Contracts for BIP: Hierarchical Interaction Models for Compositional Verification*

Susanne Graf and Sophie Quinton

Verimag/CNRS and Verimag/ENS Cachan

Abstract. This paper presents an extension of the BIP component framework to hierarchical components by considering also port sets of atomic components to be structured (ports may be in conflict or ordered, where a larger port represents an interaction set with larger interactions). A composed component consisting of a set of components connected through BIP connectors and a set of ports representing a subset of the internal connectors and ports, has two semantics: one in terms if interactions as defined by the BIP semantics, and one in terms of the actions represented by external ports where the structure of the port set of the component is derived from the internal structure of the component.

A second extension consists in the addition of implicit interactions which is done through an explicit distinction of conflicting and concurrent ports: interactions involving only non conflicting ports can be executed concurrently without the existence of an explicit connector.

Finally, we define contract-based reasoning for component hierarchies.

1 Introduction

We aim at contract-based verification. We consider a framework where a system is a hierarchically structured set of *components*. For this purpose, we extend the component framework BIP [GS05,BBS06] and in particular its instance based on hierarchical connectors [BS07] to a framework for hierarchical components enriched with contracts as defined in the SPEEDS project [BC07$^+$].

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 and we consider here also a hierarchical organisation of the components. Only leaf components represent models with behaviour explicitly defined by a transition system labelled by interactions. Originally, in BIP, atomic components have a sequential behaviour, but here they are not different from hierarchical components, at least from outside. We represent behaviours by an asynchronous transition system, and we may choose other, more efficient, representations in the future.

The behaviour of a hierarchical component is obtained as a composition of the behaviours of its leaf components depending on its internal connectors.

A hierarchical rich component (HRC) K has includes contracts, in the form of an assumption A and a guarantee G, represented both by transition systems. A

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

J. Derrick and J. Vain (Eds.): FORTE 2007, LNCS 4574, pp. 1–18, 2007.

defines a property of the environment of K, and G a property of K that should hold if K runs in an environment guaranteeing A. We define a framework for verifying that components satisfy their contracts compositionally, by showing that the contracts associated with each component dominate the contracts of its inner components, and leaf components satisfy their contracts.

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 expressing synchronous and asynchronous interaction and execution, including blocking rendez-vous. Here, we only represent the abstract setting without taking into account data flow.

A main issue in embedded systems is absence of interference between transactions, possibly executed concurrently. Using BIP interactions, we can guarantee interference freedom by construction, as only non interfering transactions are executed concurrently. As a counterpart, it must be verified that interlock situations and violations of non functional requirements cannot occur; such bad situations can be reduced a deadlock in a modified system.

In Section 3, we describe how we intend to verify the consistency of a contract hierarchy. We adapt classical assume guarantee reasoning (see [RB$^+$01] for a good overview) to our framework. To prove that a contract (A, G) of K dominates a composition of contracts $\{(A_i, G_i)\}$ — those of the subcomponents of K — it is sufficient to show that

- $A\|G_1\|....\|G_n \models G$; that is, if every K_i ensures its guarantee, then the composition ensures G, as long as the environment behaves according to A
- $A\|G_1\|...\|G_n \models A_i$ for all i; that is, each assumption A_i can be derived from A and the guarantees G_j of the peer components.

This proof rule is sound as A and G constrain different components. Notice that this proof rule is global at a given level of hierarchy, the gain comes from a hierachical structure with several layers.

In Section 4 we give a first idea on how we intend to achieve a more efficient and scalable handling of contracts. In particular, proving verification conditions is reduced to showing deadlock freedom of a transformed system, and we are presently developing efficient methods for such checks.

2 Specifications and Their Semantics

Definition 1 (Interaction set). *Let Σ be a set, and $<, \# \subseteq \Sigma \times \Sigma$ binary relations. 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 in Σ and by $\downarrow a = \{b \in \Sigma \mid b < a\}$ the downwards closure of a in Σ and we extend these notions pointwise to sets and sequences.

Denote by $a \sqcup b$ the $c \in \Sigma$ representing the least upper bound of a and b, if it exists. Define the closure *of Σ, $cl(\Sigma)$, the interaction set obtained by recursively adding elements $a \sqcup b$, whenever $a, b \in \Sigma$, not $a\#b$ and there exists no $c = a \sqcup b$ in the set.*

Note that $a\#b$ may hold even if $a \sqcup b$ exists. Interactions for which $a \sqcup b$ exists can be connected, and 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 an interaction set Σ, $cl(\Sigma)$ is an interaction model.

