Contracts for BIP: hierarchical interaction models for compositional verification*

Susanne Graf and Sophie Quinton

Verimag/CNRS and Verimag/ENS Cachan

1 Introduction

Our aim is to provide a contract-based verification framework. A system is a hierarchically structured set of *components* – which we call *Heterogeneous Rich Components* or HRC. We extend the component framework BIP [GS05,BBS06] and in particular it instance based on hierarchical connectors [BS07] to a framework for hierarchical components enriched with contracts.

In the BIP framework, components interact through ports typed by *trig* or *sync* and are connected via hierarchical n-ary connectors which are typed in the same way as ports. In BIP only connectors are hierarchical but we consider here also a hierarchical organisation of the components. Like in BIP, only leaf components represent models and have a behaviour explicitly defined by a transition system. Originally, in BIP atomic components have a sequential behaviour, but here, atomic components are not different from hirarchically defined components, at least frome outside, therfore we eliminate this restriction. Here, we represent component behaviour by an asynchronous transitions system, but we may choose other more convenient representations in the future.

The behaviour of hierarchically defined components is obtained naturally as a composition of the behaviours of its leaf components depending on its internal connectors.

An HRC K has in addition contracts, in the form of an assumption A and a guarantee G, represented by transition systems. A defines a property of the environment of K and G defines a property of K in the case that K runs in an environment guaranteeing A.

The purpose of our work is to define a framework for validating

- that the behaviour of a component satisfies its contracts
- that the contracts associated with components that are higher in the contract hierarchy dominate the contracts associated with its inner components

We want to do the validation compositionally and efficiently. Presently the verification condition that we provide involves a component and all its subcomponents which may not be alsways feasible. Many heuristics have been proposed for decomposing the verification condition that we obtain, but when non functional properties are involved, these heuristics tend to be useless.

* This work has been partially financed by the project SPEEDS and the NoE Artist

As building a set of contracts allowing such a strictly compositional approach is generally very difficult, we also consider a more flexible approach where the strictly contract based verification approach may be replaced by more global but abstract verification and by providing methods for obtaining from local error traces global error traces and sufficient conditions to conclude global error freedom from local one.

In Section 2, we define the syntactic framework of hierarchical components and connectors. We define the semantics in two steps. First, we say how to obtain a transition system defining the behaviour of a hierarchical component from the transition systems of its subcomponents and the connectors between them.

The BIP framework allows both for asynchronous and for synchronous interaction and execution. In particular, blocking rendez-vous is possible means that badly designed systems may have deadlocks or interlocks. Verification of absence of deadlock is an important issue both for guaranteeing absence of deadlocks in system specifications and as a means for property verification. In the conext of embedded real-time systems is interaction freedom between transactions. Here we define the building blocks for addressing this problem in an efficient manner, but the definition of specialised framework for this purpose is forthcoming work. Interactions as they are defined in BIP can be seen as basic transactions which can then be composed to larger transactions. Our framework allows guarantee by construction absence of interaction, but we have to verify absence of deadlocks.

Then, as we are interested in linear safety properties mainly, and as deadlock is issue, we define the properder relation on component behaviours and the satisfaction relation in terms of traces and refusals of properties and components.

In Section 3 we describe how we intend to verify the consistency of an hierarchy of contracts.

We propose an approach to contract verification exploiting the fact that assumptions are parallel compositions (conjunctions). In order to show that a contract (A, G) dominates a composition of contracts $\{A_i, G_i)\}$ — representing a set of contracts associated with the subcomponents of C — it is sufficient to show that

- $A\|G_1\|....\|G_n \models G$; this shows that every component K_i ensuring the individual guarantees, guarantees also G at least if the environment behaves according to A
- in addition, we have to show verification conditions of the form $A\|G_1\|...\|G_n \models A_i$; this shows that each of the assumptions A_i can be discharged by the environment of K_i, so as to guarantee that the restriction imposed by A_i is not too strong

This proof rule is correct in our framework as A and G constrain different components

Finally, in Section 4 we give a first idea on how we intend to achieve an efficient and scalable handling of contracts.

2 Specifications and their semantics

Definition 1 (Interaction set). *Let Σ be a set and $<, \# \subseteq \Sigma \times \Sigma$ be a pre-order and a conflict relation. Then $(\Sigma, <, \#)$, sometimes simply denoted Σ is an interaction set if the following conditions hold:*

- *$<$ is a partial order relation;*
- *$\#$ is a non reflexive and symmetric conflict relation such that $a\#b$ and $a < c$ implies $c\#b$.*

For $a \in \Sigma$, we denote by $\uparrow a = \{b \in \Sigma \mid a < b\}$ the upwards closure of a and by $\downarrow a = \{b \in \Sigma \mid b < a\}$ the downwards closure of a and we extend these notions to sets.

Denote $a \sqcup b$ the $c \in \Sigma$ representing a least upper bound of a and b, if it exists. Define the closure of Σ, $cl(\Sigma)$ the interaction set obtained by recursively adding an element denoted $a \sqcup b$, whenever $a, b \in \Sigma$, not $a\#b$ and not yet exists an element c representing the upper bound of a and b.

Note that it is not excluded that both $a\#b$ even if $a \sqcup b$ exists. Interactions for which $a \sqcup b$ exists can be connected, but only interactions for which not $a\#b$ can be executed concurrently.

Definition 2 (Interaction model). *An interaction set $(\Sigma, <, \#)$ is an* interaction model *if whenever a and b are not in conflict, that is not $a\#b$, then there exists an action $c \in \Sigma$ that is a least upper bound of a and b.*

Property 1. For every interaction set Σ, the closure $cl(\Sigma)$ is an interaction model.

Here, an interaction set can can alsways be made an interaction model by either adding by adding a conflict $a\#b$ or the element of the closure $a \sqcup b$ whenever there is no least upper bound of a and b in Σ. In particular, a union of interaction models is an interaction set. The product of interaction models, denoted $\Sigma_1 \cdot \Sigma_2 \ni a_1 \cdot a_2$ or $\Pi_i \Sigma_i \ni (a_1, ..., a_n)$, is already an interaction model.

The interaction models as they are defined in [GS05] are interaction models when defining $a\#b$ whenever $a \sqcup b$ is not defined. Here, we allow avoiding the definition of an explicit interaction $a \cdot b$ when a and b are independent, that is when not $a\#b$; indeed, such interactions are implicitly captured by $a \sqcup b$ in $(cl(\Sigma), <, \#)$.

