Component-Based Construction of Deadlock-Free Systems

Extended Abstract

Gregor Gössler[1] and Joseph Sifakis[2]

[1] INRIA Rhône-Alpes, goessler@inrialpes.fr
[2] VERIMAG, sifakis@imag.fr

Abstract. We propose a framework for building deadlock-free systems from deadlock-free components. The framework is based on a methodology for the layered construction of systems by superposing three layers. A layer of components, an interaction model and a restriction layer. The interaction model specifies the possible interactions between components. The restriction layer restricts the behavior of the two lower layers by a global constraint. Layered structuring allows separating three orthogonal aspects in system construction. Apart from its methodological interest it makes technically possible the definition of a unique and powerful associative composition operator.
We study sufficient deadlock-freedom conditions for systems built from deadlock-free components and given interaction model and restriction. We also provide a sufficient condition for individual deadlock-freedom of the components of such systems.

1 Introduction

Deadlock-freedom is an essential correctness property as it characterizes a system's ability to perform some activity over its lifetime. Deadlocks are the most common source of errors in systems of concurrent processes. They occur when processes share common resources or are in general subject to strong synchronization constraints. In that case, a process may remain blocked as long as a condition depending on the state of their environment is not satisfied.

It has been often argued that deadlock-freedom is not a relevant property as there exist systems that never deadlock e.g. hardware, time triggered systems [12, 10], synchronous systems [4, 9, 2, 14]. In such systems, components are never blocked by their environment as it is supposed that component inputs are always available whenever requested. Nevertheless, it is clear that if some strong coordination between components is needed e.g. they share a common resource, then this can be achieved by a deadlock-free protocol where the values exchanged between components are used to encode the information necessary for safely implementing the coordination. Thus, for systems that are by construction deadlock-free, verifying mutual exclusion requires analysis of the sequences of the values exchanged between the coordinating components.

Another argument for not considering deadlock-freedom as a relevant property is that any system can become trivially deadlock-free by adding some idle action loop that does not modify its overall observable behavior whenever there is a risk of deadlock e.g. at waiting states. Such a modification allows elimination of deadlocks but it leads to systems where it is possible to indefinitely postpone interaction between components by privileging idle actions. Thus instead of checking for deadlock-freedom, other properties such as livelock-freedom and fairness must be checked for system correctness.

The above remarks show that building deadlock-free systems requires the definition of an appropriate setting where absence of deadlock means satisfaction of strong coordination properties. In this paper we propose a framework for building deadlock-free systems from deadlock-free components. The framework is based on a methodology for the layered construction of systems by superposing three layers (figure 1).

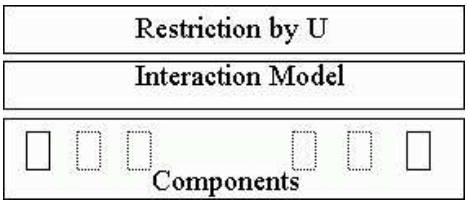

Fig. 1: Layered system description.

- The bottom layer consists of a set of components. They are characterized by the set of actions they can perform and their behavior. The latter is specified as a transition system representing the effect of actions on component states.
- The intermediate layer is an interaction model. This is used to specify the possible interactions between the components. An interaction is the result of the simultaneous occurrence (synchronization) of actions from different components. Furthermore, an interaction model specifies the set of incomplete interactions, that is the interactions that the system cannot execute without synchronizing with its environment. Interaction models are a general mechanism for specifying various kinds of parallel composition [8].
- The upper layer is a restriction by a constraint (predicate) on the system state. Restriction is used to prevent the behavior of the underlying layers from executing interactions that violate the constraint.

System behavior is obtained by successive application of the meaning of each layer. The behavior of components is composed as specified by the interaction model. Then it is restricted by application of a global constraint. We believe that the proposed construction methodology is general enough as it combines

usual parallel composition with techniques relying on the use of constraints such as invariants. For example, in a multitasking system, interaction models can be typically used to describe synchronization for mutual exclusion and restriction can be used to express scheduling policies [1].

We provide sufficient conditions for deadlock-freedom of systems built from deadlock-free components for given interaction model and restriction. The results assume that for each component a deadlock-free invariant is given, that is a set of states from which the component can block only because of its environment. The condition relates the deadlock-free invariants of the system components to the enabling conditions of the interactions of the layered system.

We also provide a sufficient condition for individual deadlock-freedom of the components in a system. A component is individually deadlock-free if it can always perform an action.

The paper is organized as follows.

Section 2 describes the layered construction principle. It deals essentially with the presentation of the interaction models and their properties. Component behaviors are described as simple non deterministic loops with guarded commands [7, 16]. The concept of restriction is taken from [1].

Section 3 presents the result for global and individual deadlock-freedom. Section 4 presents concluding remarks about the presented framework.

2 Composition

We present a composition operation on layered systems. The operation composes each layer separately. The bottom layer of the product is the union of the bottom layers of the operands. The interaction model of the product is obtained as the union of the interaction models of the operands with some "glue" interaction model. That is, the composition operator is parameterised with information about the interactions between the operands. Finally, the restriction of the product is the conjunction of the restrictions of the operands (figure 2).