Here, we consider interaction models that are defined as closures of an interaction set Σ. And, we refer to $a \sqcup b \subseteq cl(\Sigma) \setminus \Sigma$ as an *implicit* interaction.

In particular, a union of interaction (sets) 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.

In interaction models of [GS05], $a\#b$ holds (implicitly) whenever $a \cdot b = a \sqcup b$ is not explicitly defined. Here, we can avoid the definition of an explicit interaction $a \cdot b$ when a and b are independent; 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 trig or sync, where $\{sync, trig\}$ form a boolean algebra with* sync $<$ trig*.*

For $\mathcal{P}$ a set of (typed) ports, an (external) interface *Int is the interaction model $(cl(\mathcal{P}), <, \#)$ defined by the interaction set $(\mathcal{P}, <, \#)$.*

The type of implicit ports $p \sqcup r \in (cl(Sigma) \setminus \Sigma)$ is the $\vee$ of the types of p and r.

Interactions on ports of type *sync* need to realise an interaction on a connector, the collaboration of peer components, whereas those of type *trig* can go alone for realising an interaction on a connector connecting this port to others. But they need not to be system wide *complete* interactions. We might explicitly distinguish complete ports as well.

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

- *Q is a set of states and $q_0 \subseteq Q$ an initial state.*
- *$\rightarrow \subseteq Q \times cl(\mathcal{P}) \times Q$ is a transition relation. For $a \in \mathcal{P}$, $en(a)$ is the set of states in which a is enabled ($\exists q' \in Q \,.\, q \xrightarrow{a} q'$). We use $\rightarrow$ also to represent a sequences of transitions, where $a; b; c : ...$ is used to denote a sequence of interactions. $\rightarrow$ must satisfy the following constraints: for $a, b \in cl(\mathcal{P})$,*

- *if not* $a\#b$ *and not* $(a < b$ *or* $b < a)$*, then* $q \in en(a) \cap en(b)$ *implies* $(q \xrightarrow{a;b} q''$ *implies* $q \xrightarrow{b;a} q'')$ *and* $q \xrightarrow{a;b} q''$ *implies* $q \in en(a) \cap en(b)$
- $c = a \sqcup b \in \mathcal{P}$ *implies* $en(a) \cap en(b) = en(c)$ *and* $a < b$*, implies* $en(b) \subseteq en(a)$*.*

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

That is transitions of independent interactions commute, and the transition sequence $\xrightarrow{a;b}$ is semantically equivalent to $\xrightarrow{b;a}$ and also to $\xrightarrow{a \sqcup b}$, where the latter may or may not exist as an explicit transition in TS. In the semantics, we will explicitly add transitions for $a \sqcup b$. That means, TS represents an asynchronous transition as defined in [WN95].

We now define hierarchical components as compositions of components, and we define two views for them:

- an *external view*, which represents a hierarchical component to the environment exactly as an atomic component, as just defined.
- an *internal view* which makes visible the internal *structure* composition structure, consisting of a set of components K_i and a composition model CM defined by a set of hierarchical connectors as in [BS07].
- the internal and external *interfaces* are linked via a relation $\dashv$ associating subsets of ports and connectors of CM with external ports in $\mathcal{P}$ such that $\dashv$ is a structure preserving relation between the interaction set Σ defined by CM and $(cl(\mathcal{P}), <, \#)$.
  The internal view of the *behaviour* is defined by transitions with labels Σ, whereas the external one has transitions labels in $cl(\mathcal{P})$.

We now define the internal view of a hierarchical component.

Definition 5 (Connector and hierarchical connector). *Let* $(\mathcal{P} = \bigcup \mathcal{P}_i, <, \#)$ *be the union interaction set induced by the set* $\{Int_i = (\mathcal{P}_i, <_i, \#_i)\}$ *of (external) component interfaces. A* typed connector *con on* $\{Int_i\}$ *consists of:*

- *a subset of* $\mathcal{P}$ *(also denoted con) such that* $\forall p_m, p_n \in con$*, the least upper bound* $p_m \sqcup p_n$ *is defined in the union interaction model but* $p_m \sqcup p_n \notin \mathcal{P}$*.*[1]
- *a type* sync *or* trig*.*

A connector $con = \bigcup_{l=1..n} p_i$ *defines an interaction set* $\Sigma = \mathcal{P} \cup act(con)$ *where* $act(con)$ *contains:*

- *if all* $p_l \in con$ *are of type* sync *(that is, all ports must synchronise), then,* $p_1 \cdot ... \cdot p_n$ *is the only element of* $act(con)$
- *if con contains also ports of type* trig*, then* $act(con)$ *contains all* $p_1 \cdot ... \cdot p_m$ *such that* $\bigcup_{l=1..m} p_i \subseteq con$ *and* $\exists l \in \{1..n\}$ *such that* p_l *of type* trig*.*

[1] We will see that this means that either interactions of p_n and p_m are independent, or $p_m \# p_n$, and then $p_m \sqcup p_n$ is a port defined somewhere "inside" one of the components K_i, but the connector $p_m \sqcup_i p_n$ is not in the interface of K_i.

