An Approach to Modelling and Verification of Component Based Systems

G. Gössler[(1)], S. Graf[(2)], M. Majster-Cederbaum[(3)], M. Martens[(3)], J. Sifakis[(2)]

[(1)] INRIA Rhône-Alpes
Montbonnot, France
gregor.goessler@inria.fr

[(2)] VERIMAG
Grenoble, France
{graf,sifakis}@imag.fr

[(3)] University of Mannheim
Mannheim, Germany
mcb@informatik.uni-mannheim.de

Abstract. We build on a framework for modelling and investigating component-based systems that strictly separates the description of behavior of components from the way they interact. We discuss various properties of system behavior as liveness, local progress, local and global deadlock, and robustness. We present a criterion that ensures liveness and can be tested in polynomial time.

1 Introduction

Component-based design techniques are an important paradigm for mastering design complexity and enhancing reusability. In the abstract data type view or object-oriented approach subsystems interact by invoking operations or methods of other subsystems in their code and hence rely on the availability and understanding of the functionality of the invoked operations. In contrast to this, components are designed independent from their context of use. Components may be glued together via some kind of gluing mechanism. This view has lead some authors, e.g. [3,8,21,9] to consider a component as a black box and to concentrate on the combination of components using a syntactic interface description of the components.

Nevertheless, for these techniques to be useful, it is essential that they guarantee more than syntax-based interface compatibilities. No matter if a certain functionality has to be established or certain temporal properties must be ensured, knowledge about the components has to be provided. Methods based on the assume-guarantee paradigm [23] or similarly on the more recent interface automata [10] are useful e.g. for the verification of safety properties provided that they can be easily decomposed into a conjunction of component properties. Other approaches rely on some process algebra as CSP or $\pi - calculus$ [22,17,1] and consider congruences and reductions to discuss properties of component systems.

We build here on a framework for component-based modelling, called *interaction systems*, that was proposed in [14,15,13,24], which clearly separates interaction from (local) behavior of components. In [15] a notion of global deadlock-freedom, called interaction safety there, was introduced and investigated for interaction systems.

Here, we explain how the framework can be used to discuss properties of systems including liveness, progress of subsystems, robustness and fairness.

In most cases direct testing of the properties relies on an exploration of the global state space and hence cannot be performed efficiently. E.g. we have shown that deciding local and global deadlock-freedom as well as deciding liveness is NP-hard for component-based systems [20,19]. Alternatively one may establish conditions that entail a desired property and that can be tested more efficiently. In [13] a first condition was given that entails global deadlock-freedom of interaction systems. In [18] we established a condition that entails local deadlock-freedom of interaction systems and that can be tested in polynomial time.

Here we present a condition that can be tested in **polynomial** time and guarantees **liveness** of a component, of a set of components or of an interaction in an interaction system.

We present here a simple version of the framework without variables.

In section 1 we introduce the framework and model a version of the dining philosophers as an interaction system. In section 2 we consider properties of interaction systems and illustrate them by examples. In section 3 we present and analyze a condition for liveness that can be tested in polynomial time. Section 4 discusses related and future work.

2 Components, Connectors and Interaction Systems

We build on a framework of [13,15] where components $i \in K$ together with their port sets A_i are the basic building blocks. Each component offers an interface which is here given as a set of ports. Each component i has a local behavior that is here given by a local transition system T_i. The local transition system regulates the way in which ports are available for cooperation with the environment. Components can be glued together. The gluing is achieved by a set of connectors. A connector is a set of ports where no two ports belong to the same component. An interaction is a subset of a connector. Certain interactions may be declared as *complete* interactions. If an interaction is declared *complete*, it can be performed independently of the environment. If an interaction is complete, so should be all its supersets.

Given the above ingredients we define the notion of an *interaction system* with a global behavior that is obtained from the local transition systems by taking the connectors into account, more formally:

Definition 1

Let K be a set of **components**, *A_i, $i \in K$, a* **port set** *that is disjoint from the port set of every other component. Ports $a_i, b_i ... \in A_i$ 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**, *if it contains at most one port of each component $i \in K$. A* **connector set** *is a set C of connectors where:*

a) every port occurs in at least one connector of C,

b) no connector contains any other connector.

For connector set C we denote by $I(C)$ all non empty subsets (called **interactions**) *of the connectors in C, i.e. $I(C) = \{\beta \neq \emptyset \mid \exists c \in C \ \beta \subseteq c\}$. We abbreviate I(C) to IC for ease of notation. The elements $c \in C$ are maximal in IC with respect to set inclusion and are hence called* **maximal interactions**.