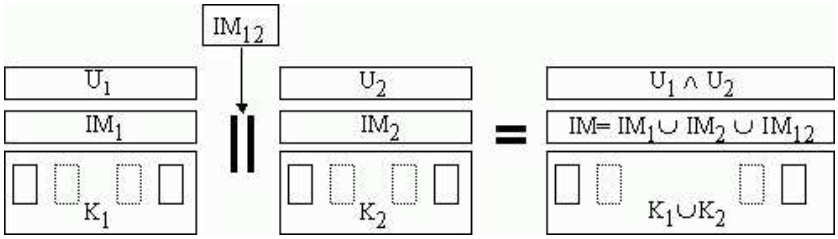

Fig. 2: Layered composition.

The two following sub-sections present the interaction models and their composition.

2.1 Interaction Models

Consider a finite set K of components with disjoint finite vocabularies of actions A_i for $i \in K$. We put $A = \bigcup_{i \in K} A_i$.

A *connector* c is a non empty subset of A such that $\forall i \in K \; . \; |A_i \cap c| \leqslant 1$. A connector defines a maximally compatible set of interacting actions. For the sake of generality, our definition accepts singleton connectors. The use of the connector $\{a\}$ in a description is interpreted as the fact that action a cannot be involved in interactions with other actions.

Given a connector c, an *interaction* α of c is any term of the form $\alpha = a_1 \shortmid \ldots \shortmid a_n$ such that $\{a_1, \ldots, a_n\} \subseteq c$. As usual [15, 3], we assume that $\shortmid$ is a binary associative and commutative operator. It is used to denote some abstract and partial action composition operation. The interaction $a_1 \shortmid \ldots \shortmid a_n$is the result of the occurrence of the actions $a_1, \ldots, a_n$. For α and α' interactions, we write$\alpha \shortmid \alpha'$ to denote the interaction resulting from their composition (if its is defined).

Notice that if $\alpha = a_1 \shortmid \ldots \shortmid a_n$ is an interaction then any term corresponding to a sub-set of $\{a_1, \ldots, a_n\}$ is an interaction. By analogy, we say that α' is a sub-interaction of α if$\alpha = \alpha' \shortmid \alpha''$ for some interaction α''. Clearly,actions are minimal interactions.

The set of the interactions of a connector $c = \{a_1, \ldots, a_n\}$, denoted by $I(c)$, consists of all the interactions corresponding to sub-sets of c (all the sub-interactions of c). We extend the notation to sets of connectors. If C is a set of connectors then $I(C)$ is the set of its interactions. Clearly, for C_1, C_2 sets of connectors, $I(C_1 \cup C_2) = I(C_1) \cup I(C_2)$.

Definition 1 (Set of connectors). *The* set of connectors *of a system consisting of a set of components K with disjoint action vocabularies A_i for $i \in K$, is a set C such that $\bigcup_{c \in C} c = \bigcup_{i \in K} A_i$, and if $c \in C$ then there exists no $c' \in C$ and $c \subsetneq c'$. That is, C contains only maximal sets.*

Definition 2 (Interaction model). *The* interaction model *of a system composed of a set of components K with a set of connectors C is a pair $IM = (I(C), I(C)^-)$ where $I(C)^- \subseteq I(C)$, is the set of the* incomplete *interactions such that it contains no maximal interactions and $\forall b, b' \in I(C)$, $b \in I(C)^-$ and $b' \subseteq b$ implies $b' \in I(C)^-$. We denote by $I(C)^+$ the set of the complete (non incomplete) interactions. Clearly, $I(C)^+$ contains all the maximal interactions of $I(C)$ and is such that $\forall b, b' \in I(C)$, $b \in I(C)^+$ and $b \subseteq b'$ implies $b' \in I(C)^+$.*

Notice that any action appears in some connector. The requirement that C contains only maximal sets ensures a bijective correspondence between the set of connectors C and the corresponding set of interactions $I(C)$. Given $I(C)$, the corresponding set of connectors is uniquely defined and is C. To simplify notation, we write IC instead of $I(C)$.

The distinction between complete and incomplete interactions is essential for building correct models. As models are built incrementally, interactions are obtained by composing actions. It is often necessary to express the constraint that some interactions of a sub-system are not interactions of the system. This is

typically the case for binary strict synchronization (rendez-vous). For example, *send* and *receive* should be considered as incomplete actions but *send*|*receive* as complete. The occurrence of *send* or *receive* alone in a system model is an error because it violates the assumption about strict synchronization made by the designer.

The execution of a complete interaction by a component does not require synchronization with interactions of its environment. The execution of an incomplete interaction requires synchronization with some other interaction to produce a larger one which may be either complete or incomplete. Incompleteness of an interaction implies the obligation to synchronize when environment offers matching interactions as specified by the connectors.

The distinction between complete and incomplete interactions encompasses many other distinctions such as output/input, internal/external, uncontrollable/ controllable used in different modeling formalisms. Clearly, internal actions of components should be considered as complete because they can be performed independently of the state of their environment. In some formalisms, output actions are complete (synchronous languages, asynchronous buffered communication). In some others such as CSP [11] and Lotos [17], all synchronizing actions are incomplete.

