i\} \times PS_j\left(q_i\right)$.

Definition 10. Let Sys be a tree-like interaction system that has exclusive communication, $i \in K$ a component and $q_i \in Q_i$. For $j \in need\left(q_i\right)$ we define

$$\begin{aligned}\mathcal{PA}\left(q_i, j\right) := \{a_i \in \bigcup_{k\in need(q_i)} \mathcal{comm}_i\left(k\right) | \exists c \in C\left[\{i,j\}\right] \\ \text{such that } a_i \in c \text{ and } \exists\left(q_i', q_j'\right) \in BWS\left(q_i\right) \times \\ BWS\left(PS_j\left(q_i\right)\right), \exists\left(q_i'', q_j''\right) \text{ that is reachable} \\ \text{from } \left(q_i^0, q_j^0\right) \text{ such that } \left(q_i'', q_j''\right) \xrightarrow{c} \left(q_i', q_j'\right)\}\end{aligned}$$

the set of ports of i that can be used to enter $BWS\left(q_i\right) \times BWS\left(PS_j\left(q_i\right)\right)$ when i is observed in parallel with j.

Note that the previous definition uses the definition of BWS once in the context of i and once in the context of j. As motivated above we have to assume that it is possible to reach a state in $\{q_i\} \times PS_j\left(q_i\right)$ once $Sys\downarrow_{\{i,j\}}$ is in $BWS\left(q_i\right) \times BWS\left(PS_j\left(q_i\right)\right)$. Thus the comparison of the sets $\mathcal{PA}\left(q_i, j\right)$ indicates whether it may be possible to reach a deadlock-state that coincides with q_i in the entry for component i. In order to calculate $\mathcal{PA}\left(idle_{cc}, b_i\right)$ for some bank b_i we have to determine which actions of cc can be used in $Sys'_{bank}\downarrow_{\{cc,b_i\}}$ to enter a state in $BWS\left(idle_{cc}\right) \times BWS\left(PS_{b_i}\left(idle_{cc}\right)\right) = \left\{\left(idle_{cc}, waiting_{b_i}\right)\right\}$. Now the only two transitions of cc in Sys_{bank} entering $idle_{cc}$ are labeled $correct_{cc}$ and $\neg correct_{cc}$ respectively. Passing over to Sys'_{bank} the possible candidates for $\mathcal{PA}\left(idle_{cc}, b_i\right)$ are the ports $correct_{cc}^{b_k}$ and $\neg correct_{cc}^{b_k}$ where $1 \leq k \leq m$. For $k = i$ it can be seen that neither of these two ports can be used to enter $\left(idle_{cc}, waiting_{b_i}\right)$ because these ports can only be performed in the connectors $\left\{correct_{b_i}, correct_{cc}^{b_i}\right\}$ respectively $\left\{\neg correct_{b_i}, \neg correct_{cc}^{b_i}\right\}$. Looking at the local transition systems it is easy to see that these connectors will never lead to $\left(idle_{cc}, waiting_{b_i}\right)$. Finally we can formulate our criterion. Note that the tree-like architecture of the system that is investigated is incorporated in the proposition as follows. All conditions that have to be checked in the two requirements only involve the analysis of those $Sys\downarrow_{\{i,j\}}$ where $\{i,j\}$ is an edge of G. This is not surprising because the assumption about the tree-like architecture was made to ensure that deadlock-freedom can be deduced from only comparing pairs of neighboring components.

Proposition 1. *Let Sys be a tree-like interaction system that has exclusive communication. If the following two conditions hold then Sys is deadlock-free.*

1. $\forall i\ \forall q_i : q_i^0 \notin BWS\left(q_i\right) \vee \exists j \in need\left(q_i\right)$ *such that* $q_j^0 \notin BWS\left(PS_j\left(q_i\right)\right)$
2. $\forall c \in C\ \exists i \in K$ *such that* $i\left(c\right) \neq \emptyset$ *and* $\forall q_i \in Q_i$:
$$i\left(c\right) \notin \bigcap_{k\in need(q_i)} \mathcal{PA}\left(q_i, k\right)$$

The conditions can be checked in time polynomial in the size of the input.