The preorder relation $<'$ on Σ is derived from $<$ and $p <' p \cdot q$. The conflict relation $\#'$ contains $\#$ and $\sigma\#\sigma'$ if their definitions involve conflicting ports or ports related via $<$.

Extend act to all $p \in Ports$ by $act(p) = \{p\}$ and represent Σ by $\bigcup_{p\in\mathcal{P}} act(p) \cup act(con)$. Now, define an extended port set $\mathcal{P}_{con} = \mathcal{P} \cup \{'con'\}$, that is the set of original ports extended by a port con representing the connector con.

Now we can define hierarchical connectors. Let be $(Ports, <', \#')$ an interaction set and $(Ports \cup CON, <, \#)$ a port set extended by a set of connector ports CON. A hierarchical connector on $\mathcal{P}$ is a connector con on $Ports \cup CON$ that may connect both ports and connector ports. The definition of con must respect exactly the constraints imposed on a connector. But the interaction set $act(con)$ for $con = \bigcup_{l=1..n} p_i$ is defined by recursively instantiating connector ports by their interaction sets:

- *if all $p \in con$ are of type* sync*, then $a_1 \cdot ... \cdot a_n$ such that $a_i \in act(p_i)$*
- *if con contains ports of type* trig *then, $a_1 \cdot ... \cdot a_m$ such that $\bigcup_{i=1..m} p_i \subseteq con$, $\exists l \in [1..m]$ s.th. p_l of type* trig*, and, as before $a_i \in act(p_i)$*

The preorder and the conflict relations are defined exactly in the same way as for a connector. Now, the connector set can be extended by adding the hierarchical connector con as a new port for the definition of new hierarchical connectors.

We denote by $CM = (Int_i, (\Sigma, <, \#), \mathcal{P} \cup CON)$, the composition model defined by the set of (hierarchical) connectors CON, where Σ is the derived interaction set.

Property 2. Let $CM = (Int_i, (\Sigma, <, \#), \mathcal{P} \cup CON)$, be a composition model. Then, for all ports, including connectors *con*, $act(con)$ is an upwards-closed interaction set with maximal element $p_1 \cdot ... \cdot p_k$, such that $\{p_1, ..., p_k\}$ is the set of ports in $\mathcal{P}$ involved in *con*, obtained by recursively replacing connector ports by the set of ports in $act(p)$. Furthermore, $\forall p, r \in Ports \cup CON$:

- $p < r$ iff $act(p) \subseteq act(r)$
- $p\#r$ iff $\exists a \in act(p) \exists b \in act(r)$ such that $a\#b$

Thus, a connector defines in turn a port, and a hierarchical connector is a connector connecting ports and connectors. A port p defines an interaction set that is the singleton containing just p, and a connector has a recursively defined interaction set containing composed interactions.

Now, we want to turn a composition of a set of components defined by a composition model CM into a (hierarchical) component. For this purpose, we introduce a new (external) interface that makes available for further connection a subset of ports and connectors as new external ports. The internal view of such a component is defined by CM with interactions in Σ, whereas the external view defines interactions in terms of the new external ports.

In a constructive approach, one may keep all ports and connectors available for further composition. Here, we suppose given some global system architecture, such that it is enough to expose those ports which are used in some connection at some level of hierarchy.

We define a relation between ports and connectors and an external interface.