Often it is convenient to consider that the complete interactions of IC^+ are defined from a given set of complete actions $A^+ \subseteq A$. That is, IC^+ consists of all the interactions of IC where at least one complete action (element of A^+) is involved. In the example of figure 3, we give sets of connectors and complete actions to define interaction models. By convention, bullets represent incomplete actions and triangles complete actions. In the partially ordered set of the interactions, full nodes denote complete interactions. The interaction between *put* and *get* represented by the interaction *put*|*get* is a rendez-vous meaning that synchronization is blocking for both actions. The interaction between *out* and *in* is asymmetric as *out* can occur alone even if *in* is not possible. Nevertheless, the occurrence of *in* requires the occurrence of *out*. The interactions between *out*, in_1 and in_2 are asymmetric. The output *out* can occur alone or in synchronization with any of the inputs in_1, in_2.

2.2 Incremental Description of Interaction Models

Consider the interaction model $IM = (IC, IC^-)$ of a set of interacting components K with disjoint action vocabularies A_i for $i \in K$. IC andIC^- denote the sets of interactions and incomplete interactions, respectively on the vocabulary of actions $A = \bigcup_{i \in K} A_i$.

Definition 3 (Glue connectors). *Given a set of disjoint subsets of K, $K_1, \ldots, K_n$ of K, we denote by $C[K_1, \ldots, K_n]$ the set of the connectors having at least one action in each set of components, that is, $C[K_1, \ldots, K_n] = \{c = c_1 \cup \cdots \cup c_n \mid \forall i \in [1, n] \,.\, c_i \in C[K_i] \ \wedge \ c \in C[K]\}$.*

Clearly, $C[K_1, \ldots, K_n]$ is the set of the connectors of $IM[K_1 \cup \cdots \cup K_n]$ which are not connectors of any $IM[K']$ for any subset K' of at most $n-1$ elements

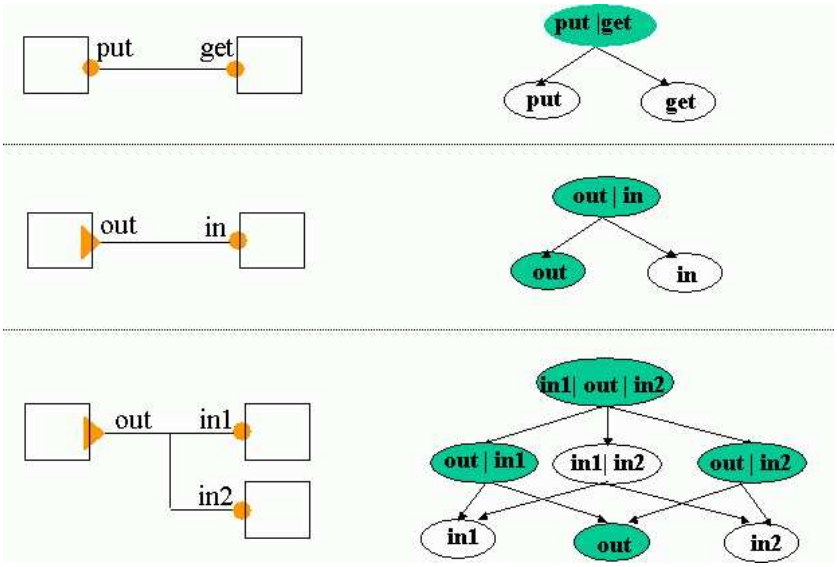

Fig. 3: Interaction models.

from $\{K_1, \ldots, K_n\}$. Notice that when the partition consists of only one set, then the above definition agrees with Definition 1.

Proposition 1. *Given K_1, K_2, K_3 three disjoint subsets of K.*

$$IC[K_1 \cup K_2] = IC[K_1] \cup IC[K_2] \cup IC[K_1, K_2]$$
$$IC[K_1 \cup K_2, K_3] = IC[K_1, K_3] \cup IC[K_2, K_3] \cup IC[K_1, K_2, K_3]$$

Proof. The first equality comes from the fact that $C[K_1] \cup C[K_2] \cup C[K_1, K_2]$ contains all the connectors of $C[K_1 \cup K_2]$ and other sets of actions that are not maximal. By definition, IC contains all the sub-sets of C. Thus, $IC[K_1 \cup K_2] = I(C[K_1] \cup C[K_2] \cup C[K_1, K_2]) = IC[K_1] \cup IC[K_2] \cup IC[K_1, K_2]$.

The second equality comes from the fact that $C[K_1, K_3] \cup C[K_2, K_3] \cup C[K_1, K_2, K_3]$ contains all the connectors of $C[K_1 \cup K_2, K_3]$ and in addition, other sets of actions that are not maximal. By definition, IC contains all the sub-sets of C. Thus, $IC[K_1 \cup K_2, K_3] = I(C[K_1, K_3] \cup C[K_2, K_3] \cup C[K_1, K_2, K_3])$ from which we get the result by distributivity of I over union. ■

Definition 4 (Union of incomplete interactions). *Consider two sets of connectors C_1, C_2 and the corresponding sets of interactions IC_1, IC_2. We take $(IC_1)^- \cup (IC_2)^- = (IC_1 \cup IC_2)^-$.*

This definition combined with proposition 1 allows to compute the incomplete interactions of a system from the incomplete interactions of its components and thus provides a basis for incremental description of interaction models.