Definition 3 (Ports and component interfaces). *A* port *is defined as in [BS07] by a name and a type either trigger or synchron.*

Let $\mathcal{P}$ be a set of (typed) ports. A (external) component interface *Int is defined by a set of typed ports $\mathcal{P}$ and two relations $<$ and $\#$ on $\mathcal{P}$ such that $\mathcal{P}$ is an interaction set and consequently $cl(\mathcal{P})$ an interaction model.*

Definition 4 (Component). *An component K is defined by $K = ((\mathcal{P}, <, \#), TS)$ where $TS = (Q, q_0, cl(\mathcal{P}), \rightarrow)$ is a transition system defined on the interaction model $cl(\mathcal{P})$, such that*

- *Q is a set of states and $q_0 \subseteq Q$ an initial state (or a set of initial states).*
- *$\rightarrow \subseteq Q \times cl(\mathcal{P}) \times Q$ is a transition relation. We define $en(a)$ for each $a \in \Sigma$ as the set of states in which a is enabled, that is $\exists q' \in Q \,.\, q \xrightarrow{a} q'$. $\rightarrow$, and we extend the transition relation in the usual way to a transition relation on sequences of labels. $\rightarrow$ must then satisfy the following conditions:*
  - *if $a, b \in cl(\mathcal{P})$, then $q \in en(a) \cap en(b)$ implies ($q \xrightarrow{a;b} q''$ iff $q \xrightarrow{b;a} q''$) and whenever $q \xrightarrow{b} q''$ then also $q \in en(a) \cap en(b)$ or $q \in en(a \sqcup b)$;*
  - *if $a, b, c = a \sqcup b \in \mathcal{P}$ then, $en(a) \wedge en(b) = en(c)$ and $a < b$ implies $en(a) \subseteq en(b)$*
  - *for interactions $a, b \in cl(IS)$, $a < b$ but $\exists p \in \mathcal{P} \,.\, a = \sqcup act(p)$, then $en(b) \subseteq en(a)$.*

We call $(\mathcal{P}, <, \#) = Int(K)$ the (external) interface of K and $TS = beh(K)$ the behaviour of K

That means that whenever ports are independent (not in conflict), then in a state in which both a and b are enabled, the transition sequence $\xrightarrow{a;b}$ is semantically equivalent to $\xrightarrow{b;a}$ and to $\xrightarrow{a \sqcup b}$, where the latter may or may not explicitly exist in TS independently of the fact that a port dominating a and b exists or not. In fact, the allowed labels of $\rightarrow$ are those in $cl(\mathcal{P})$. This means that $a; b$ and $a \sqcup b$ are in such states semantically equivalent and TS represents an asynchronous transition as defined in [WN95].

We now define hierarchical components as compositions of components and we define 2 views of hierarchical compositions:

- an *external view* which represents an hierarchical component to the environment exactly as an atomic component, defined by its external interface, that is its set of ports and a behaviour represented by transitions labelled by elements of of the interaction model $cl(\mathcal{P})$.
- an *internal view* which makes visible the internal *structure* of the component, consisting of a set of components K_i and a composed interaction model IM which is a subset of $\bigcup_{i \in I} \mathcal{P}_i \bigcup_{j \in J \subseteq I} \Pi_{j \in J,k} \mathcal{P}_j^k$ and which will be defined by a set of hierarchical connectors as in [BS07].
- the internal and external view are linked via a relation $\dashv$ associating subsets of IM to an interaction set $Int = (\mathcal{P}, <, \#)$ such that $\dashv$ is a structure preserving relation from the interaction model IM to the one defined by Int, that is $(cl(\mathcal{P}), <, \#)$.
- that means that the behaviour of a hierarchical component is defined in the internal view in terms of transitions labelled by IM and in the external view by transitions labelled in $cl(\mathcal{P})$ such that in the behaviour of K, each internal view transition $q \xrightarrow{\sigma} q'$ corresponds to a set of transitions $q \xrightarrow{p} q'$ for each p such that $\sigma \dashv p$

We first define the internal view of a hierarchical component by defining the notion of typed hierarchical connector which then allow to define the internal view of a hierarchical component.

Definition 5 (Connector). *Let* $\{Int_i = (\mathcal{P}_i, <_i, \#_i)\}$ *be a set of (external) component interfaces. A* typed connector *con on* $\{Int_i\}$ *consists of:*

- *a subset* $con \in \bigcup_i \mathcal{P}_i$ *such that* $\forall i, \forall p_m, p_n \in con \,.\, p_m \sqcup_i p_n$ *is defined in the union interaction model induced by* $\{Int_i\}$ *but* $p_m \sqcup_i p_n \notin \mathcal{P}_i$. *(meaning that either interactions of* p_n *and* p_m *are independent, or if* $p_m \#_i p_n$, *then* $p_m \sqcup_i p_n$ *has been already defined somewhere "inside" the hierarchical component* K_i *exposing the external interface* $\{Int_i\}$, *but the connector corresponding to* $p_m \sqcup_i p_n$ *does not appear in the interface* Int_i*).*
- *a type* sync *or* trig.

con defines an interaction set Σ_{con} *as a subset of* $\bigcup_{i \in I} \mathcal{P}_i \bigcup_{j \in J \subseteq I} \Pi_{j \in J,k} \mathcal{P}_j^k$ *where* Σ_{con} *contains*

- $p \in \mathcal{P}_i$ *for any* i
- $p_1 \cdot \ldots \cdot p_n$ *such that* $con = \bigcup_{l=1..n} p_i$ *and* $\forall p_l \in con, type(p_l) =$ sync *(that is all ports must synchronise)*
- *if con contains ports of type* trig *then,* $p_1 \cdot \ldots \cdot p_m$ *such that* $\bigcup_{l=1..m} p_i \subseteq con$ *and* $\exists p_l \,.\, type(p_l) =$ trig*(that is the interaction* $p_1 \cdot \ldots \cdot p_m$ *is complete)*

The preorder relation $<$ *is derived from* $<_i$ *and* $p < p \cdot q$. *The conflict relation is defined* $\bigcup_i \#_i$ *and* $\sigma \# \sigma'$ *if there definitions involve conflicting ports. That means in particular that in a hierarchical definition* $\sigma \# \sigma'$ *if they do not "hierachically contain" a common or conflicting atomic ports.*

Definition 6 (Mapping of an interaction model on a component interface). *Let* $(\Sigma, <, \#)$ *be an interaction set defined upon a set of atomic ports and* $Int = (\mathcal{P}, <, \#)$ *a component interface.*