Checking deadlock-freedom directly from the definition involves a global state space analysis. The size of the global state space is in $O\left(|T_{max}|^{|K|}\right)$ where T_{max} is the largest local transition system. In order to check the conditions of the proposition we only investigate $|K|-1$ subsystems $Sys\downarrow_{\{i,j\}}$ for $\{i,j\} \in E$ whose sizes are in $O\left(|T_{max}|^2\right)$. Note that the ideas and conditions used in the proposition and the preceding definitions can be transferred to other formal models where subsystems are cooperating via synchronous communication.

We state two of a series of corollaries that can be inferred from this result. The first can be used to show that Sys_{bank} is deadlock-free.

Corollary 2. *Let Sys be a tree-like interaction system that has exclusive communication. If the following two conditions hold then Sys is deadlock-free.*

1. $\forall i\ \forall q_i : q_i^0 \notin BWS\left(q_i\right) \vee \exists j \in need\left(q_i\right)$ *such that* $q_j^0 \notin BWS\left(PS_j\left(q_i\right)\right)$
2. $\forall i \in K\ \forall q_i \in Q_i : \bigcap_{k\in need(q_i)} \mathcal{PA}\left(q_i, k\right) = \emptyset$

We have already explained for all $1 \leq i \leq m$ that $correct_{cc}^{b_i}, \neg correct_{cc}^{b_i} \notin \mathcal{PA}\left(idle_{cc}, b_i\right)$. This means that we obtain $\bigcap_{b_k\in need(idle_{cc})} \mathcal{PA}\left(idle_{cc}, b_k\right) = \emptyset$. An analogous statement can be shown for the other states. The first condition is also true and deadlock-freedom of the bank system follows from the corollary. Corollary 2 can often be applied to systems where groups of components that compete for cooperation with $i \in K$ in the sense that i has a state where it has the choice between these components exhibit very similar or even the same behavior. Then the conditions are often satisfied. The intersections in the second condition are in fact empty because whenever we observe i in parallel with j the actions from $\mathcal{comm}_i\left(j\right)$ will not lead to a problematic state. If they did this could be interpreted as an indicator that the system has not been specified correctly. Sys_{bank} has the above property. All communication partners of cc (the banks) are essentially the same up to nomenclature and all communication partners of bank b_i but one - namely cc which never competes (in the sense described above) for cooperation with any of the ATMs - also behave in the same way.

Corollary 3 is very basic. It shows that under further assumptions checking global deadlock-freedom boils down to checking deadlock-freedom of the subsystems $Sys\downarrow_{\{i,j\}}$ for $\{i,j\} \in E$.

Corollary 3. *Let Sys be a tree-like interaction system that has exclusive communication and let* $|need\left(q_i\right)| = 1$ *for all* $i \in K$ *and for all* q_i. *If* $Sys\downarrow_{\{i,j\}}$ *is deadlock-free for all* $\{i,j\} \in E$ *then Sys is deadlock-free.*

If G is simply a star with center $i \in K$ then for all $j \in nb\left(i\right)$ and all q_j the condition of the corollary is satisfied, because i is the only possible partner for communication of q_j. For i the condition implies that it cannot choose between different communication partners in any of its states. It may offer different choices to this communication partner, though. If this is the case a deadlock can only occur if i reaches a state where it needs j while j does not offer the needed ports. This possibility can be excluded by looking at the subsystems only.

6. COMPARISON WITH OTHER COMPOSITIONALITY RESULTS AND DISCUSSION

Subsequently to the present paper we have developed an adaptation of the notion of tree-like interaction systems such that connectors of size greater than two can be treated as well [28]. The critical point was to make sure that it is still possible to reduce the check for deadlock-freedom to the analysis of subsystems of size two on the one hand and to avoid cyclic structures in the way the components communicate on the other hand. For this purpose we introduced graphs with two sorts of nodes, one of which representing the components and the other one representing the connectors. We presented a sufficient criterion for deadlock-freedom of such systems that only investigates subsystems of pairs of components.