Property 1. For K_1, K_2, K_3 three disjoint subsets of K,

$$\begin{aligned} IC[K_1 \cup K_2]^- &= IC[K_1]^- \cup IC[K_2]^- \cup IC[K_1, K_2]^- \\ IM[K_1 \cup K_2] &= (IC[K_1 \cup K_2], IC[K_1 \cup K_2]^-) \\ &= IM[K_1] \cup IM[K_2] \cup IM[K_1, K_2] \\ IM[K_1 \cup K_2, K_3] &= IM[K_1, K_3] \cup IM[K_2, K_3] \cup IM[K_1, K_2, K_3] \end{aligned}$$

By using this property, we get the following expansion formula:

Proposition 2 (Expansion formula).

$$\begin{aligned} IM[K_1 \cup K_2 \cup K_3] = & IM[K_1] \cup IM[K_2] \cup IM[K_3] \cup IM[K_1, K_2] \\ & \cup IM[K_1, K_3] \cup IM[K_2, K_3] \cup IM[K_1, K_2, K_3] \ . \end{aligned}$$

2.3 Composition Semantics and Properties

We need the following definitions.

Definition 5 (Transition system). *A* transition system *B is a tuple $(X, IC, \{G^a\}_{a \in IC}, \{F^a\}_{a \in IC})$, where*

- *X is a finite set of variables;*
- *IC is a finite set of interactions;*
- *G^a is a predicate on $\mathbf{X}$, the set of valuations of X, called* guard*;*
- *$F^a : \mathbf{X} \to \mathbf{X}$ is a* transition function*.*

Definition 6 (Semantics of a transition system). *A transition system $(X, IC, \{G^a\}_{a \in IC}, \{F^a\}_{a \in IC})$ defines a* transition relation *$\to$: $\mathbf{X} \times IC \times \mathbf{X}$ as follows: $\forall \mathbf{x}, \mathbf{x}' \in \mathbf{X}\ \forall a \in IC\ .\ \mathbf{x} \xrightarrow{a} \mathbf{x}' \iff G^a(\mathbf{x}) \wedge \mathbf{x}' = F^a(\mathbf{x})$.*

Definition 7 (Constraint). *A* constraint *on a transition system $B = (X, IC, \{G^a\}_{a \in IC}, \{F^a\}_{a \in IC})$ is a term of the form $U^X \wedge \bigwedge_{a \in IC} t^a(U^a)$, where*

- *U^X is a state predicate on X;*
- *$t^a(U^a)$ is an action predicate such that $\forall \mathbf{x} \in \mathbf{X}\ .\ t^a(U^a)(\mathbf{x})$ if $G^a(\mathbf{x}) \Rightarrow U^a(\mathbf{x})$.*

That is, $U^X \wedge \bigwedge_{a \in IC} t^a(U^a)$ holds at states which satisfy U^X and from which only actions satisfying U^a are executable.

Definition 8 (Restriction). *The restriction of a transition system $B = (X, IC, \{(G^a)\}_{a \in IC}, \{F^a\}_{a \in IC})$ with a constraint $U = U^X \wedge \bigwedge_{a \in IC} t^a(U^a)$ is the transition system $B/U = (X, IC, \{(G^a)'\}_{a \in IC}, \{F^a\}_{a \in IC})$ where $(G^a)' = G^a \wedge U^X \wedge U^a \wedge U^X([F^a(\mathbf{x})/\mathbf{x}])$.*

A system S is described by a term $S = (B, IM)/U$ where $IM = (IC, IC^{-})$ is an interaction model, B is a transition system with set of interactions IC describing the behavior of its components, and U is a constraint.

We define the behavior of S as the transition system B'/U where B' is the transition system obtained from B by removing all its incomplete interactions, that is $B' = (X, IC, \{G^a\}_{a \in IC^+}, \{F^a\}_{a \in IC^+})$, where IC^+ is the set of the complete interactions of IM.

As in the previous sub-section, we consider that $S = (B, IM)/U$ is built from a set of interacting components K with disjoint action vocabularies A_i, $i \in K$ and behaviors described by transition systems $(X_i, A_i, \{G^a\}_{a \in A_i}, \{F^a\}_{a \in A_i})$ with disjoint sets of variables.

We denote by $S[K]$ the system built from the components $k \in K$, and assume that it has the layered structure $S[K] = (B[K], IM[K])/U$ with interaction model $IM[K]$ and constraint U.

We define a composition operator $\|$ allowing to obtain for disjoint sub-sets K_1, K_2 of K, the system $S[K_1 \cup K_2]$ as the composition of the sub-systems $S[K_1]$, $S[K_2]$ for given interaction model $IM[K_1, K_2]$ connecting the two sub-systems. The operator composes separately behaviors, interaction models and restrictions of the sub-systems.