A relation $\dashv \subseteq \Sigma \times \mathcal{P}$ *(or as well a function* $act(p) = \{a \in \Sigma . a \dashv p\}$*) defines an interaction port association between* Σ *and* $\mathcal{P}$ *if:*

- *for each* $p \in \mathcal{P}$, $act(p)$ *is an interaction model such that* $act(p) = \downarrow a$ *for some element in* Σ*; sometimes we identify* p *by this maximal element of* $act(p)$
- $p < r$ *iff* $act(p) \subseteq act(r)$
- $p \# r$ *iff* $\exists a \in act(p) \exists b \in act(r)$ *such that* $a \# b$
- *if* $a \dashv p$ *and* a *is an atomic port, then* $type(a) = type(p)$

That means, an interface $Int = (\mathcal{P}, <, \#)$ is an interaction set derived from an interaction set Σ by exposing some downwards-closed subsets of Σ as an interaction p to the environment. In fact, such a downwards closed set represents exactly the interaction model of a BIP *connector*, and also here, a port representats either an atomic port or a connector on a set of components.

Definition 7 (Internal interface or composition model). *Let* $\{Int_i = (\mathcal{P}_i, <_i, \#_i)\}$ *be a set of (external) component interfaces, let be* CON *a set of connectors defined on* $\{Int_i\}$ *and* IS *the interaction set defined by* $\{Int_i\}$ *and* CON *as in definition 5. Then.*

The internal interface *or* composition model *of a hierachical component defined by* $\{Int_i\}$ *and* CON *is* $CM = ((\mathcal{P}, <, \#), IS, \dashv)$ *if*

- $\mathcal{P} = \bigcup_i \mathcal{P}_i \cup \{p_{con}, con \in CON\}$
- $\dashv$ *is an interaction port association between* IS *and* $\mathcal{P}$ *defined by* $a \dashv p$ *if* $a \in \mathcal{P}_i$ *and* $a = p$ *or* $a \dashv p$ *if* $p = p_{con}$ *and* $a \in \Sigma_{con}$ *as defined in definition 5.*

Now we provide the possibility to define connectors hierarchically and thus define more complex component compositions. Note that hierarchical connectors are needed to be able to express any interaction set IS on a set $\{Int_i\}$ as a composed components (this is proved in [BS07]).

Definition 8 (Hierarchical connectors). *Let be* $CM = ((\mathcal{P}, <, \#), IS, \dashv)$ *the composition model defined by the set* $\{Int_i = (\mathcal{P}_i, <_i, \#_i)\}$ *of component interfaces and the set* CON *of (hierarchical) connectors. A* (typed) hierarchical connector *con is defined in the same way as a connector by:*

- *a subset of* $\mathcal{P}$ *such that* $\forall i, \forall p_m, p_n \in con\ p_m \sqcup p_n$ *is defined but not* $p_m \sqcup p_n \in \mathcal{P}$*;*
- *a type* sync *or* trig.

con defines an interaction set Σ *on* $\bigcup_i \mathcal{P}_i$ *which is defined analogously to that of a connector, only that ports that are connectors in* CON *are instantiated by their interaction set in* IS*. That is the hierarchical definition of elements in* $\mathcal{P}_i$ *is hidden but the one of elements in* CON *is not.*

Thus, Σ_{con} *is defined by :*

- *if* $\forall p \in con, type(p) =$ sync *(that is all ports must synchronise), then* $a_1 \cdot ... \cdot a_n$ *such that* $con = \bigcup_{l=1..n} p_i$ *and either* $a_i = p_i \in \mathcal{P}_i$ *or* $a_i \dashv p_i$ *and* $p_i \in CON$
- *if con contains ports of type* trig *then,* $a_1 \cdot ... \cdot a_m$ *such that* $\bigcup_{l=1..m} p_i \subseteq con$, $\exists p_l \,.\, type(p_l) =$ trig*(that is the interaction* $p_1 \cdot ... \cdot p_m$ *is complete) and again either* $a_i = p_i \in \mathcal{P}_i$ *or* $a_i \dashv p_i$ *and* $p_i \in CON$

The preorder relation $<$ *and the conflict relation* $\#$ *are defined exactly in the same way as for a connector in definition 5.*

Also the internal interface $CM' = ((\mathcal{P} \cup \{p_{con}\}, <, \#), IS \cup \Sigma_{con}, \dashv \cup \dashv_{con})$ *is defined in the same way as before*

Thus the componosition model $CM = ((\mathcal{P}, <, \#), IS, \dashv)$ defined by a set of hierarchical connectors adds all the connectors as new ports and expresses all interactions as set of the ports of the connected interfaces. Now, we enclose such a composition model into a component by (1) choosing any subset of $\mathcal{P}$ to be exposed for connection with the environment and (2) by defining an external interface that hides the interactions in IS by substituting them by interactions in the subset of externally visible ports.

In a constructive approach one may choose to keep all ports and connectors available for further composition, whereas in a projective approach, for a given a global system architecture, it is enough to expose only those ports which are indeed used in some connection at some level of hierarchy.

Definition 9 (Hierarchical and atomic component). *Let K_i be a set of components $Int_i = (\mathcal{P}_i, <_i, \#_i)$ be a set of component interfaces and $CM = ((\mathcal{P}, <, \#), IS, \dashv)$ a composition model for the set $\{Int_i\}$. Then, for a ny choice of a set $\mathcal{P}' \subseteq Ports$, a hierarchical component K is defined by $K = (K_i, CM, Int)$*

- *its subcomponents K_i*
- *its internal interface CM, where $(\{K_i\}, CM)$ is sometimes referred to as the internal structure of K*
- *an external interface $Int = (Ports', <, \#)$ which is derived from CM in a straightforward manner. We sometimes call $str(K) = (str(K_i), CM, Int, \dashv)$ the structure of K*
- *The behaviour of K, $beh(K)$ is defined from the $beh(K_i)$ by composing them according to CM. the behaviour expressed in terms of actions in IS of CM is the internal view of the behaviour and the one obtained by transforming labels in IS by labels in P is the external view of the behaviour. We define next how the behaviour of K is defined as a composition of behaviours of K_i.*
- *$(Int, beh(K))$ is the component defined by K*

A component K is an atomic component if it has no subcomponents or if its internal structure is not known.