A set $U \subseteq IC$ of interactions is said to be **closed** *w.r.t. IC, if whenever $u \in U$ and $u \subset v \in IC$ then $v \in U$. Let* **Comp** *be a closed set of interactions. It represents the complete interactions.*

Let for each component $i \in K$ a transition system $T_i = (Q_i, A_i, \rightarrow_i)$ be given, where $\rightarrow_i \subseteq Q_i \times A_i \times Q_i$. We write $q_i \stackrel{a_i}{\rightarrow}_i q_i'$ for $(q_i, a_i, q_i') \in \rightarrow_i$. We suppose that $Q_i \cap Q_j = \emptyset$ for $i \neq j$. In the induced interaction system, the components cooperate via interactions in IC. For the notion of a run and the properties studied in this paper only interactions in $C \cup Comp$ will be relevant but for composition of systems as in [11] we need all interactions hence they are admitted (as labels) in the following definition.

The **induced interaction system** *is given by $Sys = (K, C, Comp, T)$, where the global behavior $T = (Q, IC, \rightarrow)$ is obtained from the behaviors of individual components, given by transition the systems T_i, in a straightforward manner:*

– $Q = \prod_{i \in K} Q_i$, *the Cartesian product of the* Q_i, *which we consider to be order independent. We denote states by tuples* $(q_1, ..., q_j, ...)$ *and call them* global states.
– *the relation* $\rightarrow \subseteq Q \times IC \times Q$, *defined by*
$\forall \alpha \in IC \; \forall q, q' \in Q \;\; q = (q_1, ..., q_j, ...) \stackrel{\alpha}{\rightarrow} q' = (q'_1, ..., q'_j, ...)$ *iff*

$\forall i \in K \;\; (q_i \stackrel{i(\alpha)}{\rightarrow}_i q'_i$ *if* i *participates in* α *and* $q'_i = q_i$ *otherwise).*

where for component i *and interaction* α, *we put* $i(\alpha) = A_i \cap \alpha$ *and say that component* i participates *in* α, *if* $i(\alpha) \neq \emptyset$. *The set of states in which* $a_i \in A_i$ *is enabled is denoted by* $en(a_i) = \{q_i \in Q_i \mid q_i \stackrel{a_i}{\rightarrow} q'_i \text{ for some } q'_i\}$

The following example shows how the model of interaction systems can be used to model a solution for the dining philosopher problem.

Example 1

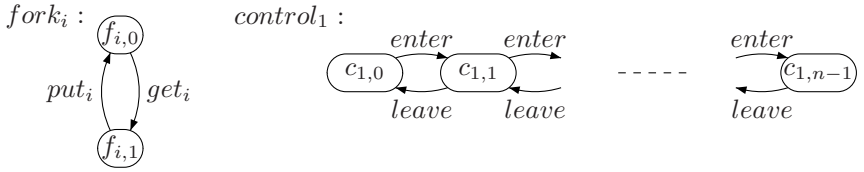
We assume that there are n philosophers as well as n forks.
The alphabet of philosopher$_i$ $i = 0, ..., n-1$ *is*

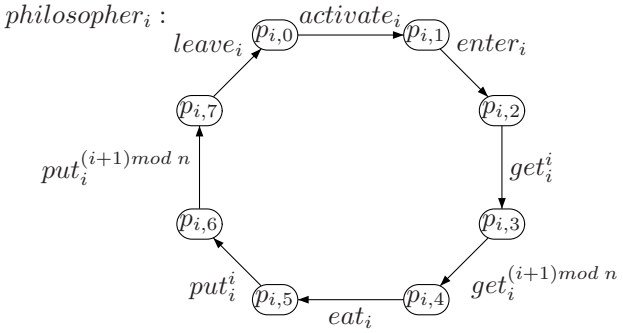
$$\{activate_i, enter_i, get^i_i, get^{(i+1)mod\ n}_i, eat_i, put^i_i, put^{(i+1)mod\ n}_i, leave_i\}.$$

The alphabet of $fork_i$ *is* $\{get_i, put_i\}$. *To achieve deadlock-freedom we introduce a control component. The alphabet of the control component* $control_1$ *is* $\{enter, leave\}$.

We introduce the following connectors
$eat = \{eat_0, eat_1, ...eat_{n-1}\}$, *where any nonempty subset is declared to be complete.*
$act = \{activate_0, activate_1, ...activate_{n-1}\}$, *and for* $i = 0, ..., n-1$
$\{enter, enter_i\}$, $\{leave, leave_i\}$,
$\{get^i_i, get_i\}$, $\{get^{(i+1)mod\ n}_i, get_{(i+1)mod\ n}\}$,
$\{put^i_i, put_i\}$, $\{put^{(i+1)mod\ n}_i, put_{(i+1)mod\ n}\}$.

The transition systems are given by





E.g. *act* is the only interaction in $C \cup Comp$ that may take place in the global state $q_0 = (p_{1,0}, ..., f_{1,0}, ..., c_{1,0})$. Then, $\{enter, enter_i\}$ can take place for a philosopher i.