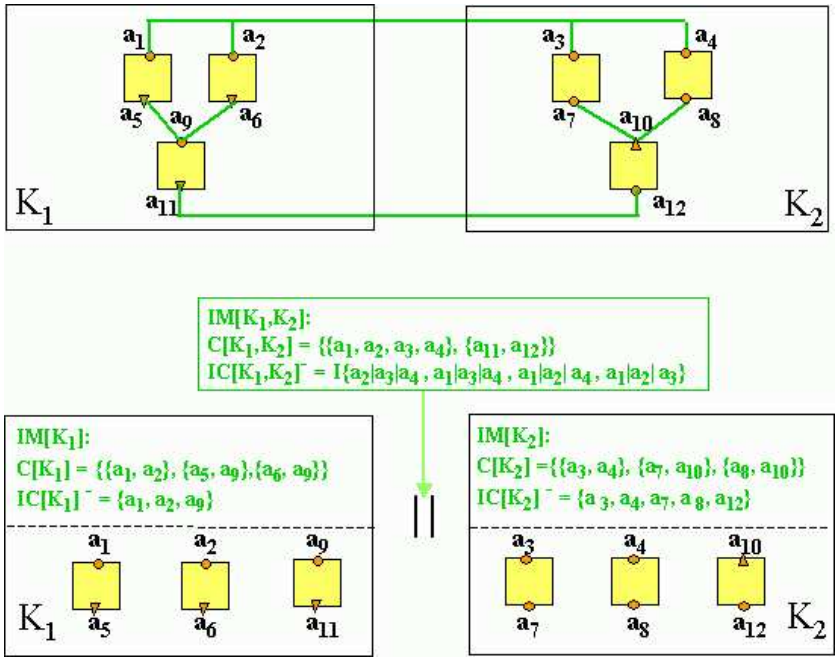

Fig. 4: The composition principle.

Definition 9 ($\|$). *The composition of two systems $S[K_1]$ and $S[K_2]$ is the system*

$$\begin{aligned} S[K_1 \cup K_2] =& (B[K_1], IM[K_1])/U_1 \parallel (B[K_2], IM[K_2])/U_2 \\ =& (B[K_1] \times B[K_2], IM[K_1] \cup IM[K_2] \cup IM[K_1, K_2])/(U_1 \wedge U_2) \end{aligned}$$

where $\times$ is a binary associative behavior composition operator such that $B[K_1] \times B[K_2] = (X_1 \cup X_2, IC[K_1 \cup K_2], \{G^a\}_{a \in IC[K_1 \cup K_2]}, \{F^a\}_{a \in IC[K_1 \cup K_2]})$ where for $\alpha = \alpha_1 \shortmid \alpha_2 \in IC[K_1, K_2]$, $G^\alpha = G^{\alpha_1} \wedge G^{\alpha_2}$ and for any valuation $(\mathbf{x}_1, \mathbf{x}_2)$ of $X_1 \cup X_2$, $F^\alpha(\mathbf{x}_1, \mathbf{x}_2) = (F^{\alpha_1}(\mathbf{x}_1), F^{\alpha_2}(\mathbf{x}_2))$.

Due to property 1 we have $(B[K_1], IM[K_1])/U_1 \parallel (B[K_2], IM[K_2])/U_2 = (B[K_1 \cup K_2], IM[K_1 \cup K_2])/(U_1 \wedge U_2)$, which means that composition of subsystems gives the system corresponding to the union of their components.

Notice that as $\times$ is an associative operator, composition is associative:

$$\begin{aligned} &\big((B[K_1], IM[K_1])/U_1 \| (B[K_2], IM[K_2])/U_2\big) \| (B[K_3], IM[K_3])/U_3 = \\ &= (B[K_1 \cup K_2], IM[K_1 \cup K_2])/(U_1 \wedge U_2) \| (B[K_3], IM[K_3])/U_3 \\ &= (B[K_1] \times B[K_2] \times B[K_3], IM[K_1 \cup K_2] \cup IM[K_3] \cup IM[K_1 \cup K_2, K_3])/ \\ &\quad (U_1 \wedge U_2 \wedge U_3) \\ &= (B[K_1 \cup K_2 \cup K_3], IM[K_1 \cup K_2 \cup K_3])/(U_1 \wedge U_2 \wedge U_3) \end{aligned}$$

by application of proposition 2. Thus we have the proposition

Proposition 3. *$\|$ is a commutative and associative operator on systems.*

3 Deadlock Freedom by Construction

3.1 Global Deadlock Freedom

Definition 10 (Invariant). *A constraint $U = U^X \wedge \bigwedge_{a \in IC} t^a(U^a)$ on $B = (X, IC, \{G^a\}_{a \in IC}, \{F^a\}_{a \in IC})$ is an* invariant *if $\forall a \in IC\ \forall \mathbf{x} \in \mathbf{X}\ .\ U^X(\mathbf{x}) \wedge G^a(\mathbf{x}) \Rightarrow U^X([F^a(\mathbf{x})/\mathbf{x}]) \wedge U^a(\mathbf{x})$.*

The following properties are easy to prove and relate restriction to invariants.

Property 2. Given a transition system B and constraints U, U_1, U_2,

- U is an invariant of B/U;
- if U_i is an invariant of B_i, $i = 1, 2$, then $U_1 \wedge U_2$ is an invariant of $(B_1 \times B_2, IM)$ for any IM, and of $(B_1 \times B_2, IM)/U$;
- $(B/U_1)/U_2 = B/(U_1 \wedge U_2)$;
- if U is an invariant of B then B/U is bisimilar to B from any state satisfying U.