Note that we do not require that K provides an explicit transition system corresponding to this behaviour. It is enough to provide the transition systems of the leave components and obtain the composed transitions hierarchical composition using the composition models CM_i of the component hierarchy. The hiding mechanism provided by the external interfaces may or may not be used. Not using it for a given component means splitting a components into its constituents of a lower level.

2.1 Semantics of components

Now we define the transition system representing the behaviour of components and hierarchical components. They have transitions labelled by interactions, such that interactions for which $a \sqcup b$ but no explicit interaction $c = a.b$ is defined are independent. Moreover, we require the transition system to respect the maximal progress requirement of BIP which says that larger interactions should be preferred over smaller ones; nevertheless transitions whose interactions are maximal in some port cannot be eliminated from TS, because they are part of a bigger interaction in the composed system.

Definition 10 (Component Semantics). *Let K be a component with an internal interface or composition model $CM = ((\mathcal{P}_{CM}, <, \#), IS, \dashv)$ and the asynchronous transition system $TS = (Q, q_0, cl(IS), \rightarrow)$ defining the internal view of the behaviour of K as defined in Definition 4.*

The internal view of the semantics *of K, is the transition system TS' which has a smaller transition relation $\rightarrow'$ such that*

– *for interactions* $a, b \in cl(IS)$, $a < b$ *and* $\nexists p \in \mathcal{P} \,.\, a = \sqcup act(p)$, *then* $en(a) \cap en(b) = \emptyset$.

The external view of the behaviour of K *is a transition system* $TS'' = (Q, q_0, cl(\mathcal{P}), \rightarrow)$ *obtained from* TS' *by*

– *renaming interactions in* $\sigma \in IS$ *to port names in* $p \in \mathcal{P}$ *such that every transition of the form* $q \xrightarrow{\sigma}_{\prime} q'$ *of* TS' *is replaced by a set of transitions of the form* $q \xrightarrow{p}_{*} q'$ *for each* p *such that* $\sigma \in act(p)$. $q \xrightarrow{\sigma}_{\prime} q'$ *is replaced by a transition* $q \xrightarrow{\tau} q'$ *if* σ *is an interaction not corresponding to a port in* $\mathcal{P}$.
– *and then by eliminating internal* τ *transitions by defining* $\rightarrow$ *as the smallest transition relation such that* $q \xrightarrow{w} q'$ *if* $w = w_1; w'$ *and* $\exists q''$ *such that* $q \xrightarrow{\tau^*;w_1;\tau^*} q''$ *and* $q'' \xrightarrow{w'} q'$, *where we extend transition relations to sequences of interactions as usually*

This constraint on the transition relation is on of BIP. Whenever an action a — which is not a maximal interaction of some components — is dominated by an action b which is enabled, then a transition with a should not be enabled and if one exists, it is eliminated.

Such a maximal proegress rule can only be applied in a subsystem because we suppose that only the interaction set defined by ports are connected and therefore able to form new global interactions together with the environment.

The external view of a component forgets about the actual interactions connecting components inside K and replaces such an interaction σ by a port names p whenever σ is a complete interaction that can be fired without any interaction of the environment or if σ represents the $\sqcup$ of the set $act(p)$ for a port p even if this p is of type *sync*. In any case, the connection with the environment does not need to know the precise interaction that has been taken on p but only that it is a complete or maximal one for p.

We must now define how to infer the behaviour of a hierarchical component from the ones of its subcomponents. For that, we need to express the interaction model of a hierarchical component with respect to all the possible interactions within its subcomponents.

Definition 11 (Behaviour of a composition). *Consider a hierarchical component* K *whose structure is defined on a set of components* $\{K_i\}$ *with* $CM = ((\mathcal{P}, <, \#), IS, \dashv)$ *and* $Int(K_i) = (\mathcal{P}_i, <_i, \#_i)$, *the external view of the behaviour of the components* K_i *is defined by* $TS_i = (Q_i, q_{i0}, cl(Ports_i), \rightarrow_i)$.

Then, the behaviour of K *is defined by a transition system of the form* $TS = (Q, q_0, cl(IS), \rightarrow)$, *where*

– $Q = \Pi_{i=1..n} Q_i$ *where we write* $q = (q_1, ...q_n)$ *for* $q \in Q$;
– $q_0 = (q_{10}, ...q_{n0})$;
– *the transition relation* $\rightarrow$ *defining the internal view of the behaviour of* K *is the smallest transition relation such as:*
  - *if* $\sigma = (x_{i_1}..x_{i_J}) \in IS$ *and* $q_{i_j} \xrightarrow{x_{i_j}}_j q'_{i_j}$ *for* $j \in J$, *then* $(q_1, ...q_n) \xrightarrow{\sigma} (q''_1, ...q''_n)$ *where* $q''_i = q_i$ *for* $i \in J$ *and* $q''_i = q_i$ *for* $i \notin J$.

- *if* $q_i \xrightarrow{a}_i q_i'$ *and internal transition of* TS_i *then* $(q_1, ..q_i..q_n) \xrightarrow{a} (q_1, ..q_i'..q_n)$

TS is transformed into a transition system TS' satisfying also the maximal progress rule as defined in Definition 10 and then into TS'' defining the external view of the behaviour of K.

If the composition model CM is clear from the context, we denote the resulting TS' or the external view TS'' by $\|_{CM}TS_i$ or simply by $TS_1\|...\|TS_n$

As already stated TS needs not to be provided explicitly for defining the component K. In the sequel, when we denote the behaviour of a component by TS, we do not require this to stand for an explicit representation.

Property 2. The transition system $TS'' = (Q, q_0, \Sigma, \rightarrow_{\prime})$ defining the external view of the behaviour of K as defined before satisfies all the requirements of Definitions 4 and 10, that is, it is an asynchronous transition system. Moreover, the transition systems TS'' and TS' are bisimilar.

We define also some derived semantic sets that will be used for the definition of the partial order relation between components and for the satisfaction relation.