Next we compare our criterion to the sufficient condition for deadlock-freedom from [24] mentioned above that can be applied to general interaction systems. Both conditions make use of an over-approximation of the projection of the set of globally reachable states to subsystems. In our case these subsystems are of size two whereas the subsystems investigated in [24] are of size smaller than or equal to three. Both approaches then check a condition on the computed over-approximation. The criterion from [24] fails whenever the investigated subsystem contains a reachable state such that in this state component i waits for component j and component j waits in turn for component k. It is easy to see that this kind of situation may arise in Sys_{bank}. This shows that there are tree-like systems that can be shown to be deadlock-free using our criterion whereas the criterion from [24] fails. Having said that, we are also interested in the opposite direction. Therefore we examine a tree-like interaction system for which the criterion from [24] yields deadlock-freedom. As stated above this means that for no subsystem a state was reached where component i waits for j and j waits for k. Since our criterion only checks pairs of neighboring components we are only interested in those subsystems where $i = k$ and i and j are neighboring components in G. But for these subsystems the above statement simply means, that all sets of problematic states are empty. Then it is clear that our criterion also yields deadlock-freedom. This means, that our criterion is more powerful than the one from [24] as far as tree-like interaction systems as considered in this paper are concerned.

We also mention [7] which investigated deadlock-freedom in a setting that assumes architectural constraints. Note that [7] only considers systems composed of three components at most.

At last we compare our approach to another one that also deals with tree-like architectures. In [8] a process algebra based abstract description language, called PADL, for concurrent systems is developed, and a criterion for deadlock-freedom of tree-like systems is presented. This condition basically ensures that no component is confined by any cooperation with one of its neighbors in the tree. A simple example shows that our criterion can be applied to cases where the test from [8] fails to prove deadlock-freedom: The system we consider consists of three components. We have $K = \{1, 2, 3\}$. The T_i are given in Fig. 2. We define $C = \{\{a_1, a_2\}, \{b_1, b_2\}, \{a_1, a_3\}, \{b_1, b_3\}\}$. The system is tree-like. Passing over to Sys', Corollary 2 shows that the

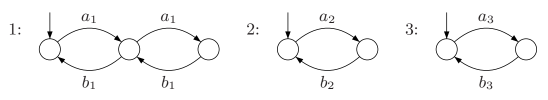

Figure 2: 1 is restricted by 2 respectively 3

system is deadlock-free. Nonetheless T_1 is constrained by the cooperation with 2 or 3, i.e., $Sys' \downarrow_{\{1,i\}}$ restricted to $\mathcal{A}_1$ is not weakly bisimular to T_1' for $i = 2, 3$ (here T_1' denotes the adapted transition system constructed in the proof of Lemma 1). This means that the condition for deadlock-freedom from [8] is violated. On the other hand it is possible to show that every deterministic tree-like system that can be proven to be deadlock-free using the condition from [8] can also be treated in a slightly more general version of our Proposition 1.

Having spoken about the applicability of our approach there are of course deadlock-free interaction systems that cannot be shown to be deadlock-free using Proposition 1. This is not surprising though, because the complexity results mentioned in Sect. 1 show that we cannot expect to find a characterization of deadlock-freedom that can be checked efficiently for all systems. We shortly want to take a closer look at the limitations of our approach. First of all it is clear that there is a large class of interaction systems that our criterion is not even capable of dealing with because we confine the interaction systems we investigate by demanding that they be tree-like. But even in the class of tree-like interaction systems there are deadlock-free systems where our criterion fails. There are two main reasons for this phenomenon. First of all for every edge $\{i, j\}$ of G we compute an over-approximation of the projection of the reachable global states to $Sys \downarrow_{\{i,j\}}$. This means that we may find pairs of states which are problematic with respect to each other even though these states can never be globally reached in this combination. Note that keeping track of the actions that lead to problematic states, as we do, helps to make the over-approximation more precise. Secondly, the sufficient condition we check makes sure that no component can ever reach a state where all possible communication partners do not offer the required actions. Of course there can be a global state q, where the system is not in a deadlock, even though there is a set K' of components for which the possible communication partners block each other. This is because other components that are independent (in q) from the components in K' may be able to proceed. Note that it is not possible to distinguish this kind of state from a real deadlock-state when we only look at subsystems, because we obviously loose the information by projecting Sys to a subset of components.

