Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 215 (2008) 57–74

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Liveness in Interaction Systems

Mila Majster-Cederbaum,[1]
Moritz Martens[2] and Christoph Minnameier[3]

Institute of Computer Science
University of Mannheim
Mannheim, Germany

Abstract

Interaction systems were proposed and implemented by Sifakis et al. as a model for the design and study of component based systems. We investigate here the property of liveness in interaction systems where liveness of an action, a component or a set of components means that the action (component, set of components) will repeatedly participate in every run of the global system. We show that deciding liveness is NP-hard. Then we present a characterization of liveness. Finally, by exploiting local information, we establish a polynomial-time criterion that guarantees liveness. We combine the criterion with the characterization to obtain a test for liveness.

Keywords: Component-based Modeling, Interaction Systems, Liveness

1 Introduction

In the last decade a variety of formal approaches to the specification and analysis of component based systems at different levels and with different specific objectives have been proposed [1,2,9,33,3,34,38,35,7,21,5,23,10,16,15].

We investigate here the approach of *interaction systems* that was proposed and discussed by Sifakis et al. in [16,17,15,40,41,39] and in more detail in [18]. The model clearly separates the issues of 1. interfaces, 2. behavior of the components and 3. interaction between components. It has been implemented

1 Email: mcb@informatik.uni-mannheim.de
2 Email: mmartens@informatik.uni-mannheim.de
3 Email: cmm@informatik.uni-mannheim.de

doi:10.1016/j.entcs.2008.06.021

successfully in the Prometheus tool [13] and the BIP system [4]. BIP has been extended to a framework for hierarchical components [19] and enriched with contracts as defined in the SPEEDS project [6]. The model was used to specify various applications, e.g. [4,13,14,16,40,27,31,29]. In this approach a component offers a certain set of *ports*. The communication behavior of a component is given by a labeled transition system where the labels are taken from the port set. That is, the transition system restricts the order of calls to the ports. Components are put together via *connectors* where a connector is a finite nonempty set of ports such that no two ports stem from the same component. The transitions which the induced global system can perform are regulated by these connectors. One port can be contained in various connectors of various sizes such that the cooperation between components can be regulated freely in a simple way. In approaches using process algebra the basic cooperation scheme is usually fixed and binary and it is cumbersome to realize more flexibility. I/O-automata [23] can be considered as a subclass of interaction systems. They also use some kind of transition systems to model components but have a more restrictive scheme of cooperation. This is also true for the interface-automata defined in [10].

Different aspects of component-based systems have been studied, for example compatibility [37], deadlock-freedom [21,5,1], reliability prediction [22,36,38].

In the framework of interaction systems properties such as local/global deadlock-freedom, progress of a component, availability of ports and robustness against failure of components have been discussed [18,14,24,25]. In general it is difficult to test these properties as they involve the exploration of the global state space. Indeed, it was shown that deciding deadlock-freedom in interaction systems is NP-hard [32]. Recently this result was strengthened by showing that deciding local and global deadlock-freedom is PSPACE-hard [26]. One way to proceed in such situations is to establish criteria that ensure desired properties and can be tested more easily. In [25] for example we presented a condition that ensures deadlock-freedom and can be tested in polynomial time. In [24] we investigated criteria for robustness.

In this work we concentrate on liveness in interaction systems. A component is considered to be live if, no matter how the global system evolves and independently of the point of time we consider, it will repeatedly participate in some step of the system. We show first that deciding liveness of a set of components is NP-hard. Then we give a characterization of liveness of a set of components in an interaction system. Moreover, we establish a criterion that entails liveness and can be tested in polynomial time. We present a hybrid algorithm for testing liveness that combines the characterization and the criterion mentioned above.

The paper is structured as follows. Section 2 summarizes the basic definitions. Section 3 contains the notions of deadlock-freedom and liveness. Liveness is investigated in Sections 4, 5 and 6 which are concerned with NP-hardness respectively present the characterization of, the criterion for liveness, and the hybrid algorithm.

2 Components, Connectors and Interaction Systems

We build on a model for component-based systems, called *interaction systems*, that was proposed in [16,17,15,40,41]. We start with a set K of components where we usually refer to a component as $i \in K$. For every component $i \in K$ a set A_i of ports (or actions) is specified which are used for cooperation with other components. The cooperation is determined by so-called connectors. A connector is a finite nonempty set of ports that contains at most one port for every component in K. Any nonempty subset of a connector constitutes an interaction of the system. A port may be part of various connectors which may be of different size. Hence the cooperation can be regulated in a very flexible way. An interaction describes a step of the system where the ports contained in that interaction are performed together.

Definition 2.1 A *component system* is a pair $CS = \left(K, \{A_i\}_{i\in K}\right)$ where K is the set of *components*, A_i is the *port set* of component i, and any two port sets are disjoint. Ports are also referred to as *actions.*

The union $A = \bigcup_{i\in K} A_i$ of all port sets is the *port set* of K. A finite nonempty subset c of A is called a *connector* for CS, if it contains at most one port of each component $i \in K$, that is $|c \cap A_i| \leq 1$ for all $i \in K$. A *connector set* is a set C of connectors for CS that covers all ports and contains only maximal elements:

$$1.\ \bigcup_{c\in C} c = A \qquad 2.\ c \subseteq c' \Rightarrow c = c' \text{ for all } c, c' \in C.$$

If c is a connector, $I(c)$ denotes the set of all nonempty subsets of c and is called the set of *interactions of* c. For a set C of connectors $I(C) = \bigcup\limits_{c\in C} I(c)$ is the set of *interactions of* C. We also call connectors $c \in C$ the *maximal* interactions. For component i and interaction α, we put $i(\alpha) = A_i \cap \alpha$. We say that component i *participates* in α, if $i(\alpha) \neq \emptyset$.