Definition 12 (Semantic sets). *Let be a component K whose structure is defined on a set of components $\{K_i\}$ with $CM = ((\mathcal{P}, <, \#), IS, \dashv)$ and $Int(K_i) = (\mathcal{P}_i, <_i, \#_i)$ such that $TS' = (Q, q_0, IS, \rightarrow_{\prime})$ represents the internal and $TS'' = (Q, q_0, \mathcal{P}, \rightarrow)$ satisfies the external view of this behaviour.*

In the following, we define semantic notions always with respect to some interaction model $IM = (\Sigma, <, \#)$, such that the external version is obtained by choosing IM to be the set of ports, and the internal one by choosing IS as the interaction set

- $\text{traces}(K) = \text{traces}(TS, \Sigma) \subseteq \Sigma^* : \{w = \in \Sigma^* \,|\, \exists q' \in Q \,.\, q_0 \xrightarrow{w} q'\}$ *these are the possible traces of K in terms of external interactions*
- $\text{acc}(K) \subseteq \text{traces}(K) \times 2^{\Sigma}$ *:* $\{(w, B) \in \Sigma^* \times 2^{\Sigma^*} \,|\, \exists q' \in Q \,.\, q_0 \xrightarrow{w} q' \wedge B = \{\sigma | q' \xrightarrow{\sigma}\}\}$ *for each trace w, this defines the set of maximal sets of interaction that may be enabled in K after some execution of an observable trace w. This means, we consider internal transitions to be under the control of the component that cannot be forbidden by an non cooperative environment.*
- $\text{ref}\ (K) = \text{ref}(TS, \Sigma) = \{(w, B') \setminus \downarrow B) \,|\, B' \subseteq \Sigma, \exists (w, B) \in \text{acc} \wedge B' \subseteq \Sigma \setminus \downarrow B\}$ *is a downwards closed set that defines all interaction sets that may be refused by K after w.*
- $\text{REF}\ (K) = \text{REF}(TS, \Sigma)$ *is the set of* refused traces*; they are traces $w; b$, where w is a trace of K and b potentially refused interaction in B such that (w, B) is a refusal of K.*
- $\text{dead}\ (K) \subseteq \Sigma^* : \{w | (w, \emptyset) \subseteq \text{acc}(K)$. *These are deadlocks of K, which can only be avoided by environments that avoid w.*
- $\text{dead}^{pot}\ (K) \subseteq \Sigma^* : \{w | (w, \emptyset) \subseteq \text{acc} \setminus inc(K)$ *where B contains interactions which are not complete in K, that is which define a* sync *port of K and are not part of any other set $act(p)$. These are potential deadlocks that may occur in non cooperative environments.*

For each trace w, exists a refusal set B if there exist an execution for w in TS to a state q in which a superset of B is refused. We consider a local version of acceptance/refusal sets where acceptance sets contain any actions that are accepted in S after w, that means all those that *may be accepted* in global context of K.

Definition 13 (Deadlock freedom of a specification). *Let K be a component. Then,*

- *K is internally deadlock free if* $\text{dead}(K) = \emptyset$ *that is K has no deadlocks in TS.*
- *K is deadlock free* deadlock *if* $\text{dead}^{pot}(K) = \emptyset$*, that is K has no potential deadlocks.*

Property 3. Let K be a component as above. Then, we have:

- $traces(K); \Sigma \cap \overline{traces(K)} \subseteq REF(K)$
- $dead^{pot}(K) \subseteq dead(K)$
- K is (internally) deadlock free for the internal semantics if and only if it is for the external semantics where the second is defined on a much smaller action set

2.2 Preorder and Satisfaction Relations

We define first a preorder relation that is adequate for the intended property verification, in the sense that smaller models satisfy more properties and smaller properties are satisfied by more models. A stronger preorder whose equivalence is also a congruence will be considered later.

We define a preorder that compares transition systems with respect to a given interface only. The reason is that we consider the problem of comparing components for a given hierarchical definition of interfaces. In the following, we always provide the definitions based on the external semantics. We can do this without loss of generality as we have shown the internal and the external view of the semantics to be bisimilar.

Definition 14 (Preorder and Equivalence on behaviours). *Let K be a component with $IM = (\Sigma, <, \#)$ its internal or external interaction model. Let TS, TS' be transition systems on Σ.*

We define the following preorder relation between transitions system with respect to IM:

- $TS \preceq_{IM} TS'$*, iff*
  1. $\text{traces}(TS, \Sigma) \subseteq \text{traces}(TS', \Sigma)$ *and*
  2. $\text{ref}(\Sigma, TS) \supseteq \text{ref}(\Sigma, TS')_{|\text{traces}(\Sigma, TS)}$ *where* $\text{ref}(\Sigma, TS')_{|\text{traces}(\Sigma, TS)} = \{(w, B) \in \text{ref}(\Sigma, TS') \mid w \in \text{traces}(TS, \Sigma)\}$
- $TS \approx_{IM} TS'$ *iff* $TS \preceq_{IM} TS'$ *and* $TS' \preceq_{IM} TS$

- *we extend the preorder and equivalence to components K and K' for the interaction model IM such that the behaviour of K is TS and the one of K' is TS'.*
  $K \preceq K'$ iff $TS \preceq_{IM} TS'$ and
  $K \approx K'$ iff $TS \approx_{IM} TS'$.

Property 4 (Minimal and Maximal behaviours for an interface). The following properties hold

- For a given interaction model IM, the smallest component, called $dead_{IM}$ is defined by any transition system TS which has $\{\epsilon\}$, as its set of traces and refuses everything after ϵ. This means *dead* is internally deadlocking
- For any interaction model IM, the largest component, called $true_{IM}$ is defined by any transition system TS which has Σ^* as its set of traces and refuses never anything. Thus, *true* has no internal deadlock but if no interaction in Σ is complete, then *true* may deadlock in an adversary environment
- For any component K with external interface Int and behaviour TS
  - $dead_{IM} \preceq K$
  - $K \preceq true_{IM}$

We define now the satisfaction relation expressing that a component K for the interaction model $IM = (\Sigma, <, \#)$ and behaviour TS_K satisfies some property, where a property is expressed by a transition system TS on $\Sigma' \subseteq \Sigma$.

Definition 15 (Property for an interaction model). *Let $IM = (\Sigma, <, \#)$ be an interaction model and let TS be a transition system on Σ'. where Σ' is a subset of Σ that is downwards closed in Σ. That is $IM' = (\Sigma', <, \#)$ is a sub interaction model of IM. Then, TS represents a property for IM.*

In order to define the satisfaction relation, we compare a behaviour TS defiend on IM of a component K with a property TS' on IM' which is a sub interaction model of IM. In order to do so, we project TS on IM'.