7. CONCLUSION AND FUTURE WORK

We presented a criterion that ensures deadlock-freedom for the important class of tree-like component-systems, i.e., systems where the communication structure forms a tree. The criterion exploits compositionality in the sense that a condition is *locally* checked on pairs of neighboring components. If the condition is satisfied we can derive the *global* property of deadlock-freedom. The criterion can be tested efficiently. Since the violation of a safety-property can be modeled as a deadlock our criterion can also help to analyze other safety-properties apart from deadlock-freedom.

As a simple application we discussed a system consisting of banks, ATMs, and a clearing company. By means of the example, we explained that the strength of the criterion can be pinpointed for systems where whenever several components (e.g., the ATMs) compete for interaction with one component (the corresponding bank) then they exhibit analogous behavior. In such a setting Corollary 2 is often applicable. We shortly discussed other approaches that exploit compositionality. Further work considering other classes of architectures is in progress. In particular we generalized the approach in order to be able to admit connectors of size greater than two, as well [28]. Thus we want to be able to exhibit a range of design patterns that can be followed to obtain component-systems that are correct by construction.

We want to address attention to the fact that the reachability analysis we use in order to over-approximate the globally reachable state-space not only compares the reachable states of different systems. It also compares the actions that can be used to enter problematic states to ensure that states that indicate a possible global deadlock are only interpreted as problematic if the actions they can be entered through are consistent.

Interaction systems can easily be extended to account for variables and value-passing. Also, it is an interesting question to consider systems with protocols. Here a component i that may communicate with other components does so by providing a communication protocol cp_i^j for every partner j (who does so analogously). The protocols must be "compatible" with the behavior of i in a sense to be made precise. The question is under what restrictions it is sufficient to analyze pairs of protocols in order to derive information on the global system.

8. REFERENCES

[1] R. Allen and D. Garlan. A Formal Basis for Architectural Connection. *ACM Trans. Softw. Eng. Methodol.*, 6(3):213–249, 1997.

[2] F. Arbab. Abstract Behavior Types: A Foundation Model for Components and Their Composition. In *Proceedings of FMCO'02*, volume 2852 of *LNCS*, pages 339–360, 2003.

[3] P. Attie and H. Chockler. Efficiently Verifiable Conditions for Deadlock-Freedom of Large Concurrent Programs. In *Proceedings of VMCAI'05*, volume 3385 of *LNCS*, pages 465–481, 2005.

[4] E. Badouel, A. Benveniste, M. Bozga, B. Caillaud, O.Constant, B. Josko, Q. Ma, R. Passerone, and M. Skipper. SPEEDS Metamodel Syntax and Draft Semantics, January 2007. Deliverable D2.1c.

[5] R. Bastide and E. Barboni. Software Components: A Formal Semantics Based on Coloured Petri Nets. In *Proceedings of FACS'05*, ENTCS, 2005.

[6] A. Basu, M. Bozga, and J. Sifakis. Modeling Heterogeneous Real-Time Components in BIP. In *Proceedings of SEFM'06*, pages 3–12. IEEE Computer Society, 2006.

[7] H. Baumeister, F. Hacklinger, R. Hennicker, A. Knapp, and M. Wirsing. A Component Model for Architectural Programming. In *Proceedings of FACS'05*, volume 160 of *ENTCS*, pages 75–96. Elsevier, 2006.

[8] M. Bernardo, P. Ciancarini, and L. Donatiello. Architecting Families of Software Systems with Process Algebras. *ACM Trans. on Software Engineering and Methodology*, 11:386 – 426, October 2002.

[9] M. Bozga, O.Constant, B. Josko, Q. Ma, and M. Skipper. SPEEDS Metamodel Syntax and Static Semantics, February 2007. Deliverable D2.1b.

[10] M. Broy. Towards a Logical Basis of Software Engineering. In M. Broy and R. Steinbrüggen, editors, *Calculational System Design, IOS 1999*, volume 158 of *NATO ASI Series, Series F: Computer and System Sciences*, pages 101 – 131. 1999.

[11] S. Chouali, M. Heisel, and J. Souquières. Proving Component Interoperability with B Refinement. In *Proceedings of FACS 05*, volume 160 of *ENTCS*, pages 67–84, 2006.

[12] L. de Alfaro and T. A. Henzinger. Interface Automata. In *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering (FSE 01)*, pages 109–120, 2001.