We give a simple example to illustrate these concepts. We will extend this example throughout the text whenever we encounter new notions.

Example 2.2 We consider a component system $CS_5 = (K_5, \{A_i\}_{i \in K_5})$ consisting of five components, where $K_5 := \{1, 2, 3, 4, 5\}$ and the port sets of the components are given by $A_1 := \{a_1\}$, $A_2 := \{b_1, b_2\}$, $A_3 := \{d_1, d_2\}$, $A_4 := \{e_1, e_2\}$, and $A_5 := \{f_1, f_2\}$. Additionally we fix the connector set $C_5 := \{\{a_1, b_1\}, \{a_1, e_1\}, \{d_1, f_1\}, \{a_1, e_2, f_2\}, \{b_2, e_2, f_2\}, \{b_2, d_2, e_2\}, \{a_1, d_2, e_2\}\}$. This component system is illustrated in Fig. 1 where the components are shown as boxes and the ports of the components are the black boxes. The connectors are represented by lines connecting the respective ports. Here

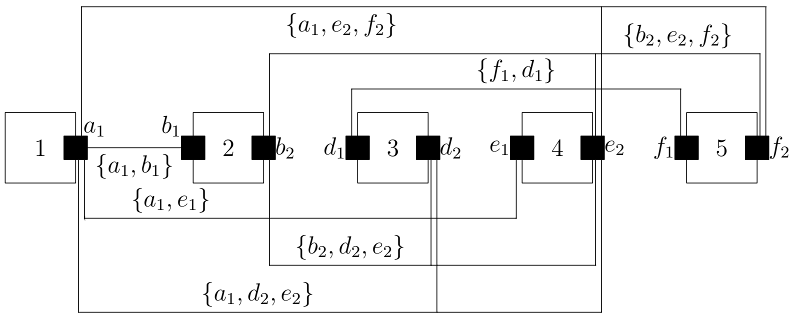

Fig. 1. The connectors for Example 2.2

for example components 1 and 2 may perform their respective first actions together whereas 2, 4, and 5 may perform their respective second actions together.

In the following, we always assume that $K = \{1, \ldots, n\}$ for some $n \in \mathbb{N}$ or that K is countably infinite.

An *interaction model* for a component system CS is defined by a connector set C together with a set $Comp$ of interactions that are declared to be *complete*. If an interaction is declared complete it can be performed independently of the environment. Note that it is a design decision which interactions are chosen to be complete. This choice is not restricted in any way and only depends on the system one wishes to model.

Definition 2.3 Let C be a connector set for the component system CS and let $Comp \subseteq I(C)$ be a subset of interactions. The pair $IM := (C, Comp)$ is an *interaction model* for CS. The elements of $Comp$ are called *complete interactions*.

Example 2.2 continued Let $IM_5 := (C_5, \emptyset)$ be an interaction model for

CS_5, i.e. only interactions in C_5 are independent of the environment.

If for some reason the interactions $\{e_2\}$ and $\{f_2, b_2\}$ for example should be independent of other actions, we could set $Comp := \{\{e_2\}, \{f_2, b_2\}\}$.

The notions presented so far are only concerned with the possible structure of communication between the different components. In a further level of description of the components the order in which a component may perform the actions it provides is restricted. For that purpose for every component $i \in K$ a labeled transition system T_i describing the behavior of i with respect to interaction is introduced. In the simplest case T_i is the "union" of transition systems T_{ij} where T_{ij} is the protocol regulating the cooperation of component i with component j.

Definition 2.4 Let $CS = \big(K, \{A_i\}_{i\in K}\big)$ be a component system and $IM = (C, Comp)$ an interaction model for CS. Let for each component $i \in K$ a transition system $T_i = (Q_i, A_i, \to_i, Q_i^0)$ be given where $\to_i \subseteq Q_i \times A_i \times Q_i$ and $Q_i^0 \subseteq Q_i$ is a non-empty set of *initial states*. We write $q_i \xrightarrow{a_i}_i q_i'$ instead of $(q_i, a_i, q_i') \in \to_i$.

The *induced interaction system* is given by $Sys := \big(CS, IM, \{T_i\}_{i\in K}\big)$ where the *global behavior* $T = (Q, C \cup Comp, \to, Q^0)$ is obtained from the local transition systems of the individual components in a straightforward manner:

(i) $Q := \prod_{i\in K} Q_i$, the Cartesian product of the Q_i which we consider to be order independent. We denote states by tuples $q := (q_1, \ldots, q_j, \ldots)$ and call them *(global) states*.

(ii) $Q^0 := \prod_{i\in K} Q_i^0$, the Cartesian product of the local initial states. We call the elements of Q^0 (global) *initial states*.

(iii) $\to \subseteq Q \times (C \cup Comp) \times Q$, the transition relation for Sys defined by

$$\forall \alpha \in (C \cup Comp)\ \forall q, q' \in Q : q = (q_1, \ldots, q_j, \ldots) \xrightarrow{\alpha} q' = \big(q_1', \ldots, q_j', \ldots\big) \Leftrightarrow$$

$$\forall i \in K : q_i \xrightarrow{i(\alpha)}_i q_i' \text{ if } i \text{ participates in } \alpha \text{ and } q_i' = q_i \text{ otherwise.}$$

The global system can perform either complete or maximal interactions α where α may be performed in a global state q if all partners that are involved in α are offering their corresponding action.

Without loss of generality we always assume that every local state of every component is reachable from some initial state in the local transition system.