Definition 16 (Projection of a Transition Systems to smaller interface). *Let $IM = (\Sigma, <, \#)$ be an interaction model and $IM' = (\Sigma, <, \#)$ is a sub interaction model of IM. Let $TS = (Q, q_0, \Sigma, \rightarrow)$ be a transition system. Then, define $TS' = (Q, q_0, \Sigma', \rightarrow_*)$ by*

- *if $q \xrightarrow{\sigma} q'$ and $\sigma' = \sqcup\{x \in \Sigma' \,.\, x < \sigma\}$, then $q \xrightarrow{\sigma'}_* q'$*
- *if $q \xrightarrow{\sigma} q'$ and $\sqcup\{x \in \Sigma' \,.\, x < \sigma\} = \emptyset$, then $q \xrightarrow{\tau}_* q'$*

We call TS' the projection $proj(TS, \Sigma')$ of TS to Σ'.

that is the reachable states of TS are not changed, only the transition labels are simplified.

Definition 17 (Satisfaction Relation). *Let TS on $IM = (\Sigma, <, \#)$ be a behaviour of a component K and TS_P a property for the subinteraction model $IM' = (\Sigma', <, \#)$ of IM. Then,*

$$K \models P \quad \textit{iff} \quad \text{traces}(proj(TS, \Sigma')) \cap \text{REF}(TS_P, \Sigma') = \emptyset$$

That is $K \models P$ if no trace w of K projected to Σ' may be refused by P.

Property 5. Let TS and TS_1 TS_2 be behaviours on the interaction model $IM = (\Sigma, <, \#)$ define components K, K_1, K_2 and $TS_P, TS_{P'}$ define properties P and P' defined on $IM' = (\Sigma, <, \#)$. Then,

- $K \models P$ implies $\mathit{traces}(\mathit{proj}(TS, \Sigma')) \subseteq \mathit{traces}(P)$ and $\mathit{ref}(P) \subseteq \mathit{ref}(\mathit{proj}(TS, \Sigma'))$, more precisely, everything that may be refused by P must be refused by K
- Call a component K deterministic if for each $w \in \mathit{traces}(K)$, $w \notin \mathit{REF}(K)$, which means that K has a deterministic transition relation.
  If TS_P is deterministic, then $K \models P$ *if and only if* $\mathit{traces}(\mathit{proj}(TS, \Sigma')) \subseteq \mathit{traces}(P)$.
- if $P \preceq_{IM'} P'$ and $K \models P$, then $K \models P'$
- if $K' \preceq_{IM} K$ and $K \models P$, then $K' \models P'$
- if $K_1 \preceq K_2$ then $K_1 \| K \preceq K_2 \| K$, that is, $\preceq$ is preserved by composition.
- $K \models P$, then $K \| K_2 \models P$.
- $K \models P$ and $K \models P'$ iff $K \models P \| P'$ (as for a common sequence of P and P', $P \| P'$ may refuse what at least one of P or P' may refuse)
- When $K_1 \models P$, then $K_2 \preceq K_1$ implies $K_2 \models P$
- When $K_1 \models P$, then $P \preceq P'$ implies $K_1 \models P'$
- $K \models P$ implies $K \preceq P$

That is, the satisfaction relation implies trace inclusion in all cases and is identical to trace inclusion for deterministic specifications and the preorder $<$ on specifications is adequate for the satisfaction relation.

2.3 Decompositions and recompositions of components

An interaction model does not uniquely define a maximal interface. As it is shown in [BS07], there are generally alternative ways of defining connectors on a set of ports for obtaining a given interaction set IS. can be mapped to many different interfaces; this is due to the fact that E.g., if the interaction a is dominated by more than one port, then one can only say that a must be part of the action sets of at least one of these ports (otherwise a wouldn't be in Σ).

We have defined the notion of interaction model as in BIP, as it is simple and provides enough information for deriving several useful properties. In particular, components are defined by an interaction model and a behaviour, where the behaviour is defined by a transition system on the set of interactions of the component. The ports are only needed for defining the way the component may be composed.

[BS07] provides the following useful theorem which allows to decompose a component into its set of components K_i and then to compose it in such a way that it is a component obtained as a composition of one of the components K_i with a component K' grouping all the other subcomponents.

This allows us to obtain for any component the composition model relating K_i to its environment and for a closed system, this allows us also represent the environment of K_i by a set of ports connected to the ports of K_i via some typed connectors.

Theorem 1 (Decomposition of a connector). *Given an arbitrary connector x and a port p it is always possible to construct a connector $\tilde{x}$ such that x defines the same interaction model as $\tilde{x}$ and $\tilde{x}$ is of the form $(p, con_1, ...con_k)$ and p does not appear in $con_2, ...con_n$.*

The same transformation can be done for a set of ports

Then, we are interested in the interface for S_K defined by the sets of ports $\mathcal{P} \cup \mathcal{P}_E$ and the interaction model defined by Con.

Definition 18 (A component and its environment). *Let $Int(K) = (\mathcal{P}, <, \#)$ be the external interface of a component K.*

Then consider a component K_E with interface $Int(K_E) = (\mathcal{P}_E, <_E, \#_E)$ and no internal structure such that each port of $\mathcal{P}_E$ is connected to ports in $\mathcal{P}$ via a set of connectors, defining the the composition model between K and its environment.

Then, the internal structure of the component S_K obtained by composing K with its environment K_E is defined as $(\{K, K_E\}, CM_{EK})$ where $CM_{EK} = (Ports \cup \mathcal{P}_E \cup Con, IS, <, \#)$ such that Con is a set of connectors on $\mathcal{P} \cup \mathcal{P}_E$ and $IS, <, \#, \dashv$ are as defined by definition 7.

We can now also define a composition model relating any subcomponent K_i of K to its environment K_i^E which is defined by the peer K_j and K_E and their composition models.

$(\{str(K_i)\}, CM, \mathcal{P}, \dashv)$ define the structure of K, where $\mathcal{P}$ is a subset of $\bigcup_i \mathcal{P}_i \cup \mathcal{P}_{CON}$ corresponding to the hierarchical connectors CON defining the internal composition model CM of K with $CM = ((\bigcup_i \mathcal{P}_i \cup \mathcal{P}_{CON}, <, \#), IS\ \dashv)$.