As in the previous section, consider a system $S = (\times_{k\in K} B_k, IM)/U$ built from a set of interacting components K where the transition systems $B_k = (X_k, A_k, \{G^a\}_{a\in A_k}, \{F^a\}_{a\in A_k})$ have disjoint action vocabularies and sets of variables. We assume that $IM = (IC, IC^+)$, and $B = (\times_{k\in K} B_k, IM) = (X, IC^+, \{G^a\}_{a\in IC^+}, \{F^a\}_{a\in IC^+})$. In this section we study deadlock-freedom of $S = B/U$ and of its components.

Let $B/U = (X, IC^+, \{(G^a)'\}_{a\in IC^+}, \{F^a\}_{a\in IC^+})$ be the restriction of B by some constraint $U = U^X \wedge \bigwedge_{a\in A} t^a(U^a)$ with restricted guards $(G^a)' = G^a \wedge U^X \wedge U^a \wedge U^X([F^a(\mathbf{x})/\mathbf{x}])$ for any interaction $a \in IC^+$.

Definition 11 (Complete states). *Given $S = (B, IM)$, $U = U^X \wedge \bigwedge_{a\in A} t^a(U^a)$, and k some component of B, the set of* complete states *of k, that is, states of k from which its progression cannot be prevented by its environment, is characterized by the largest predicate complete(k) on X_k such that complete$(k) \implies dlf_k \wedge \bigvee_{\substack{\alpha\in IC^+ \\ \alpha\cap A_k\neq\emptyset}} (G^\alpha)'$.*

Definition 12 (Blocking states). *For $k \in K$, let dlf_k be some non-empty invariant on $\mathbf{X}_k$ implying deadlock-freedom of k, that is, $dlf_k \Rightarrow \bigvee_{a\in A_k} G^a$. We take $dlf = \bigwedge_{k\in K} dlf_k$ and define for a component k the predicate*

$$blocking(k) = dlf_k \wedge \neg \bigvee_{\substack{\alpha\in IC^+ \\ \alpha\cap A_k\neq\emptyset}} (G^\alpha)'$$

characterizing the states where k is blocked due to interaction or restriction with U.

Definition 13 (Dependency graph). *Consider a system $S = (B, IM)/U$ built from a set K of components. For each component k put each predicate blocking(k) in the form $\bigvee_{i\in I_k} c_i$ with $c_i = \bigwedge_{k'\in K} D_{k',i}$ where $D_{k',i}$ is a predicate depending on $X_{k'}$. The dependency graph of S is a labelled bipartite graph with two sets of nodes: the components of K, and constraint nodes $\{c_i \mid \exists k \in K \,.\, i \in I_k\}$, where I_k is the set of conjunctive terms occurring in blocking(k). For a component node k and a constraint node c_i,*

- *there is an edge from k to c_i labeled with $D_{k,i}$ if $D_{k,i} \neq false$; and*
- *there is an edge from c_i to k labeled with $D_{k,i}$ if $i \in I_k$.*

Notice that constraint nodes represent conditions under which a component is blocked. If c is a constraint node for a component k then it has at least one incoming and one outcoming edge. At some state, k is blocked if all the predicates labelling the incoming edges of c are true.

Let γ be a circuit of the dependency graph of a system $(B, IM)/U$. The predicate

$$DL(\gamma) = \bigwedge_{(k \xrightarrow{D} c \xrightarrow{D'} k')\in\gamma} \big(D \wedge D'\big)$$

characterizes the system states for which all components in γ may be blocked cyclically awaiting for each other.

Theorem 1 (Deadlock freedom). *A system* $(B, IM)/U$ *is* deadlock-free *from any state satisfying* $dlf \wedge U^X$ *if its dependency graph contains a non-empty sub-graph* G *such that*

- *if* G *contains a component node then* G *contains all its predecessor constraint nodes, and if* G *contains a constraint node then* G *contains one of its predecessors; and*
- *for any elementary circuit* γ *of* G, $DL(\gamma) = false$.

Proof. Assume that the system is at some global deadlock state $\mathbf{x}$. As all the components are initialised at states satisfying *dlf* there exists for each component at least one action a having its guard G^a enabled at this state in B.

Consider a component k of G and a term c_i of $blocking(k)$ such that $c_i(\mathbf{x})$. As $dlf_k(\mathbf{x})$, the label of the input edge (c_i, k) is true at this state (the guards are contained in dlf_k). Then consider some predecessor k' of c_i in G. The label of the edge (k', c_i) is true as it is a factor of c_i. Again, the component k' has at least one guard enabled. Move backwards in G from this node by iterating the process. Then an elementary circuit γ of G is found. By construction, all the predicates labelling the edges of this circuit are true at state $\mathbf{x}$. This contradicts the assumption of $DL(\gamma) = false$. ■

Theorem 2 (Deadlock freedom). $(B, IM)/U$ *is* deadlock-free *from* $dlf \wedge U^X$ *if* $dlf \wedge U^X \Rightarrow \bigvee_{\alpha \in IC^+} (G^\alpha)'$.