Example 2.2 continued The behavior of component i is given in Fig. 2

for $i \in \{1, \ldots, 5\}$. For every component i we put $Q_i^0 = Q_i$. The induced global transition system is called $T(5)$. For example in the global state

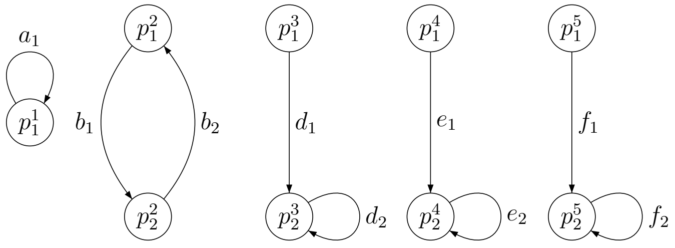

Fig. 2. The local behavior of the components of Example 2.2

$(p_1^1, p_1^2, p_1^3, p_1^4, p_1^5)$ a transition labeled with $\{d_1, f_1\}$ is enabled. Our example system $Sys_5 := (CS_5, IM_5, \{T_i\}_{i \in K_5})$ is now completely specified.

Remark 2.5 In what follows, we often mention $Sys = \big(CS, IM, \{T_i\}_{i\in K}\big)$. It is understood that $CS = \big(K, \{A_i\}_{i\in K}\big)$, $IM = (C, Comp)$, $T_i = (Q_i, A_i, \rightarrow_i, Q_i^0)$ for $i \in K$, and T are given as above. Usually we will display the local transition systems graphically. If not explicitly stated otherwise the local initial states will be marked by an ingoing arrow which we will omit for component i whenever $Q_i^0 = Q_i$.

3 Liveness in Interaction Systems

Remark 3.1 From now on for all $i \in K$ we will assume that T_i has the property that every state offers at least one action.

In order to define liveness we first need a notion of (global) deadlock-freedom. Note that in [25] we also investigated a notion of local deadlock of a subset $K' \subseteq K$ of components which describes a situation where in the current state every component in K' needs the cooperation of at least one other component in K' which in turn does not offer the needed ports.

Definition 3.2 Let Sys be an interaction system.

(i) Let $q \in Q$. q is *reachable in Sys* if there is a sequence $q^0 \xrightarrow{\alpha_0} q^1 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{n-1}} q$ such that $q^0 \in Q^0$ and $\alpha_i \in C \cup Comp$ for all $0 \leq i \leq n-1$.

(ii) Sys is called *deadlock-free* if for every reachable state q there exists $\alpha \in C \cup Comp$ and $q' \in Q$ such that $q \xrightarrow{\alpha} q'$.

This definition is justified by the fact that maximal as well as complete interactions are independent of the environment. They do not have to wait for any other components and can be performed immediately. Started in a global initial state a deadlock-free system may always proceed with some maximal or complete interaction. Thus cyclic waiting involving all components cannot occur, whereas two or more components may be engaged in a local deadlock.

Example 2.2 continued It is easy to see that Sys_5 is deadlock-free. We have $Q_i^0 = Q_i$ for all components which means that every global state is reachable. Therefore one has to show that every global state offers at least one maximal interaction which boils down to a simple case distinction.

In deadlock-free systems runs always exist, where a run is simply an infinite thread of execution starting in a reachable state of the system.

Definition 3.3 Let Sys be a deadlock-free interaction system and let $q \in Q$ be a reachable state. A *run* of Sys is an infinite sequence $\sigma : q = q^0 \xrightarrow{\alpha_0} q^1 \xrightarrow{\alpha_1} q^2 \dots$ with $q^l \in Q$ and $\alpha_l \in C \cup Comp$ for all $l \in \mathbb{N}$.

Let $i \in K$ be a component and let σ be a run of Sys. If there exists l such that i participates in α_l we say that *i participates in σ*.

Now we can define when a set of components $K' \subseteq K$ is live. Basically this is the case if for any point of time no matter how the system behaves some component in K' will eventually participate in some interaction. From now on we identify singleton sets with their element if it is convenient to do so.

Definition 3.4 Let Sys be a deadlock-free interaction system and let $K' \subseteq K$ be a nonempty set of components. We say that K' is *live* in Sys if for every run σ of Sys there exists some $i \in K'$ such that i participates in σ.

Remark 3.5 It should be noted that this notion of liveness applied to a component i is the same as requesting that this component should participate infinitely often in a run because runs may start in any reachable state.

This notion of liveness is different from the one introduced for Petri nets [8] that corresponds to our notion of local progress of a component i in interaction systems [14], which means that at any point in any run we may proceed in such a way that component i will participate. Clearly liveness of i implies local progress but not vice versa. A general form of liveness-properties for a component system as a whole is defined in [7]. The questions referring to single components that we are interested in cannot be directly formulated and investigated in [7] because the identity of a component may be lost in the system which means that it is not meaningful to consider liveness of a component.

If i is live in Sys any set of components containing i is also live. The converse does not hold: even if K' is live in Sys there does not need to be any $i \in K'$ that is live. Also note that for $K' = K$ liveness follows from deadlock-freedom.

4 Deciding Liveness is NP-Hard

We will show that deciding liveness in interaction systems is NP-hard by reducing the question whether a formula F in 3-KNF [11] is not satisfiable to the question of deciding whether a certain component κ is live in a certain deadlock-free system. Note that the reduction technique only has to be slightly adapted in the case of I/O-automata [23] yielding an analogous result for this formalism.

The idea of the reduction is as follows. Each clause of F will be represented by one component as will be each literal of every clause. Other than that there is one component κ that is live if and only if F is not satisfiable. The clause components only have one state other than the starting state. This state represents an evaluation of the clause to *true*. The literal-components have a starting state from which two states representing the evaluation of the literal to *true* respectively $false$. The choice of the connector set will then make sure that all literals can only be set consistently. That means if one variable is set to a certain value all literals with the same variable must be set appropriately. Then it is clear that a global state where for each clause there is one literal-component that is in its state representing *true* can be reached if and only if F is satisfiable. This means all clause-components can move to their respective *true* state if and only if F is satisfiable. Then the choice of the connectors will ensure that it is only possible to start a run not involving κ in such a state.