[13] P. Godefroid and P. Wolper. Using Partial Orders for the Efficient Verification of Deadlock Freedom and Safety Properties. *Form. Methods Syst. Des.*, 2(2):149–164, 1993.

[14] G. Gössler. Component-based Design of Heterogeneous Reactive Systems in Prometheus. Technical report 6057, INRIA, December 2006.

[15] G. Gössler, S. Graf, M. Majster-Cederbaum, M. Martens, and J. Sifakis. An Approach to Modelling and Verification of Component Based Systems. In *Proceedings of SOFSEM 07*, volume 4362 of *LNCS*, pages 295–308, 2007.

[16] G. Gössler, S. Graf, M. Majster-Cederbaum, M. Martens, and J. Sifakis. Ensuring Properties of Interaction Systems. In *Program Analysis and Computation, Theory and Practice*, volume 4444 of *LNCS*, pages 201–224, 2007.

[17] G. Gössler and J. Sifakis. Component-Based Construction of Deadlock-Free Systems. In *Proceedings of FSTTCS'03*, volume 2914 of *LNCS*, pages 420–433, 2003.

[18] G. Gössler and J. Sifakis. Composition for Component-Based Modeling. In *Proceedings of FMCO 02*, volume 2852 of *LNCS*, pages 443–466, 2003.

[19] G. Gössler and J. Sifakis. Priority Systems. In *Proceedings of FMCO'03)*, volume 3188 of *LNCS*, pages 314–329, 2004.

[20] G. Gössler and J. Sifakis. Composition for component-based modeling. *Sci. Comput. Program.*, 55(1-3):161–183, 2005.

[21] P. Inverardi and S. Uchitel. Proving Deadlock Freedom in Component-Based Programming. In *Proceedings of FASE'01*, volume 2029 of *LNCS*, pages 60–75, 2001.

[22] N. A. Lynch and M. R. Tuttle. An Introduction to Input/Output Automata. *CWI-Quarterly*, 2(3):219–246, Sept. 1989.

[23] M. Majster-Cederbaum and M. Martens. Robustness in Interaction Systems. In *Proceedings of FORTE'07*, volume 4574 of *LNCS*, pages 325–340, 2007.

[24] M. Majster-Cederbaum, M. Martens, and C. Minnameier. A Polynomial-Time-Checkable

Sufficient Condition for Deadlock-freeness of Component Based Systems. In *Proceedings of SOFSEM 07*, volume 4362 of *LNCS*, pages 888–899, 2007.

[25] M. Majster-Cederbaum, M. Martens, and C. Minnameier. Liveness in Interaction Systems. In *Proceedings of FACS'07*, ENTCS, 2007.

[26] M. Majster-Cederbaum and C. Minnameier. Deriving Complexity Results for Interaction Systems from 1-safe Petri Nets. In *Proceedings of SOFSEM'08*, volume 4910 of *LNCS*, pages 352–363. Springer, 2008.

[27] M. Majster-Cederbaum and C. Minnameier. Everything is PSPACE-complete in Interaction Systems, 2008. Accepted for Publication at ICTAC'08.

[28] M. Martens and M. Majster-Cederbaum. Deadlock-Freedom for Acyclic Component Architectures with Multiway Cooperation, 2008. submitted for publication.

[29] S. Moschoyiannis and M. Shields. Component-Based Design: Towards Guided Composition. In *Proceedings ACSD'03*, pages 122–131. IEEE Computer Society, 2003.

[30] O. Nierstrasz and F. Achermann. A Calculus for Modeling Software Components. In *Proceedings of FMCO'02*, volume 2852 of *LNCS*, pages 339–360, 2003.

[31] P. Parizek and F. Plasil. Modeling Environment for Component Model Checking from Hierarchical Architecture. In *Proceedings of FACS'06*, volume 182 of *ENTCS*, pages 139–153. Elsevier, 2007.

[32] J. Sifakis. Modeling Real-time Systems. In *Proceedings of RTSS'04*, pages 5–6. IEEE Computer Society, 2004.

[33] J. Sifakis. A Framework for Component-Based Construction. In *Proceedings of SEFM'05*, pages 293 – 300. IEEE Computer Society, 2005.