Proof. Since for any $k \in K$, dlf_k is invariant by hypothesis, *dlf* and $dlf \wedge U^X$ are by property 2 invariants of $(B, IM)/U$. As $dlf \wedge U^X$ implies enabledness of some complete interaction, its invariance amounts to deadlock-freedom. ■

Theorem 3. *The conditions of theorem 1 imply that* $dlf \wedge U^X \Rightarrow \bigvee_{\alpha \in IC^+} (G^\alpha)'$.

Proof (sketch). Suppose that there is some non-empty sub-graph G of the dependency graph as specified in theorem 1 such that for any elementary circuit γ of G, $DL(\gamma) = false$. Thus, for any valuation $\mathbf{x}$ such that $dlf(\mathbf{x})$ there is some component k in G with $\neg blocking(k)(\mathbf{x})$ by construction of the dependency graph. By definition of *blocking*, it follows that there is some interaction α such that $(G^\alpha)'(\mathbf{x})$.

The condition of theorem 2 allows an efficient check for deadlock freedom. However, in order to provide a diagnostic when this condition is not verified, it may be useful to construct a dependency graph for the states satisfying $dlf \wedge \neg \bigvee_{\alpha \in IC^+} (G^\alpha)'$.

3.2 Individual Deadlock-Freedom

We give some results about deadlock-freedom preservation for transitions systems. Similar results have been obtained for timed transition systems with priorities in [5].

In general, deadlock-freedom or even liveness of a system of interacting components do not imply that any component remains deadlock-free in the system. Guaranteeing this stronger property is the objective of the following definitions and theorem.

Definition 14 (Run). *A* run *of a transition system* $B = (X, IC, \{G^a\}_{a\in IC}, \{F^a\}_{a\in IC})$ *is an infinite sequence of interactions* $\mathbf{x}_0 \xrightarrow{\alpha_0} \mathbf{x}_1 \xrightarrow{\alpha_1} \dots \mathbf{x}_n \xrightarrow{\alpha_n} \dots$.

Definition 15 (Individual deadlock-freedom). *Given a system* S*, a component* $k \in K$ *is* deadlock-free *in* S *if for any run* σ *of* S *and any prefix* σ_n *of* σ*, there exists a run* σ' *such that* $\sigma_n\sigma'$ *is a run of* S*, and some interaction of* σ' *contains an interaction of* k*.*

Definition 16 (Controllable predecessors). *Let* $B = (X, IC, \{G^a\}_{a\in IC}, \{F^a\}_{a\in IC})$ *be the behavior of a component. For* $Y \subseteq \mathbf{X}$*, define* $pre(Y) \subseteq \mathbf{X}$ *such that* $\mathbf{x} \in pre(Y)$ *if*

- *if* $\mathbf{x}$ *is complete then* $\exists \mathbf{x}' \in \mathbf{X}\ \exists a \in IC^+ \,.\, \mathbf{x} \xrightarrow{a} \mathbf{x}' \ \wedge\ \mathbf{x}' \in Y$*;*
- *if* $\mathbf{x}$ *is incomplete then* $\forall \mathbf{x}' \in \mathbf{X}\ \forall a \in IC^- \,.\, \mathbf{x} \xrightarrow{a} \mathbf{x}' \Rightarrow \mathbf{x}' \in Y$*, and such* a *and* $\mathbf{x}'$ *exist.*

For $\mathbf{X}_0 \subseteq \mathbf{X}$ *we denote by* $PRE(\mathbf{X}_0)$ *the least solution of* $Y = \mathbf{X}_0 \cup pre(Y)$*.*

Clearly, $PRE(\mathbf{X}_0)$ exists, as it is the fixed point of a monotonic functional. $PRE(\mathbf{X}_0)$ represents the set of the predecessors of $\mathbf{X}_0$ in the transition graph such that from anyone of its states a state of $\mathbf{X}_0$ can be reached by appropriately choosing complete interactions. In this context, complete interactions can be characterized as controllable, as when they are enabled some interaction containing them can occur in the product system. On the contrary incomplete interactions are uncontrollable as their occurrence in the product depends on the state of the environment. Predicate transformers taking into account controllability have been studied in [13].

Definition 17 (Controllability). *Given a system* S*, we call a component* $k \in K$ *with behavior* $(X_k, A_k, \{G^a\}_{a\in A_k}, \{F^a\}_{a\in A_k})$ controllable *with respect to some state constraint* U *on* X_k *if* $PRE(U) = X_k$*.*

Theorem 4 (Individual Deadlock-Freedom). *Given a system* $S = (B, IM)$ *built from a set* K *of components, a component* $k \in K$ *is deadlock-free from dlf in* S *if*

- *the dependency graph of* S *contains a sub-graph* G *satisfying the requirements of theorem 1 with* $k \in K'$*, where* K' *is the set of component nodes in* G*,*

– *any* $k \in K'$ *is controllable with respect to* G^a *for any* $a \in A_k$ *such that* $\exists \alpha \in IC\, \exists \alpha' \in IC^{-}\ .\ a \in \alpha\ \wedge\ \alpha' \subseteq \alpha$, *and*
– *all* n*-ary interactions with* $n \geqslant 3$ *are rendez-vous, that is, for any interactions* $\alpha \in IC^{+}$ *with* $|\alpha| \geqslant 3$ *and* $\alpha' \in IC$, *we have (1) if* $\alpha \cap \alpha' \neq \emptyset$, *then* $\alpha' \in IC^{-}$, *and (2) if* $a \in \alpha \cap A_i$ *and* $a' \in \alpha' \cap A_i$, *then* $G^a \wedge G^{a'} = false$ *for any* $i \in K$.