3 Properties of Interaction Systems

We consider in the following several essential properties of component based systems and show how they can be clearly defined in our setting. In what follows, we consider an interaction system

$$Sys = (K, C, Comp, T)$$

where $T = (Q, IC, \rightarrow)$ is constructed from given transition systems T_i, $i \in K$, as described in Definition 1.

Let P be a predicate on the global state space. We assume here that P is an **inductive invariant**, i.e. $\forall q, q' \in Q\ \forall \alpha \in C \cup Comp\ (P(q) \wedge q \xrightarrow{\alpha} q' \Rightarrow P(q'))$. As an example we consider the predicate $P_{reach(q_0)}$ describing all global states that are **reachable** (via interactions in $C \cup Comp$) from some designated starting state q_0.

The first property under consideration is P-deadlock-freedom which is a generalization of the concept of deadlock-freedom of [13,15]. An interaction system is considered to be P-deadlock-free if in every global state that satisfies P it may perform a maximal or complete interaction. This definition is justified by the fact that both for complete and maximal interactions there is no need to wait for other components to participate. Deadlock-freedom is an important property of a system. But it does not provide any information about local deadlocks in which some set $\emptyset \neq K' \subseteq K$ of components might be involved and hence we consider this situation as well.

Definition 2
*Let Sys be an interaction system. Sys is called P-**deadlock-free** if for every state $q \in Q$ satisfying P there is a transition $q \xrightarrow{\alpha} q'$ with $\alpha \in C \cup Comp$.*

*Let $K' \subseteq K, K' \neq \emptyset$. K' is involved in a **local** P-**deadlock** in state q satisfying P, if for any $i \in K'$ and for any $a_i \in A_i$ if $q_i \in en(a_i)$ then for any $\alpha \in C \cup Comp$ with $a_i \in \alpha$ there is some $j \in K'$ and some $a_j \in A_j \cap \alpha$ such that $q_j \notin en(a_j)$. Sys is called* **locally** *P-**deadlock-free** if in any P-state q there is no $\emptyset \neq K' \subseteq K$ that is involved in a local P-deadlock in q. For $P = true$ we speak of* **(local) deadlock-freedom**.

Remark 1
If Sys is locally P-deadlock-free then it is P-deadlock-free. The converse does not hold.

In addition to deadlock-properties it is interesting to consider the property of P-progress of K', i.e. the property that at any point of any P-run of the system, there is an option to proceed in such a way that some component of K' will eventually participate in some interaction. A subset K' of components is said to be P-live if K' participates infinitely often in every P-run. Please note that we admit only transitions labelled by elements in $C \cup Comp$ for the definition of a P-run.

Definition 3
*Let Sys be a P-deadlock-free interaction system. A P-**run** of Sys is an infinite sequence $\sigma = q_0 \xrightarrow{\alpha_0} q_1 \xrightarrow{\alpha_1} q_2 \ldots$ where $q_l \in Q$ and $P(q_l) = true$ and $\alpha_l \in C \cup Comp$ for any l. For $n \in \mathbb{N}$, σ_n denotes the prefix $q_0 \xrightarrow{\alpha_0} q_1 \xrightarrow{\alpha_1} q_2 \ldots \xrightarrow{\alpha_{n-1}} q_n$. Let $\emptyset \neq K' \subseteq K$. K' **may** P-**progress in** Sys, if for any P-run σ of Sys and for any $n \in \mathbb{N}$ there exists σ' such that $\sigma_n\sigma'$ is a P-run of Sys and some $i \in K'$ participates in some interaction α of σ'.*

*$K' \subseteq K$ is P-**live** in Sys, if every P-run of Sys encompasses an infinite number of transitions where some $i \in K'$ participates, i.e. for every P-run σ and for all $n \in \mathbb{N}$ there is an m with $m \geq n$ and there is $i \in K'$ with $i(\alpha_m) \neq \emptyset$. An interaction $\alpha \in IC$*

is P**-live**, *if every* P*-run encompasses an infinite number of transitions* $q \xrightarrow{\beta} q'$ *with* $\alpha \subseteq \beta$.

Sys is called P**-fair** *if every component* $i \in K$ *is* P*-live in* Sys.

If $P = true$, *we speak of liveness, similarly we speak of runs, fairness etc. Also we identify a singleton set with its element.*

Remark 2
If Sys *is* P*-deadlock-free and at least one state satisfies* P *then* P*-runs exist as* P *is an inductive invariant.*

Lemma 1
Let Sys *be* P*-deadlock-free,* $\emptyset \neq K' \subseteq K$. *If* K' *may* P*-progress then* K' *is not involved in a local* P*-deadlock in any* P*-state. If* Sys *is locally* P*- deadlock-free this does not imply that every component may* P*-progress.*
Proof: see Appendix

If we consider a setting where a component may involve a technical device, as e.g. in an embedded system, that may break down, we might be interested to know how the properties behave on the failure of that component. As an example we treat here deadlock-freedom and progress.