Definition 6 (Mapping an interaction set on a set of ports). *Let* $CM = (Int_i, (\Sigma, <, \#), \mathcal{P} \cup CON)$ *be a composition model for a set of components, and* $\mathcal{P}'$ *a set of new ports.*

A relation $\dashv \subseteq \Sigma \cup CON \times \mathcal{P}'$ *defines an interaction-port association if*

- *for each* $p' \in \mathcal{P}'$ *there is a* $p \in \mathcal{P} \cup CON$ *of the same type as* p *such that* $a \dashv p'$ *iff* $a \in act(p)$
- *for each* $p \in \mathcal{P} \cup CON$*, either there exists* $p' \in \mathcal{P}'$ *with* $type(p) < type(p')$ *and* $act(p) \dashv p'$ *(exported port) or* $p \in CON$ *is of type* trig *(internal port) or* $p \in CON$ *is the least upper bound of ports* p_i *mapped to* $\mathcal{P}'$ *where* $\vee types(p_i) < type(p)$ *(implicit* $\sqcup$*-port)*

Definition 7 (Hierarchical component). *Let* K_i *be a set of components with (external) interfaces* $Int_i = (\mathcal{P}_i, <_i, \#_i)$ *and* $CM = (\{Int_i\}, (\Sigma, <, \#), \mathcal{P} = \bigcup \mathcal{P}_i \cup CON)$ *a composition model, and* $\mathcal{P}'$ *a set of new ports and* $\dashv$ *a interaction-port association between* Σ *and* $\mathcal{P}$*.*

Then, a hierarchical component K *is defined as* $K = (\{K_i\}, CM, \dashv, Int)$*. Where,*

- *we call* K_i *its subcomponents.*
- *The composition model* CM *is sometimes referred to as the internal structure of* K
- *We call* $(CM, \dashv, \mathcal{P}')$ *the internal interface.*
- $Int = (Ports', <', \#')$ *which is derived from* CM *and* $\dashv$ *in a straightforward way is the external interface 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 interactions in* Σ *is the internal view, and the one obtained by replacing labels in* Σ *by labels in* $\mathcal{P}'$ *the external view of the behaviour of* K*. We define next how the behaviour of* K *is defined as a composition of behaviours of* K_i*.*
- $(Int, beh(K))$ *is the component* K *as seen from outside* K*.*

We do not require that K provides an explicit transition system expressing its behaviour. It is implicitly defined by the transition systems of its subcomponents.

We can show that the interaction model of a hierarchical component K does not depend on how atomic components are grouped into subcomponents; this is done by showing that a hierarchical component interface is equivalent to composition of all its atomic components obtained by hierarchically flattening K.

2.1 Semantics of Components

Now, we define the semantic transition system representing the behaviour of a component. First, we transform a transition system defining the behaviour of K into a semantic transition system, that will be interpreted as a set of traces and refusals. and then we compose semantic transition systems to behaviours of hierarchical components.

Definition 8 (Component Semantics). *Let K be a component with an external interface $Int = (Ports', <', \#')$, and if it is a hierarchical component, an internal interface $(CM, \dashv, \mathcal{P}')$ with $CM = (\{Int_i\}, (\Sigma', <, \#), \mathcal{P}')$.*

Suppose that for K, an asynchronous transition system $TS = (Q, q_0, \Sigma, \rightarrow)$ as in Definition 4 is given, where Σ may be either $cl(\Sigma)$ for the internal view of the behaviour and $cl(\mathcal{P}')$ for the external view of the behaviour.

The (internal or external) view of the semantics *of K defined by TS, is $TS' = (Q, q_0, \Sigma, \rightarrow_*)$, where $\rightarrow_*$ is like $\rightarrow$, except that:*

- *for $a, b \in \Sigma$, $a < b$ and such that $\forall p \in \mathcal{P}' \,.\, a \dashv p$ implies $b \dashv p$, then $q \xrightarrow{a} q'$ and $q \xrightarrow{a} q''$ implies $q \xrightarrow{a}{}''_* $ but $q \not\xrightarrow{a}_*$.*
- *for $a, b \in \Sigma$, $a \sqcup b \in cl(\Sigma) \setminus \Sigma$, whenever $q \in en(a) \cap en(b)$ and $q \xrightarrow{a;b} q'$, then there is a new transition $q \xrightarrow{a \sqcup b}_* q'$*

If TS' defines the internal view of the semantics on $cl(\Sigma')$, then the external view is defined by $TS'' = (Q, q_0, cl(\mathcal{P}), \Rightarrow)$ obtained from TS' by

- *renaming internal interactions in $\sigma \in \Sigma'$ to external interactions $p \in \mathcal{P}'$: transitions $q \xrightarrow{\sigma}_* q'$ of TS' are replaced by a set of transitions $q \xrightarrow{p}_x q'$ for each p such that $\sigma \dashv p$. If σ is an internal interaction not related to a port in $\mathcal{P}$, then $q \xrightarrow{\sigma}_* q'$ is replaced by $q \xrightarrow{\tau}_x q'$.*
- *then by eliminating internal τ transitions: $\Rightarrow$ is the least transition relation such that $q \xRightarrow{p} q'$ if $\exists q''$ such that $q \xrightarrow{\tau^*;p;\tau^*}_x q''$.*

The maximal progress rule giving priority to larger interactions is as in BIP: In a global system, when $a < b$, then a b-transition has priority over an a-transition. We can apply the maximal progress rule partly in the semantics of a subsystem K, because the external ports define exactly the set of interactions that can be extended to larger connectors in the environment of K and our rule never eliminates all transitions corresponding to a given port, and the executability of an interaction in the global system does not depend on the particular interaction that is executed, only on the port.