Let $F = k_1 \wedge \ldots \wedge k_n$ with $k_i = \left(l_{(i,1)} \vee l_{(i,2)} \vee l_{(i,3)}\right)$ be a propositional formula in 3-KNF, where $l_{(i,1)}$, $l_{(i,2)}$, and $l_{(i,3)}$ are literals. For l a literal let $var\,(l)$ denote the variable occurring in l. Let $var\,(F) := \left\{var\left(l_{(i,j)}\right) | 1 \leq i \leq n, 1 \leq j \leq 3\right\}$ denote the set of variables occurring in F.

We construct a deadlock-free interaction system $Sys(F)$ with component-set $K\,(F)$ containing a component κ such that

$$(F \notin \text{3-SAT}) \Leftrightarrow (\kappa \text{ is live in } Sys(F))$$

where 3-SAT is the set of satisfiable formulas in 3-KNF. Besides the component κ we represent each clause k_i by a component $(i, 0)$ and each literal $l_{(i,j)}$ by a component (i, j). We define

$$Sys\,(F) := \left(CS\,(F)\,, IM\,(F)\,, \left\{T_{(i,j)}\right\}_{j=0,\ldots,3}^{i=1,\ldots,n} \cup \{T_\kappa\}\right)$$

where the components and their port sets are given by:

$$\begin{aligned} K(F) &:= \{(i,j) \mid 1 \le i \le n, 0 \le j \le 3\} \cup \{\kappa\} \\ A_{(i,0)} &:= \{true_i, SAT_i\} \text{ for } 1 \le i \le n \\ A_{(i,j)} &:= \left\{set1_{(i,j)}, set0_{(i,j)}, true_{(i,j)}, a_{(i,j)}\right\} \text{ for } 1 \le i \le n, j \ne 0 \\ A_\kappa &:= \{a_\kappa\} \end{aligned}$$

We define the following connectors. First we have:

$$sat := \{SAT_1, \ldots, SAT_n\}$$

Next we define:

$$\begin{aligned} set1_x &:= \left\{a_\kappa, set1_{(i_1,j_1)}, \ldots, set1_{(i_m,j_m)}\right\} \text{ and} \\ set0_x &:= \left\{a_\kappa, set0_{(i_1,j_1)}, \ldots, set0_{(i_m,j_m)}\right\} \end{aligned}$$

where $x = var\left(l_{(i_1,j_1)}\right) = \ldots = var\left(l_{(i_m,j_m)}\right)$ and there is no other literal l with $x = var(l)$. Other than that we set:

$$\begin{aligned} t_{(i,j)} &:= \left\{a_\kappa, true_{(i,j)}, true_i\right\} \\ c_a &:= \{a_\kappa\} \cup \left\{a_{(i,j)} | 1 \le i \le n, j \ne 0\right\} \end{aligned}$$

We set

$$C := \{sat\} \cup \{c_a\} \cup \bigcup_{x \in var(F)} \{set1_x, set0_x\} \cup \bigcup_{1 \le i \le n, j \ne 0} \left\{t_{(i,j)}\right\}$$

and choose $Comp$ to be the empty set.

The local transition system for κ is given in Fig. 3 (a). $T_{(i,0)}$ is given in Fig. 3 (b) and $T_{(i,j)}$ for $j \ne 0$ and $l_{(i,j)}$ a positive (resp. negative) literal is given in Fig. 3 (c) (resp. (d)).

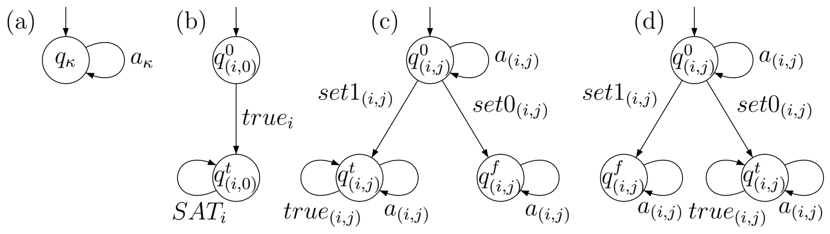

Fig. 3. The local transition systems for the components of $Sys(F)$

Theorem 4.1 *Let $Sys(F)$ be defined as above.*

(i) *Going from F to $Sys\,(F)$ there is no exponential blow-up in notation.*

(ii) *$Sys\,(F)$ is deadlock-free.*

(iii) *F is not satisfiable if and only if κ is live in $Sys\,(F)$.*

The proof can be found in the technical report [28]. The following example illustrates the idea behind the reduction of the above theorem. For a satisfiable formula F it will be shown how a run in $Sys\,(F)$ can be found in which κ only participates finitely many often. Let $F = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$. F is satisfiable, namely $v(F) = 1$ for $v(x_1) = 1, v(x_2) = 1, v(x_3) = 0$.