Definition 4
Let Sys *be a deadlock-free interaction system. In* Sys, *deadlock-freedom is called* **robust** *with respect to* **failure of port** $a_i \in A_i$, *if in every state* $q \in Q$ *there is a transition* $q \xrightarrow{\alpha} q'$ *with* $\alpha \in C \cup Comp$ *and* $a_i \notin \alpha$. *In* Sys *deadlock-freedom is called* **robust** *with respect to* **failure of component** i, *if in every state* $q \in Q$ *there is a transition* $q \xrightarrow{\alpha} q'$ *with* $\alpha \in C \cup Comp$ *and* $i(\alpha) = \emptyset$.

Let in Sys *deadlock-freedom be robust with respect to failure of port* $a_i \in A_i$. *Suppose that* $j \in K, i \neq j$, *may progress in* Sys. *The progress property of* $\{j\}$ *is* **robust** *with respect to* **failure of port** $a_i \in A_i$, *if for any run* σ *of* Sys *and for any* $n \in \mathbb{N}$ *there exists* σ' *such that* $\sigma_n\sigma'$ *is a run of* Sys *and such that there is some interaction* α *of* σ' *with* $j(\alpha) \neq \emptyset$ *and no interaction of* $\sigma_n\sigma'$ *contains* a_i.

Remark 3
The philosopher system in Example 1 is $P_{reach(q_0)}$*-deadlock-free where* $q_0 = (p_{1,0}, ..., f_{1,0}, ..., c_{1,0})$. *This is due to the control component that admits at most* $n-1$ *philosophers to the table. By a pigeon hole argument at least one philosopher can get both forks and continue. When he leaves, another philosopher may be admitted to the table. As we will see in section 1 each philosopher is* $P_{reach(q_0)}$*-live in the system and will eat at some time. Hence the system is* $P_{reach(q_0)}$*-fair.*

In the following we show how we can model a system of n identical tasks that have to be scheduled as they all need the same resource in mutual exclusion. Here no explicit representation of a *scheduler* or a *controller* is used. For this example we introduce a rule of *maximal progress*. The maximal progress rule restricts the transition relation for Sys to maximal transitions, i.e. to those transitions such that $q \xrightarrow{\alpha} q'$, implies that there is no β, q'' with $\alpha \subsetneq \beta$ and $q \xrightarrow{\beta} q''$.

Example 2
We consider a set of tasks T_i $(i \in K = \{1, ..., n\})$ *that compete for some resource in mutual exclusion. The basic behavior of each task is given in Figure 1 and needs not to be further explained. Let the set of ports of each component* i *be:*

$$A_i = \{activate_i, start_i, resume_i, preempt_i, finish_i\}.$$

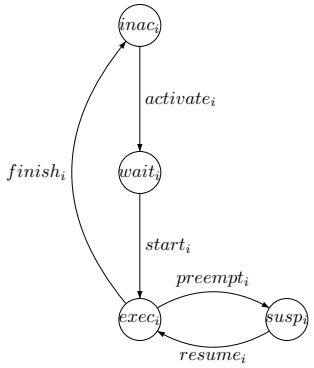


Fig. 1. Basic behavior of each task T_i

We want to guarantee mutual exclusion with respect to the exec state, i.e. no two tasks should be in this state at the same time, in the sense that this is an inductive invariant.

Mutual exclusion, in this sense, can be achieved using the rule of maximal progress and by the following connectors: for $i, j \in K, i \neq j$, $conn_1^i = \{activate_i\}$, $conn_2^{ij} = \{preempt_i, start_j\}$, $conn_3^{ij} = \{resume_i, finish_j\}$. $\{start_j\}$ *and* $\{finish_j\}$ *are defined to be complete. Let* Sys_{tasks} *be the system defined this way.*

Observation 1: On every run starting in a state where no two components are in their exec state Sys_{tasks} guarantees mutual exclusion with respect to the exec state due to the rule of maximal progress. For detailed explanations see appendix.

Observation 2: Let $P = (\exists i\ q_i \neq susp_i)$. P is an inductive invariant and Sys_{tasks} is P-deadlock-free and each component i may P-progress. The property of P-deadlock-freedom is robust with respect to failure of the operation $resume_i$.

Observation 3: Let $P' = true$ then Sys_{tasks} is not P'-deadlock-free as in the state $q = (susp_1, ..., susp_n)$ no interaction in $C \cup Comp$ is available. We may modify the system by introducing a new action $reset_i$ for each i and by enriching the transition system T_i with an edge labelled $reset_i$ from the state $susp_i$ to $inac_i$. In addition we introduce a connector $conn_g = \{reset_1, ..., reset_n\}$. The resulting system Sys'_{tasks} is P'-deadlock-free. P'-Deadlock-freedom is not robust with respect to failure of port $reset_i$ and hence not robust with respect to failure of component i, $i \in K$. Alternatively we might consider the state $q_0 = (inac_1, ..., inac_n)$. The state $q = (susp_1, ..., susp_n)$ is not reachable from q_0 and Sys_{tasks} is $P_{reach(q_0)}$-deadlock-free.