Then, due to the theorem above for any given $i \in I$ one can define a composition model CM' for form $\{K_i, i \in I\} \cup \{K_E\}$ where $CM' = ((\mathcal{P}_{CM'}, <', \#'), IS', \dashv')$ is defined as

- *the set of ports $\mathcal{P}_{CM'}$ is a set of ports defined by a union of 3 sets of connectors: $\mathcal{P}_i$, $CON_{Ei} \supseteq \bigcup_{j \neq i} \mathcal{P}_j \cup \mathcal{P}_E$ a hierarchical set of connectors connecting only ports not in $\mathcal{P}_i$, and finally a hierarchical set of connectors CON_{i-Ei} connecting ports in $Ports_i$ with ports in CON_{Ei}*
- *the interaction set IS' which is obtained according to definition 7 is IS*
- *the definitions of $<'$, $\#'$ and $\dashv'$ are straightforward*

We define then an external interface of a component K_{Ei}, $Int(K_{Ei}) = (\mathcal{P}_{Ei}, <, \#)$ representing the elements of CON_{Ei} used by some connector of CON_{i-Ei} and $<$ and $\#$ are derived straightforwardly.

Then, the component S_K can also be defined by composing K_i with its environment K_{Ei} and the corresponding composition model is defined as $(\{K_i, K_{Ei}\}, CM_{K_i})$ where $CM_{K_i} = (Ports_i \cup \mathcal{P}_{Ei} \cup CON_{i-Ei}, IS_{Ei}, <_{Ei}, \#_{Ei})$ such that IS_{Ei} is obtained by renaiming interactions of IS in terms of $Ports_{Ei}$ and also $<_{Ei}, \#_{Ei}, \dashv_{Ei}$ are as defined by definition 7.

3 Components enriched with contracts and compositional verification

3.1 HRC: hierarchical components enriched with contracts

First, we introduce the notion of Rich Component (HRC), similar to the one introduced in [BCSM07,BBB+07] but adapted to our hierarchical BIP components: the structure of an HRC K is the structure of a component, enriched with a composition model with its environment K_E as defined in Definition 18 and a set of contracts.

A contract is a pair of transition systems (A, G), defined on $\mathcal{P}_K$, respectively $\mathcal{P}_E$. A expresses an assumption of the behaviour of the environment and G defines a property that K must — or is assumed to — satisfy under the condition that the environment behaves according to A.

A rich component has, exactly as a component, a behaviour that is either explicitly given for a leave component or implicitly defined by the set of leave components. In the context of contract based reasoning, we want to be able to do some reasoning without having already defined all the leave components and/or their behaviour.

Definition 19 (Assumption, Guarantee, Contract). *Let be* $Int = (\mathcal{P}, <, \#)$ *an interface. A* contract *for* K *is given by a pair* (A, G) *where* A *and* G *are transition systems with labels in* $\mathcal{P}$*;* A *is called the* assumption *and* G *is called the* guarantee.

Definition 20 (Rich component and their interfaces (HRC)). *The structure of a rich component or HRC is of the form* $((str(K_i), CM, (\mathcal{P}_K, <, \#), \dashv), (\mathcal{P}_E, CM_{EK}), CONTR)$ *or* $((\mathcal{P}_K, <, \#), (\mathcal{P}_E, CM_{EK}), CONTR)$ *where*

- $(\mathcal{P}_E, CM_{EK})$ *is a set of ports representing the environment and a set of connectors connecting* K *to its environment as defined in Definition 18; if* K *is defined as a part of a larger system then the second construction of this definition is used, whereas if* K *is a unique outermost component described, then the first construction of Definition 18 is used.*
- $\{K_i\}$ *are HRC and* $(\{str(K_i)\}, CM, \mathcal{P}, \dashv)$ *is defined like a structure of a hierarchical component or alternatively,* $((\mathcal{P}, <, \#), TS)$ *defines an atomic component without a defined substructure.*
- $CONTR$ *a set of contracts of the form* (A_i, G_i) *where* A_i *is defined on the alphabet* $\mathcal{P}_E$ *and* G_i *is defined on the alphabet* $\mathcal{P}$ *.*

A rich component K *is defined by a structure* $str(K)$ *and by* $beh(K)$ *defining a transition system on the external interface* $(\mathcal{P}_K, <, \#)$ *of* K*.*

Notice that for assume guarantee reasoning, we are mainly interested in the structure of the component K, whereas the behaviour of K may not always be given or computed.

Given the structure of an HRC K we are now able to consider the environment K_E of K like any other component. How to obtain a valid K_E is defined in section 2.3.

3.2 Compositional verification of HRC

For being able to define compositional verification, we need to define a satisfaction relation, defining what it means for a contract (A, G) to be satisfied by K, and a dominance relation such that (A, G) dominates (A', G') if all components satisfying (A', G') satisfy also (A, G). We use the dominance relation for showing that a contract (A, G) associated with a hierarchical component dominates the implicitly defined contract defined by a set of contracts (A_i, G_i) associated with the subcomponents of K.

Intuitively, K satisfies a contract (A, G) if in the system defined by the environment of E_K and K, where the environment behaves like A, this guarantees that K satisfies the property G.

We first consider the slightly simpler case where each component has a single contract. The extension to multiple contracts is more complicated to formulate but not very difficult to do.

Definition 21 (Satisfaction of contracts). *Let K be a rich component with an external interface $Int_K = (\mathcal{P}, <, \#)$ and K_E an environment with $Int_E = (\mathcal{P}_E, <, \#)$ defined by $(\mathcal{P}_E, CM_{EK})$ and a behaviour representing a transition system TS on $\mathcal{P}$. Then, K satisfies its contract (A, G), denoted $K \models (A, G)$ if*

$$S_K \models G$$

For S_K defined as the composition via CM_{EK} of the components K and K_E defined by the external interface Int_E and behaviour A.

According to the satisfaction relation of definition 17, as G is defined as a property on the alphabet of K, and the behaviour of S_K is of the form $A\|_{CON}TS$ on the composition of the alphabets of K and E_K, $A\|_{CON}TS \models G$ means that the projection of $A\|_{CON}TS$ onto the alphabet of K satisfies G.

Theorem 2. *Let K be a rich component with an external interface $Int_K = (\mathcal{P}, <, \#)$ and K_E an environment with $Int_E = (\mathcal{P}_E, <, \#)$ defined by $(\mathcal{P}_E, CM_{EK})$ and a behaviour representing a transition system TS on $\mathcal{P}$.*

If K satisfies its contract (A, G), then

$$A\|TS \preceq A\|G$$

where the parallel composition on transition systems is on both sides the one induced by the composition model of S_K.

In the particular case that G is deterministic, we have for $TS = G$ that $K \models (A, G)$

proof sketch*: Noting that the behaviour of S_K is equal $A\|TS$, we have $A\|TS \models G$ implies $A\|TS \preceq G$ by property 5; together with $S_K \preceq A$ which is due to monotonicity, this allows to derive the first property. The second is straightforward using the property saying that $G \models G$ for deterministic G and the fact that $A\|TS \preceq G$*

This important property expresses the fact that G defines an upper bound on all components that satisfy G in the environment defined by A; and this allows the definition of a simple proof rule for contract dominance.