Consider $K\,(F) = \{(1,0),(1,1),(1,2),(1,3),(2,0),\ldots,(3,3),\kappa\}$ and $Sys\,(F) := \left(CS\,(F)\,, IM\,(F)\,, \{T_{(i,j)}\}_{j=0,\ldots,3}^{i=1,\ldots,3} \cup \{T_\kappa\}\right)$ as above. The evaluation v given above yields the following path starting in q^0 ending in a state where the connector sat can repeatedly be applied resulting in a run as above.

$$\sigma := q^0 \overset{set1_{x_1}}{\longrightarrow} q^1 \overset{set1_{x_2}}{\longrightarrow} q^2 \overset{set0_{x_3}}{\longrightarrow} q^4 \overset{t_{(1,1)}}{\longrightarrow} q^5 \overset{t_{(2,2)}}{\longrightarrow} q^5 \overset{t_{(3,3)}}{\longrightarrow} q^6 \overset{sat}{\longrightarrow} q^6 \overset{sat}{\longrightarrow} \ldots$$

In the first step component $(1,1)$ moves to its *true*-state and $(2,1)$ and $(3,1)$ move to their respective *false*-state representing the evaluation of x_1 to 1. Analogously the other six literal-components change their state according to $set1_{x_2}$ and $set0_{x_3}$. In steps four to six components $(1,0)$, $(2,0)$, and $(3,0)$ move to their *true*-state together with $(1,1)$, $(2,2)$, respectively $(3,3)$. Note that in the fifth step $t_{(2,3)}$ could also have been performed because $q^5_{(2,3)} = q^t_{(2,3)}$ as well. Then sat is enabled in q^6.

5 Characterizing Liveness of a Set of Components

In this section we consider a (not necessarily finite) deadlock-free interaction system. We present a characterization of all subsets $K' \subseteq K$ that are live. The benefit of this characterization is that it can be used in combination with the sufficient criterion for liveness that we will present in Sect. 6 for cases where the criterion alone does not imply liveness.

Definition 5.1 Let $K' \subseteq K$ be a subset of components. Let $excl\,(K') := \{\alpha \in C \cup Comp | \forall i \in K' : i\,(\alpha) = \emptyset\}$ denote the set of maximal or complete interactions in which no component from K' participates.

Definition 5.2 Let Sys be a deadlock-free interaction system and let $K' \subseteq K$ be a non-empty subset of components. We define

$$\bar{K}' := \{k \in K | \exists \alpha \in excl\,(K') : k\,(\alpha) \neq \emptyset\}\,.$$

Further we define the following labeled transition system

$$\bar{T} := \left(\bar{Q}, excl\,(K'), \rightarrow\right)$$

where $\bar{Q} := \prod_{k \in \bar{K}'} Q_k$ and $\rightarrow \subseteq \bar{Q} \times excl\,(K') \times \bar{Q}$ is the transition relation which is defined as follows: for any two states $\bar{p}, \bar{q} \in \bar{Q}$ and any interaction $\alpha \in excl\,(K')$ $\bar{p} \xrightarrow{\alpha} \bar{q} \Leftrightarrow \left(\forall i \in \bar{K}'\ \bar{p}_i \overset{i(\alpha)}{\rightarrow} \bar{q}_i \text{ if } i(\alpha) \neq \emptyset \text{ and } \bar{p}_i = \bar{q}_i \text{ otherwise}\right)$.

$\bar{K}'$ contains those components that participate in at least one maximal or complete interaction not involving any component from K'. We clearly have $K' \subseteq K \backslash \bar{K}'$. $\bar{Q}$ can be understood as the projection of Q to $\bar{K}'$ where we only allow those transitions labeled with some $\alpha \in excl\,(K')$.

Theorem 5.3 *Let Sys be deadlock-free and let $K' \subseteq K$. K' is live in Sys if and only if $\bar{T}$ does not contain any infinite path starting in a state $\bar{q}$ for which there exists $q' \in \prod_{i \in K \backslash \bar{K}'} Q_i$ such that $(\bar{q}, q')$ is reachable in Sys.*

For finite systems the characterization amounts to cycle detection and involves (partial) global state space analysis in the worst case and its usefulness to detect liveness depends on the size of K'. If K' contains very few elements usually it will not be helpful to analyze $\bar{T}$, because its number of states may still be exponential in the number of components. But even in this case the characterization can be helpful when the set $excl\,(K')$ is small and $\bar{T}$ sparse. In the extreme case every maximal or complete interaction involves some component from K' and $excl(K')$ is empty. Then it is clear anyway that K' is live.

6 Testing Liveness

In this section we present a hybrid algorithm that tests liveness of a subset K' of components. The algorithm is based on a sufficient condition and the characterization given in the previous section applied to a subsystem.

The condition that has to be checked comes down to a reachability analysis in a graph where the components are the nodes. The graph is constructed by checking certain dependencies between pairs of components that can be checked by investigating the *local* transition systems only. Therefore the graph can be constructed in time polynomial in the number of components and the size of the local transition systems such that the criterion avoids the investigation of the global state space.

In this section we always assume that *Sys* is a deadlock-free interaction system with a finite set of components K and finite port sets A_i.

Definition 6.1 Let Sys be an interaction system as above and let $j \in K$ be a component.

(i) Let $A'_j \subseteq A_j$ be a subset of actions of j. A'_j is *inevitable* in T_j if only finitely many transitions labeled with $a_j \in A_j \backslash A'_j$ can be performed in T_j before some action from A'_j must be performed.

(ii) Let $\Lambda \subseteq I(C)$ be a nonempty set of interactions and let $j \in K$ be a component. We define $\Lambda[j] := A_j \cap \bigcup\limits_{\alpha \in \Lambda} \alpha$ the set of ports of j that participate in one of the interactions of Λ.

A subset of actions of a component is inevitable if on every infinite path in the transition system of that component there are infinitely many transitions that are labeled with some action from that set. The second part of the definition gives a sort of a projection-operator that yields those actions of component j that participate in one of the interactions in Λ.

In the following we define the graph $G := (K, E)$. The set of edges is given by the union $\bigcup_{m \geq 0} E_m$ where the sets $E_m \subseteq K \times K$ are defined inductively.

Definition 6.2 Let

$$E_0 := \{(i,j) \,|A_j \backslash (excl(i)[j]) \text{ is inevitable in } T_j\}$$

and define E_{n+1} inductively as follows:

$$E_{n+1} := \{(i,j) \,|A_j \backslash (excl(Reach^n(i))[j]) \text{ is inevitable in } T_j\}$$

Here $Reach^n(i) := \{j | j \text{ is reachable from } i \text{ in } (K, \bigcup_{m=0}^{n} E_m)\}$.