Observation 4: When a finite interaction system is P-deadlock-free for some in inductive invariant P, then it is also P-deadlock-free when we apply the rule of maximal progress.

Observation 5: In Sys'_{tasks} every component may P'-progress under the rule of maximal progress. For detailed explanation see appendix.

4 Testing Liveness of Components in Interaction Systems

In the previous examples we have verified some properties of the systems directly. As the properties are conditions on the global state space, they cannot be established directly in an efficient way in general. E.g. we have shown that deciding local and global deadlock-freedom as well as deciding liveness is NP-hard for component-based systems [20,19]. However, one may define (stronger) conditions that are easier to test and entail the desired properties. In [15], a condition for deadlock-freedom of an interaction system (called interaction safety there) is presented that uses a directed graph. The absence of certain cycles in some finite subgraph ensures deadlock-freedom of the system. This criterion can be extended to *P-deadlock-freedom*, it can be modified to ensure *local progress* as well as *robustness* of P-deadlock-freedom with respect to the failure of a port or a whole component, and it can be extended to apply to a broader class of systems including various solutions for the dining philosophers. In [18] we present a condition that entails local deadlock-freedom and can be tested in polynomial time.
Here we focus on liveness. We present a condition that can be tested in polynomial time and entails liveness for a component $k \in K$. In what follows, we assume for simplicity that the local transition systems T_i have the property that they offer *at least one action* in every state. The general case can be reduced to this case by introducing idle actions or by adapting the definitions and results below to include this situation. To test for liveness we construct a graph, where the nodes are the components, and, intuitively, an edge $i \to j$ means " j needs i ", in the sense that i eventually has to participate in an interaction involving j when j does progress. In transition system T_j we call a set $A \subseteq A_j$ *inevitable*, if every infinite path in T_j encompasses an infinite number of transitions labelled with some action in A.

Theorem 1
Let Sys be a P-deadlock-free interaction system for some finite set K of components and finite alphabets $A_i, i \in K$. The graph G_{live} is given by $(K, \to)$ where

$$i \to j \;\; if \;\; A_j \setminus excl(i)[j] \text{ is inevitable in } T_j.$$

Here $excl(i) = \{\alpha \in C \cup Comp \text{ with } i(\alpha) = \emptyset\}$ and $excl(i)[j]$ is the projection of $excl(i)$ to j, i.e. the actions of j with which j participates in elements of $excl(i)$.

Let $k \in K$. We put $R_0(k) = \{j : \exists \text{ a path } from \;\; k \text{ to } j \text{ in } G_{live}\}$ and
$R_{i+1}(k) = \{l \in K \setminus R_i(k) : \forall \alpha \in C \cup Comp \;\; l(\alpha) \neq \emptyset \Rightarrow \exists j \in R_i(k) \;\; j(\alpha) \neq \emptyset\} \cup R_i(k)$.
If $\bigcup_{i \geq 0} R_i(k) = K$ then k is P-live in Sys.

Proof: Appendix

Lemma 2
Testing the condition of Theorem 1 can be done in polynomial time in the sum of the sizes of the T_i and the size of $C \cup Comp$.

Proof: For the construction of the graph $G_{live} = (K, \to)$, we inspect each local transition system separately. To check if there is an arrow $i \to j$, we remove in the transition system T_j all edges labelled by elements in $A_j \setminus excl(i)[j]$ and determine if there are directed cycles in the resulting transition system. If this is not the case then we include the arrow, otherwise there are infinite paths in T_j that do not contain an element in

$A_j \setminus excl(i)[j]$, hence $A_j \setminus excl(i)[j]$ is not inevitable in T_j. Clearly the graph can be constructed in $O(|K|\Sigma|T_i|+|K|^2|C\cup Comp|)$. Its size is $O(|K|^2)$. Once the graph G_{live} is constructed, it remains to perform a simple reachability analysis to determine $R_0(k)$ which can be achieved in $O(|K|^2)$. The iteration is performed at most $|K|$ times and each $R_i(k)$ has at most $|K|$ elements. In each iteration we consider all $\alpha \in C \cup Comp$. Hence we may calculate $\bigcup R_i(k)$ in $O(|K|^3|C\cup Comp|)$ where $|C\cup Comp|$ is the number of elements in $C \cup Comp$. So testing the condition can be done in polynomial time in the sum of the size of the input.

Remark 4
The condition given in the above theorem can easily be adapted to establish the P-liveness of a set $K' \subseteq K$ of components as well as to establish the P-liveness of an action $a_k \in A_k$ in Sys.

As an application we consider our model for the dining philosophers where we designate q_0 as the starting state and choose the predicate $P_{reach(q_0)}$.
Example 1 continued: G_{live} for the problem of the dining philosophers is as follows where the abbreviations are self-explanatory and we set $n = 3$ for better readability.