Definition 22. *Let K be a hierarchical rich component with a structure of the form $((str(K_i), CM, (\mathcal{P}_K, <, \#), \dashv), (\mathcal{P}_E, CM_{EK}), (A, G))$ such that $str(K_i)$ define a contract (A_i, G_i). Then (A, G) dominates the set of contracts $\{(A_i, G_i)\}$ iff*

$$beh(K_i) \models (A_i, G_i) \quad implies \quad beh(K) \models (A, G)$$

Note that the behaviour of K is defined as the composition of the transition systems defining the behaviour of the K_i according to the composition model CM and renaming the resulting interactions to port names in $\mathcal{P}$ according to $\dashv$.

In [BBB+07] an explicit contract (A', G') is associated with the set $\{(A_i, G_i)\}$ and dominance is then defined as a relationship between the contracts (A', G') and (A, G) which are defined on the same alphabets. But there, the semantics is defined as a set of prefixes and the contract (A', G') is defined in terms of negations (complements of prefix sets), whereas in our framework negation is not a defined operation on behaviours. But also in our case, one can characterise contract dominance. We do this without constructing a contract (A', G') explicitly.

Theorem 3. *Let K be a hierarchical rich component with a structure of the form $((str(K_i), CM, (\mathcal{P}_K, <, \#), \dashv), (\mathcal{P}_E, CM_{EK}), (A, G))$ such that $str(K_i)$ has a contract (A_i, G_i).*

Then (A, G) dominates the set of contracts $\{(A_i, G_i)\}$ if the following conditions hold

- *for the component K obtained by choosing $beh(K_i) = G_i$, we have $K \models (A, G)$*
- *for all indices i, the component S_K defined as a composition of K_i with K_{Ei} obtained from $\mathcal{P}_{Ei}$ and Definition 18, and by choosing A for the behaviour of E_K and G_j for the behaviours of the K_j, for $j \neq i$, we have*

  $$S_K \models A_i$$

  meaning that the assumption A_i is not more restrictive than the one defined by the environment of K_i as defined by the guarantees of the pairs and the assumption A of K.

Proof sketch*: the fact that the K_i satisfying (A_i, G_i) are smaller than G_i in an environment granting A_i (Theorem 2), guarantees by the first verification condition that that for K_i having as behaviour the projection of $A_i \| G_i$ as previously defined, one has $K \models (A, G)$.*

The second condition guarantees that the restriction A_i can be eliminated as it is already guaranteed by A and the peer G_j; indeed, $A \| (A_1 \| G_1)_p \| ... \| (A_n \| G_n)_p$ is equivalent to $A \| G_1 \| ... \| G_n$, where the parallel composition is the one respecting the interaction model and $(A_i \| G_i)_p$ represents the projection onto K_i

4 Handling verification conditions contructively

We have defined a framework for architecture and system modelling based on BIP and we have adapted it for the use in the context of compositional verification, where components are annotated with contracts specifying assumptions on the environment and derived a set of verification conditions for showing the correctness of a contract hierarchy.

Contracts state properties on a specific component under some condition on its environment. We have defined verification conditions which are small if each component has only a small number of subcomponents. In general, this is unlikel to happen as component must on the other hand be units which are not too tightly coupled with their environment in order to make compositional verification feasible.

The verification conditions involve the verification of properties on compositions of component behaviours. $K \models P$ holds if the traces of K cannot be refused by P which means that $K \not\models P$ if for an approapiate composition model, the composition $K \| P$ can reach a deadlock state.

Together with the fact that we want to guarantee deadlock freedom of individual components and globally of the system, this means that methods for showing absence of deadlock are an important issue.

In [GS03,GGMC+07b,GGMC+07a] we have started to study specific methods for showing deadlock freedom without building products for the BIP framework which are currently being implemented and experimented.

Even if these methods avoid the exploration of the global state graph, they are global and they compute approximative results. Combining such methods or slightly more costly and more precise methods with a compositional approach may lead to interesting results.

We have defined components which have in there interface not only the possible interactiona and a set of contracts, but we define a notion of conflict and dependence on the set of ports of the components themselves defining corresponding properties of the transition system which can be exploited for obtaining efficient partial order reductions

The abstraction defined by the use of typed connectors is particularly intersting if we succeed to construct on-the-fly reductions. But we may also envisage an approach based on incremantal contruction and abstraction.

References

BBB+07. E. Badouel, A. Benveniste, M. Bozga, B. Caillaud, O. Constant, B. Josko, Q. Ma, , R. Passerone, and M. Skipper. SPEEDS meta-model syntax and draft semantics. SPEEDS deliverable D2.1c, February 2007.

BBS06. A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time systems in bip. In *4th IEEE International Conference on Software Engineering and Formal Methods (SEFM06), Invited talk, September 11-15, 2006, Pune, pp 3-12*, 2006.

BCSM07. M. Bozga, O. Constant, M. Skipper, and Q. Ma. SPEEDS meta-model syntax and static semanticsd21b. SPEEDS deliverable D2.1b, January 2007.

BS07. Simon Bliudze and Joseph Sifakis. The algebra of connectors structuring interaction in bip. Techreport, Verimag, February 2007.

GGMC+07a. Gregor Gössler, Susanne Graf, Mila Majster-Cederbaum, M. Martens, and Joseph Sifakis. An approach to modeling and verification of component based systems. In *Current Trends in Theory and Practice of Computer Science, SOFSEM'07*, number 4362 in LNCS, 2007.

GGMC+07b. Gregor Gössler, Susanne Graf, Mila Majster-Cederbaum, M. Martens, and Joseph Sifakis. Ensuring properties of interaction systems by construction. In *Program Analysis and Compilation, Theory and Practice*, LNCS, 2007.

GS03. Gregor Gößler and Joseph Sifakis. Component-based construction of deadlock-free systems. In *proceedings of FSTTCS 2003, Mumbai, India*, LNCS 2914, pages 420–433, 2003. downloadable through http://www-verimag.imag.fr/ sifakis/.

GS05. G. Goessler and J. Sifakis. Composition for component-based modeling. *Science of Computer Programming*, pages 161–183, March 2005.

WN95. G. Winskel and M. Nielsen. *Models for concurrency.* Vol. 4, Oxford Univ. Press, 1995.