Let $E := \bigcup_{m=0}^{\infty} E_m$ and define $G := (K, E)$.

Note that $excl(i)[j]$ contains those ports of j that occur in some maximal or complete interaction that does not involve i. Thus $A_j \backslash (excl(i)[j])$ is the set of ports of j that only occur in maximal or complete interactions that also involve i. Then the intuitive meaning of an edge $(i,j) \in E$ is that j can only participate in finitely many global steps before i also has to participate in such a step.

Theorem 6.3 *Let $K' \subseteq K$ be a set of components. If all components in $K \backslash K'$ are reachable from K' in G then K' is live in Sys.*

Example 2.2 continued We have already explained why Sys_5 is deadlock-free.

Figure 4 depicts the part of G only containing the edges from E_0. The only component that is reachable from 1 is 2, but it can be seen that 3 respectively 5 can only advance finitely many times before component 4 has

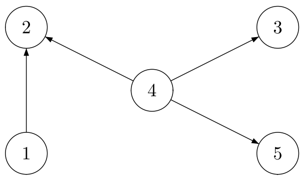

Fig. 4. (K, E_0) for Example 2.2

to participate in some step. Amongst others the edge $(1,4)$ is added to E_1 in the next iteration step. Hence all components are reachable from component 1 in $\left(K, \bigcup_{m=0}^{1} E_m\right)$ and therefore also in G. Then liveness of component 1 follows from Theorem 6.3 above.

In the following we present the algorithm that tests a given set K' of components for liveness. The algorithm first applies the sufficient condition of Theorem 6.3 to K' which causes polynomial cost. If the condition of the criterion is not satisfied then the algorithm applies the characterization of Theorem 5.3 to the set *Reach* of components that can be reached in G from K'. Note that the algorithm requires a system where each global state is a potential starting state. It can easily be adapted to the general case. In this case the algorithm reports "don't know" if it detects a cycle in $\overline{Q}$ in the second part of the algorithm as we do not know if the cycle is reachable in the global system. A further refinement could use the techniques of [25] to find out whether the cycle is indeed reachable from a global starting state.

Lemma 6.4 *Algorithm 1 terminates and correctly tests a given set K' of components for liveness. If it yields a positive answer in line 20 the total cost is polynomial in the size of the input.*

*The **else**-block starting in line 21 causes cost in the size of $\overline{Q}$ which is exponential in $|K \backslash Reach|$ in the worst case.*

Proof. The correctness of the algorithm follows from Theorems 5.3 and 6.3.

Each iteration of the loop in line 5 causes cost polynomial in $|K|$, $|C \cup Comp|$, and $\sum_{j \in K} |T_j|$, and this iteration will be performed at most $|K|^2$ times. Note that the test for inevitability in T_j from line 9 only causes cost polynomial in $|T_j|$. It can be performed by checking whether the system T_j' obtained by deleting all edges labeled with a port from $A_j \backslash \left(excl\left(Reach\left(i\right)\right)[j]\right)$ does not contain a cycle. The cost of the **else**-block is dominated by the cost for the computation of $\overline{Q}$ and the search for a cycle in $\overline{Q}$. □

Example 2.2 continued From the above explanations it is clear that Algorithm 1 launched with Sys_5 and $K' = \{1\}$ terminates in line 20 detecting liveness of component 1.

Algorithm 1 $LIVENESS\,(Sys, K')$

```
Require: $Sys = \left(CS, IM, \{T_i\}_{i \in K}\right)$ deadlock-free, $T_i = (Q_i, A_i, \rightarrow_i, Q_i)\,, K' \subseteq K$
Ensure: TRUE if $K'$ is live, FALSE otherwise
 1: $V \leftarrow K$, $E \leftarrow \emptyset$, $numberEdges \leftarrow 0$
 2: for all $i \in K$ do
 3:    $Reach\,(i) \leftarrow \{i\}$
 4: end for
 5: repeat
 6:    $numberEdges \leftarrow |E|$
 7:    for all $i \in K$ do
 8:       for all $j \in K \backslash \{i\}$ do
 9:          if $A_j \backslash (excl\,(Reach\,(i))\,[j])$ is inevitable in $T_j$ then
10:             $E \leftarrow E \cup \{(i, j)\}$
11:          end if
12:       end for
13:    end for
14:    for all $i \in K$ do
15:       $Reach\,(i) \leftarrow \{j \in K | \exists$ path from $i$ to $j$ in $(V, E)\}$
16:    end for
17: until $numberEdges = |E|$
18: $Reach \leftarrow \bigcup_{i \in K'} Reach\,(i)$
19: if $Reach = K$ then
20:    return TRUE {$K'$ is live}
21: else
22:    compute $excl\,(Reach)$
23:    compute $\overline{Reach}$
24:    $\overline{Q} \leftarrow \prod_{k \in \overline{Reach}} Q_k$
25:    if $\nexists$ cycle in $\overline{Q}$ then
26:       return TRUE {$K'$ is live}
27:    else
28:       return FALSE {$K'$ is not live}
29:    end if
30: end if
```

In the following example the **else**-block will be applied and yields liveness of component 1 if the algorithm is launched with the given system and $K' = \{1\}$.

Example 6.5 Consider a system consisting of the four components 1, 2, 3, and 4 whose behavior is given by Fig. 5 where it is understood that the port sets of the components are given implicitly by the transition systems. For every component i we put $Q_i^0 = Q_i$.