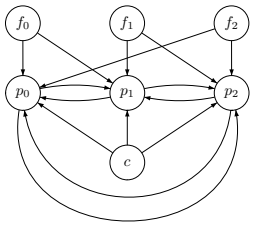


Fig. 2. G_{live} for three philosophers

The criterion now yields that $philosopher_i$ is $P_{reach(q_0)}$-live: take a component that is not in $R_0(philosoher_i)$, e.g. $control_1$. Then for any

$$\alpha \in C \cup Comp \;\; control_1(\alpha) \neq \emptyset \Rightarrow \exists j \in R_0(philosopher_i) \;\; j(\alpha) \neq \emptyset.$$

This is because $control_1$ only participates in interactions in which some philosopher participates as well and all philosophers are connected in the graph. The same holds true for any fork as any interaction involving some fork also involves some philosopher.

5 Discussion and related work

An important motivation for introducing a clean theoretical framework for component based systems is the hope that this will provide means for proving properties such as deadlock-freedom, progress and liveness by exploiting local information and compositionality as much as possible. We showed how the model of interaction systems can be used to deal with important properties of *component based systems*. Testing the definition of these properties directly is not feasible as it usually involves global state space analysis. An alternative is to find conditions that ensure a given property and are easier to test. First conditions for global, resp. local deadlock-freedom have been

treated in [13], resp. [18]. More refined conditions and the treatment of progress can be found in [11]. Here we focussed on liveness. In particular we gave a sufficient condition for the liveness of a component in a component based system that can be tested in polynomial time.

If a condition entailing a desired property is not satisfied we may try to exploit compositionality in the following way: in [11] we define an (associative) composition operator that combines component systems by gluing two component systems with some new connectors. Then we derive conditions under which a property of one/both component systems can be lifted to the new combined and more complex system. Thus incremental construction and verification of systems can be achieved.

Our model of interaction systems has some similarity with the model of input/output-automata in [16] with the difference that in [16] components, represented by automata, share actions. In each step of the global system exactly one action takes place and all components that offer this action participate in this step.

Even though there are many approaches to model component based systems [3,22], [1,17,8,9,21,7,10], to our knowledge the question of properties of component based systems has not yet been studied systematically to great extent. In [7] one can find a condition that ensures the deadlock-freedom of a component based system consisting of two components. In [4] a condition for deadlock-freedom in a strictly interleaving setting of processes that communicate over shared variables is presented. Interface automata [10] have been introduced as a means to specify component interfaces with behavioral types. Two interface automata are compatible if there exists an environment guaranteeing their composition to be deadlock-free. Verifying compatibility amounts to synthesizing the interface automaton of the most permissive environment avoiding deadlock. Liveness, progress or fairness properties are not addressed. In [25] general definitions of fairness are discussed.

In [1] components, ports and (binary) connectors are specified in a CSP-variant and the question under what conditions deadlock-freedom is preserved under refinement is investigated.

There have been attempts to model component based systems with Petri-nets [6,5,2]. Once a system is specified as a Petri-net one might use Petri-net tools to investigate properties of systems provided the specification formalism supports compositionality on the Petri-net level, which is not the case e.g. in [6,5].

Extended versions of our framework, including local variables of components and priority rules as an additional control layer, are presently being implemented. The implementation in the Prometheus tool focusses on compositional verification, in particular. A second implementation, called BIP, focusses on the efficient execution of systems and includes also timed specifications.

The work presented here shows some typical results that can be established in this framework. Further results can be found in [12,11]. The investigation can – and needs to – be extended in different ways for fully *incremental* construction and verification of a large system. The notion of component can be extended with various additional information but also observability criteria and associated equivalence relations are important. Other possible interesting extensions concern introduction of time and probability, as well as dynamic reconfiguration.