Notice that under the hypotheses above, any component that is blocked, is either waiting for one (or more) binary interaction, or for exactly one n-ary interaction with $n \geqslant 3$.

Proof. Consider some product state $\mathbf{x}$ in which k is blocked, and no complete interaction involving k is enabled. By theorem 1, some (direct or transitive) predecessor of k can progress. Let $k_{i_1} \overset{\alpha_1}{\dashrightarrow} k_{i_2} \overset{\alpha_2}{\dashrightarrow} \ldots \overset{\alpha_{n-1}=\alpha}{\dashrightarrow} k_{i_n} = k$ be a chain of components in K' where $k_i \overset{\alpha}{\dashrightarrow} k_j$ means that k_j is in an incomplete state waiting for interaction α with k_i, and such that only k_{i_1} is able to progress. By controllability, k_{i_1} can be led (by appropriately choosing some complete action, or by any incomplete action) towards a state $\mathbf{x}'$ where its action participating in α_1 is enabled. If α_1 is binary then this unblocks k_{i_2}; otherwise the interaction is by hypothesis a rendez-vous, and both components k_{i_1} and k_{i_2} remain blocked. In that case we apply the same reasoning to any chain of components blocking k_{i_2}. Finally, α_2 will be enabled. The same reasoning can now be applied to k_{i_3} and recursively descending the chain, until α becomes enabled. ■

4 Discussion

This work is in progress and needs further validation by examples and case studies. It pursues similar objectives as the work by Th. Henzinger and his colleagues [6]. It lies within the scope of a lasting research program.

The paper presents compositionality and composability results for deadlock-from of systems built from components following the proposed construction methodology. The concept of component is very general and can be applied to various types of descriptions, e.g. a block of code, hardware, provided they have disjoint state spaces and well defined interface and behavior. Interaction models provide a powerful framework for synchronization, that encompasses both strict and non strict synchronization. Restriction appears to be a very useful concept for imposing global invariants. The layered description principle allows separation of concerns and has been used to some extent in [1]. Furthermore, layered structuring is instrumental for the definition of an associative composition operation.

The provided sufficient deadlock-freedom conditions require that components satisfy specific properties such as existence of non trivial deadlock-free invariants, and of sets of controllable predecessors. These properties can be checked algorithmically only when the components are finite-state. Deadlock-freedom conditions also require the computation of the guards of the constructed system, which may be a source of exponential explosion for general interaction models and restrictions.

References

1. K. Altisen, G. Gössler, and J. Sifakis. Scheduler modeling based on the controller synthesis paradigm. *Journal of Real-Time Systems, special issue on "control-theoretical approaches to real-time computing"*, 23(1/2):55–84, 2002.
2. A. Benveniste, P. LeGuernic, and Ch. Jacquemot. Synchronous programming with events and relations: the SIGNAL language and its semantics. *Science of Computer Programming*, 16:103–149, 1991.
3. J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *TCS*, 37(1):77–121, 1985.
4. G. Berry and G. Gonthier. The ESTEREL synchronous programming language: Design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, 1992.
5. S. Bornot, G. Gössler, and J. Sifakis. On the construction of live timed systems. In S. Graf and M. Schwartzbach, editors, *Proc. TACAS'00*, volume 1785 of *LNCS*, pages 109–126. Springer-Verlag, 2000.
6. L. de Alfaro and T.A. Henzinger. Interface theories for component-based design. In T.A. Henzinger and C. M. Kirsch, editors, *Proc. EMSOFT'01*, volume 2211 of *LNCS*, pages 148–165. Springer-Verlag, 2001.
7. E.W. Dijskstra. *A Discipline of Programming*. Prentice Hall, 1976.
8. G. Goessler and J. Sifakis. Composition for component-based modeling. In *proc. FMCO'02 (to appear)*, LNCS. Springer-Verlag, 2003.
9. N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous dataflow programming language LUSTRE. *Proceedings of the IEEE*, 79(9):1305–1320, September 1991.
10. T.A. Henzinger, B. Horowitz, and C. Meyer Kirsch. Embedded control systems development with Giotto. In *Proc. LCTES'01*, 2001.
11. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.
12. Hermann Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer, 1997.
13. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In E.W. Mayr and C. Puech, editors, *STACS'95*, volume 900 of *LNCS*, pages 229–242. Springer-Verlag, 1995.
14. F. Maraninchi. Operational and compositional semantics of synchronous automaton compositions. In *proc. CONCUR*, volume 630 of *LNCS*. Springer-Verlag, 1992.
15. R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267–310, 1983.
16. J. Misra and K.M. Chandy. *Parallel Program Design: A Foundation*. Addison-Wesley, 1988.
17. K.J. Turner, editor. *Using Formal Description Techniques — An Introduction to Estelle, LOTOS and SDL*. Wiley, 1993.