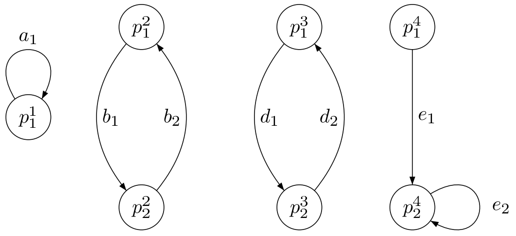

Fig. 5. Example illustrating Theorem 5.3

We define $C = \{\{a_1, b_1\}, \{b_2, d_1\}, \{d_2, e_1\}, \{d_1, e_1\}, \{d_2, e_2\}\}$ and set $Comp = \emptyset$.

It is easy to check that the global system is deadlock-free. We apply Algorithm 1 to test liveness of $K' = \{1\}$. The algorithm finds $Reach = \{1, 2\} \neq K$. The **else**-block will therefore be applied to $\underline{\{1, 2\}}$. Because $excl(\{1, 2\}) = \{\{d_2, e_1\}, \{d_1, e_1\}, \{d_2, e_2\}\}$ it is clear that $\overline{Reach} = \{3, 4\}$. There is no cycle in $\bar{Q}$. Thus the algorithm affirms liveness of 1.

Had we used Theorem 5.3 directly to test liveness of component 1 we would have got $\bar{K}' = \{2, 3, 4\}$, i.e. a larger transition system would have had to be investigated. The combination of the criterion from Theorem 6.3 with the characterization of Theorem 5.3 in Algorithm 1 may yield greater benefit if applied to larger examples.

7 Conclusion and Related Work

This work treats various aspects concerning liveness. The contribution is threefold:

(i) We showed that deciding liveness in interaction systems is NP-hard by reducing 3-SAT to resolving the question whether a certain component is live in an interaction system[4].

(ii) We presented a characterization for liveness.

(iii) We established a sufficient criterion for liveness that can be tested in polynomial time. In Algorithm 1 we combined this criterion with the characterization mentioned above.

Liveness has been treated in other settings for component-based systems. For example in the channel-based approach of [7] general liveness-properties

[4] Work in progress suggests the conjecture that the problem is even PSPACE-hard.

have been investigated although no procedures that can be used to test liveness are provided. Moreover Petri-nets have been used for component-based modeling [3]. For Petri-nets a notion of liveness has been investigated in detail [8] and depending on the considered class of Petri-nets the complexity of deciding this property is presented. This notion of liveness corresponds to our notion of local progress [14]. Liveness has been also discussed in [12,30].

The problem of repeated reachability of a set of accepting states of a Büchi automaton has been dealt with in depth in the context of model checking of LTL formulae, see e.g. [20]. This problem corresponds to our condition in Theorem 5.3. However we propose an alternative idea. We exploit local information about the components and derive a criterion in Theorem 6.3, that guarantees liveness without considering (parts of) the global state space in *any* form. Algorithm 1 proceeds as follows: only if the criterion fails to establish liveness we apply cycle search in the projection of the global state space to $\overline{Reach}$. To implement this part of the algorithm efficiently we could apply the ideas presented in [20].

Another approach to establish properties of systems while avoiding global state space analysis is to exploit compositionality. In [14] we defined a composition operator for interaction systems and presented some first conditions under which properties of subsystems are preserved under composition.

In [14,25,24] we formulated and investigated further properties of interaction systems, in particular global and local deadlock, progress, and robustness. Currently we are enhancing the model by introducing probability. It is then possible to make statements such as "with probability p no deadlock will arise".

References

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

[2] Farhad Arbab. Abstract Behavior Types: A Foundation Model for Components and Their Composition. In *Formal Methods for Components and Objects (FMCO 02)*, volume 2852 of *LNCS*, pages 339–360, 2003.

[3] Remi Bastide and Eric Barboni. Software Components: A Formal Semantics Based on Coloured Petri Nets. In *Proceedings of the International Workshop on Formal Aspects of Component Software (FACS 05)*, ENTCS, 2005.

[4] A. Basu, M. Bozga, and J. Sifakis. Modeling Heterogeneous Real-time Systems in BIP. In *Proceedings of the Third IEEE Conference on Software Engineering and Formal Methods (SEFM 2006)*, pages 3–12, 2006.

[5] Hubert Baumeister, Florian Hacklinger, Rolf Hennicker, Alexander Knapp, and Martin Wirsing. A Component Model for Architectural Programming. In *Proc. 2nd Int. Wsh. Formal Aspects of Component Software (FACS 05)*, volume 160 of *Elect. Notes Theo. Comp. Sci.*, pages 75–96, 2006.

[6] M. Bozga, O.Constant, B. Josko, Q. Ma, and M. Skipper. SPEEDS metamodel syntax and static semantics, January 2007. Deliverable D2.1b.

[7] 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.

[8] Allan Cheng, Javier Esparza, and Jens Palsberg. Complexity Results for 1-safe Nets. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science (FSTTSC 93)*, volume 761 of *LNCS*, pages 326–337, 1993.

[9] S. Chouali, M. Heisel, and J. Souquières. Proving Component Interoperability with B Refinement. In *Proc. 2nd Int. Wsh. Formal Aspects of Component Software (FACS 05)*, volume 160 of *Elect. Notes Theo. Comp. Sci.*, pages 67–84, 2006.

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

[11] M. R. Gary and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* New York: W.H. Freeman, 1979.

[12] 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 the 33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 07)*, volume 4362 of *LNCS*, pages 295–308, 2007.

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

[14] Gregor Gössler, Susanne Graf, Mila Majster-Cederbaum, Moritz Martens, and Joseph Sifakis. Ensuring Properties of Interaction Systems. In *Program Analysis and Computation, Theory and Practice*, volume 4444 of *LNCS*, pages 201–224, 2007.