References

1. Robert Allen and David Garlan. A formal basis for architectural connection. *ACM Trans. Softw. Eng. Methodol.*, 6(3):213–249, 1997.
2. Nasreddine Aoumeur and Gunter Saake. A component-based petri net model for specifying and validating cooperative information systems. *Data Knowl. Eng.*, 42(2):143–187, 2002.
3. Farhad Arbab. Abstract behavior types: A foundation model for components and their composition. In *Proceedings of FMCO 2002*, volume 2582 of *LNCS*. Springer Verlag, 2002.
4. P. C. 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.
5. Remi Bastide and Eric Barboni. Component-based behavioural modelling with high-level petri nets. In *MOCA '04 - Third Workshop on Modelling of Objects, Components and Agents, Aahrus, Denmark*, pages 37–46. DAIMI, 2004.
6. Remi Bastide and Eric Barboni. Software components: A formal semantics based on coloured petri nets. In *Proceedings of FACS'05*, ENTCS. Elsevier, 2005.
7. H. Baumeister, F. Hacklinger, R. Hennicker, A. Knapp, and M. Wirsing. A component model for architectural programming. In *Proceedings of FACS'05*, ENTCS. Elsevier, 2005.
8. Klaus Bergner et al. A formal model for componentware. In Gary T. Leavens and Murali Sitaraman, editors, *Foundations of Component-Based Systems*, pages 189–210. Cambridge University Press, 2000.
9. S. Chouali, M. Heisel, and J. Souquières. Proving component interoperability with B refinement. In *Proceedings of FACS'05*, ENTCS. Elsevier, 2005.
10. Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *Proceedings of ESEC 2001*, pages 109–120, 2001.
11. Gregor Gössler, Susanne Graf, Mila Majster-Cederbaum, Moritz Martens, and Joseph Sifakis. Establishing properties of interaction systems, 2006. Full paper in preparation.
12. Gregor Gössler, Susanne Graf, Mila Majster-Cederbaum, and Joseph Sifakis. Dynamic construction of deadlock-free interaction systems. Technical report, INRIA, 2006. In preparation.
13. Gregor Gössler and Joseph Sifakis. Component-based construction of deadlock-free systems. In *Proceedings of FSTTCS 2003, Mumbai, India*, volume 2914 of *LNCS*, pages 420–433, December 2003.
14. Gregor Gössler and Joseph Sifakis. Priority systems. In *Proceedings of FMCO'03*, volume 3188 of *LNCS*, April 2004.
15. Gregor Gössler and Joseph Sifakis. Composition for component-based modeling. *Sci. Comput. Program.*, 55(1-3):161–183, 2005.
16. Nancy A. Lynch and Mark R. Tuttle. An introduction to input/output automata. *CWI-Quarterly*, 2(3):219–246, September 1989.
17. J. Magee, N. Dulay, S. Eisenbach, and J. Kramer. Specifying Distributed Software Architectures. In W. Schafer and P. Botella, editors, *Proceedings of ESEC 95*, volume 989 of *LNCS*, pages 137–153. Springer, 1995.
18. Mila Majster-Cederbaum, Moritz Martens, and Christoph Minnameier. A polynomial-time-checkable sufficient condition for deadlock-freeness of component based systems. Accepted to SOFSEM 07.
19. Moritz Martens. Deciding liveness in component-based systems is np-hard, 2006. Internal report.
20. Christoph Minnameier. Deadlock-detection in component-based systems is np-hard. Technical report tr-2006-015, University of Mannheim, Fakultät Mathematik uns Informatik, 2006. submited for publication.
21. S. Moschoyiannis and M. W. Shields. Component-based design: Towards guided composition. In *Proceedings of ACSD'03*, pages 122–131. IEEE Computer Society, 2003.

22. Oscar Nierstrasz and Franz Achermann. A Calculus for Modeling Software Components. In *Proceedings of FMCO 2002*, volume 2582 of *LNCS*, pages 339–360. Springer, 2002.
23. A. Pnueli. In transition from global to modular temporal reasoning about programs. In *Logics and Models for Concurrent Systems*. NATO, ASI Series F, Vol. 13, Springer Verlag, 1985.
24. Joseph Sifakis. A Framework for Component-based Construction. In *Proceedings of SEFM 05*. IEEE Computer Society, 2005.
25. H. Völzer, D. Varacca, and E. Kindler. Defining fairness. In *Proceedings of CONCUR'05*, volume 3653 of *LNCS*, pages 458–472. Springer-Verlag, 2005.

6 Appendix

A1) Sketch of Proof of Lemma 1:

Assume K' may P-progress. We have to show that K' is not involved in a local P-deadlock in any state satisfying P. Assume K' is involved in a local P-deadlock in some state q satisfying P. We consider a run, in which q occurs, e.g. a run starting in q. Then every $k' \in K'$ needs for any interaction in which it might engage some partner in K' which does not provide the desired action. This situation creates a cyclic waiting among groups of components of K' which cannot be resolved by components outside K'. Hence no component in K' may progress yielding a contradiction.
In the following simple example no local deadlock exists but one component may not progress. The example consists of three components with alphabets $A_i = \{a_i, b_i\}$, $i = 1, 2$ and $A_3 = \{a_3\}$. The transition system T_i has two states q_1^i, q_2^i with a transition labelled a_i from q_1^i to q_2^i and a transition b_i from q_2^i to q_1^i, $i = 1, 2$. The transition system T_3 has one state q_1^3 and an edge labelled a_3. The connectors are $c_1 = \{a_1, a_2\}, c_2 = \{b_1, b_2\}, c_3 = \{a_1, b_2, a_3\}$. Let P describe the set of states that are reachable from (q_1^1, q_1^2, q_1^3). The system is locally P-deadlock-free but the last component may not P-progress.

A2) Detailed explanation of Observation 1: Sys_{tasks} guarantees mutual exclusion with respect to the exec state.

Mutual exclusion is guaranteed because whenever T_j enters $exec_j$, either by $start_j$ or $resume_j$, then either there is no other task in its $exec$-state or the task T_i that is in the state $exec_i$ must leave this state. The following items explain why this is the case for each of the two transitions: i) for $resume_j$, the reason is that $resume_j$ can never happen alone. It can only be executed together with the $finish_i$ action if process T_i is currently in the critical state $exec_i$, ii) for $start_j$, which is complete, the reason is the rule of maximal progress: when T_i is in the critical state $exec_i$, it can execute the $preempt_i$ action. Therefore, $start_j$ cannot be executed alone as also the pair $\{preempt_i, start_j\}$ is enabled. On the other hand, if there is no process in the critical section, process j can enter it by executing $start_j$ alone.

A3) Detailed explanation of Observation 5: Let $P' = true$. In Sys'_{tasks} every component may P'-progress under the rule of maximal progress.

As all components have identical behavior it suffices to consider one of them, say component 1. The only situation in which component 1 cannot proceed by itself is when

it is in state $susp_1$. We have to show that we can reach a global state where it can perform a transition: case 1) all other components are in the state *susp*. Then $conn_g$ can happen and component 1 has proceeded, case 2) at least one component j is in the state $exec_j$. Then $\{resume_1, finish_j\}$ may happen, case 3) not all other components are in state *susp* and none is in state *exec*. Then there must be one component j that is in state $inac_j$ or $wait_j$. If it is in state $inac_j$ then it performs the complete action $activate_j$ and reaches state $wait_j$. As there is no component in state *exec* there is no *preempt* action available and $start_j$ may be performed alone even under the rule of maximal progress. Now, $\{resume_1, finish_j\}$ may happen and component 1 has made progress.

A4) For the proof of Theorem 1 we use the following auxiliary lemma:

Lemma 3
Let $\sigma = q_0 \stackrel{\alpha_0}{\rightarrow} q_1 \stackrel{\alpha_1}{\rightarrow} q_2...$ be a P-run. If there is a path $k_0 \rightarrow k_1 \rightarrow .... \rightarrow k_l$ in G_{live} and k_l participates infinitely often in σ then k_0 participates infinitely often in σ.

Proof: by induction on the length l the path.

Start of induction: $l = 1$. Then there is an edge $k_0 \rightarrow k_1$. As k_1 participates infinitely often in transitions of σ and as the set of actions of k_1 that need cooperation of k_0 is inevitable in T_{k_1} we conclude that k_0 participates infinitely often in transitions of σ.

Induction step: $l \rightarrow l+1$. Let $k_0 \rightarrow k_1 \rightarrow ... \rightarrow k_l \rightarrow k_{l+1}$ be a path of length $l+1$ and let k_{l+1} participate infinitely often in σ then by induction assumption k_1 participates infinitely often in σ and as above we conclude that k_0 participates infinitely often.

A5) Proof of Theorem 1:

Let $\sigma = q_0 \stackrel{\alpha_0}{\rightarrow} q_1 \stackrel{\alpha_1}{\rightarrow} q_2...$ be a P-run. We have to show that σ encompasses an infinite number of transitions where k participates. As K is finite and σ infinite there must be some component $\hat{k}$ that participates in infinitely many transitions of σ.

1. $\hat{k} = k$, then we are done.

2. $\hat{k} \neq k$ then we now that $\hat{k} \in \bigcup R_i(k)$.

   case 1: if $\hat{k} \in R_0(k)$ then by the above lemma k and the definition of $R_0(k)$ we conclude that k participates infinitely often in σ.

   case 2: let $\hat{k} \in R_i(k)$ for some $i > 0$. Then we show by induction on i that k participates infinitely often in σ.

   Start of induction $i = 1$: if $\hat{k} \in R_1(k)$ then for all $\alpha \in C \cup Comp$ with $\hat{k}(\alpha) \neq \emptyset$ $\exists j \in R_0(k)$ with $j(\alpha) \neq \emptyset$. As $\hat{k}$ participates infinitely often in σ and as there are only finitely many elements in $C \cup Comp$ there must be some α with $\hat{k}(\alpha) \neq \emptyset$ which occurs infinitely often in σ. By definition of $R_1(k)$ $\exists j \in R_0(k)$ with $j(\alpha) \neq \emptyset$.

Hence j participates infinitely often in σ. As $j \in R_0(k)$ we conclude by the above lemma that k participates infinitely often in σ.

Induction step $i \to i+1$: let $\hat{k} \in R_{i+1}(k)$. As before there is an $\alpha \in C \cup Comp$ with $\hat{k}(\alpha) \neq \emptyset$ and α occurs infinitely often in σ. Some $j \in R_i(k)$ participates in this α, hence j participates infinitely often in σ and by induction assumption k participates infinitely often in σ.