[15] Gregor Gössler and Joseph Sifakis. Component-based construction of Deadlock-free Systems. In *Proceedings of Foundations of Software Technology and Theoretical Computer Science (FSTTCS 03)*, volume 2914 of *LNCS*, pages 420–433, 2003.

[16] Gregor Gössler and Joseph Sifakis. Composition for Component-Based Modeling. In *Formal Methods for Components and Objects (FMCO 02)*, volume 2852 of *LNCS*, pages 443–466, 2003.

[17] Gregor Gössler and Joseph Sifakis. Priority Systems. In *Formal Methods for Components and Objects FMCO 03)*, LNCS, pages 314–329, 2004.

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

[19] S. Graf and S. Quinton. Contracts for BIP: Hierarchical Interaction Models for Compositional Verification. In *Proceedings of FORTE'07*, volume 4574 of *LNCS*, pages 1–18. Springer, 2007.

[20] R.H. Hardin, R.P. Kurshan, S.K. Shukla, and M.Y. Vardi. A New Heuristic for Bad Cycle Detection Using BDDs. In *Proceedings of CAV'97*, volume 1254 of *LNCS*, pages 268–278, 1997.

[21] Paola Inverardi and Sebastian Uchitel. Proving Deadlock Freedom in Component-Based Programming. In *Proceedings of the 4th International Conference on Fundamental Approaches to Software Engineering (FASE 01)*, volume 2029 of *LNCS*, pages 60–75, 2001.

[22] Bernd J. Krämer, Heinz W. Schmidt, Iman H. Poernomo, and Ralf H. Reussner. Predictable Component Architectures Using Dependent Finite State Machines. In *Radical Innovations of Software and Systems Engineering in the Future, 9th International Workshop (RISSEF 2002)*, pages 310–324, 2002.

[23] Nancy A. Lynch and Mark R. Tuttle. An Introduction to Input/Output Automata. *CWI-Quarterly*, 2(3):219–246, September 1989.

[24] M. Majster-Cederbaum and M. Martens. Robustness in Interaction Systems. In *Proceedings of the 27th International Conference on Formal Methods for Networked and Distributed Systems (FORTE 07)*, volume 4574 of *LNCS*, pages 325–340, 2007.

[25] M. Majster-Cederbaum, M. Martens, and C. Minnameier. A Polynomial-Time-Checkable Sufficient Condition for Deadlock-freeness of Component Based Systems. In *Proceedings of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 07)*, volume 4362 of *LNCS*, pages 888–899, 2007.

[26] M. Majster-Cederbaum and C. Minnameier. Deriving Complexity Results for Interaction Systems from 1-safe Petri Nets., 2007. Submitted for publication.

[27] M. Majster-Cederbaum, N. Semmelrock, and V. Wolf. Interaction Models for Biochemical Reactions. In *Proceedings of BIOCOMP 07*. CSREA Press, 2007.

[28] M. Martens, C. Minnameier, and M. Majster-Cederbaum. Deciding Liveness in Component-Based Systems is NP-hard. Technical report TR-2006-017, University of Mannheim, Fakultät Mathematik und Informatik, 2006. http://madoc.bib.uni-mannheim.de/madoc/portal/inst_inf/index.php.

[29] Moritz Martens. Liveness in Interaction Systems - An Application to the Problem of the Dining Philosophers., 2006. Nordic Workshop on Programming Theory 2006.

[30] Moritz Martens. Liveness in Interaction Systems. Technical report TR-2007-001, University of Mannheim, 2007. http://madoc.bib.uni-mannheim.de/madoc/portal/inst_inf/index.php.

[31] Moritz Martens. Using Interaction Systems to Model a Bank System, 2007. Internal Report.

[32] Christoph Minnameier. Deadlock-Detection in Component-Based Systems is NP-hard. *Information Processing Letters*, 3630, 2007.

[33] S. Moschoyiannis and M.W. Shields. Component-Based Design: Towards Guided Composition. In *Proceedings of Application of Concurrency to System Design (ACSD'03)*, pages 122–131. IEEE Computer Society, 2003.

[34] Oscar Nierstrasz and Franz Achermann. A Calculus for Modeling Software Components. In *Formal Methods for Components and Objects (FMCO 02)*, volume 2852 of *LNCS*, pages 339–360, 2003.

[35] F. Plášil, D. Bálek, and R. Janeček. SOFA/DCUP: Architecture for Component Trading and Dynamic Updating. In *CDS '98: Proceedings of the International Conference on Configurable Distributed Systems*, page 43, Washington, DC, USA, 1998. IEEE Computer Society.

[36] Ralf H. Reussner, Heinz W. Schmidt, and Iman Poernomo. Reliability Prediction for Component-Based Software Architectures. *Journal of Systems and Software – Special Issue of Software Architecture – Engineering Quality Attributes*, 66(3), 2003.

[37] Heinz W. Schmidt. Compatibility of Interoperable Objects. In Bernd Krämer, Michael P. Papazoglou, and Heinz W. Schmidt, editors, *Information Systems Interoperability*. Research Studies Press, 1998.

[38] Heinz W. Schmidt. Architecture-Based Reasoning About Performability in Component-Based Systems. In *Proceedings of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 07)*, volume 4362 of *LNCS*, pages 130–137, 2007.

[39] J. Sifakis. The Algebra of Connectors Structuring Interaction in BIP. In *Proceedings of FMCO'07*, LNCS. Springer, 2007. To appear.

[40] Joseph Sifakis. Modeling Real-time Systems. In *IEEE Real-Time Symposium (RTSS04)*, pages 5–6, 2004.

[41] Joseph Sifakis. A Framework for Component-based Construction. In *IEEE Conference on Software Engineering and Formal Methods (SEFM 05)*, pages 293 – 300, 2005.