The external view of the semantics of K forgets about the actual interactions due to the composition model defined on the subcomponents of K, and replaces interaction σ by port names p defined by $\dashv$.

Property 3. Due constraints on the selection of external ports, τ-transitions may always be executed independently of the environment of K, that is they are complete interactions in the sense of BIP.

We now define the behaviour of a hierarchical component.

Definition 9 (Semantics of a hierarchical component). *Consider a hierarchical component K defined by $K = (\{K_i\}, CM, \dashv, Int)$ with $Int(K_i) = (\mathcal{P}_i, <_i, \#_i)$, composition model $CM = (\{Int_i\}, (\Sigma, <, \#), \mathcal{P})$ and external interface $Int = (Ports', <', \#')$.*

Suppose that the external view of the behaviour of the components K_i is given by a transition system $TS_i = (Q_i, q_{i0}, cl(Ports_i), \rightarrow_i)$ satisfying the requirements of the relation $\rightarrow_*$ of Definition 8 (using the semantic transition relations simplifies the definition, but is not strictly required).

Then, the internal view of the behaviour of K can be defined through the transition system $TS = (Q, q_0, cl(\Sigma), \rightarrow)$, where

- *$Q = \Pi_{i=1..n} Q_i$ where we write $q = (q_1, ...q_n)$ for $q \in Q$; $q_0 = (q_{10}, ...q_{n0})$;*
- *$\rightarrow$ is the smallest transition relation such that:*
  - *if $\sigma = (x_{i_1}..x_{i_J}) \in \Sigma$ such that $\forall j, k \in J$, $x_{i_j} \in Ports_j$, and $\mathcal{P}_j \neq \mathcal{P}_k$, if $q_{i_j} \stackrel{x_{i_j}}{\rightarrow}_j q'_{i_j}$ for $j \in J$, then $(q_1, ...q_n) \stackrel{\sigma}{\rightarrow} (q''_1, ...q''_n)$ where $q''_i = q'_i$ for $i \in J$ and $q''_i = q_i$ for $i \notin J$.*
  - *if $q_i \stackrel{\tau}{\rightarrow}_i q'_i$ and internal transition of TS_i then $(q_1, ..q_i..q_n) \stackrel{\tau}{\rightarrow} (q_1, ..q'_i..q_n)$*

TS may then be transformed in turn into the two different semantic transition systems as in Definition 8.

In the following, we denote the resulting semantic transition system by $\|_{CM} TS_i$, respectively by $\|_{(CM,\mathcal{P})} TS_i$. If the parameters are clear from the context, we may omit them.

Notice that for $\sigma = p_1 \cdot ... \cdot p_k$ and $p_j, p_l \in \sigma \cap \mathcal{P}_i$, $p_j \sqcup p_l$ is always defined. In the definition above, we use the fact that the semantic transition relation of Definition 8 contains explicit transitions for such elements in $cl(\Sigma)$ which simplifies the definition of the product. Nevertheless, we could also directly compose the original asynchronous transition systems in which, for $a \sqcup b \in cl(\Sigma)$, the sequence $a; b$ is enabled implies both a and b are enabled.

Property 4. The transition system $TS'' = (Q, q_0, \Sigma, \Rightarrow)$ defining the external view of the behaviour of K is a again an asynchronous transition system as required by Definition 4. Moreover, TS'' and the transition system TS' defining the internal behaviour are bisimilar.

We derive now the set of traces and refusals used for the definition of the comparison of component behaviours and of the satisfaction relation.

Definition 10 (Traces and refusals). *Let K be hierarchical component $K = (\{K_i\}, CM, \dashv, Int)$ with $Int(K_i) = (\mathcal{P}_i, <_i, \#_i)$, composition model $CM = (\{Int_i\}, (\Sigma', <, \#), \mathcal{P})$ and external interface $Int = (Ports', <', \#')$.*

Let $TS = (Q, q_0, \Sigma, \rightarrow)$ represent either the internal or the the external view of the behaviour of K, depending on the choice of Σ.

- $\text{traces}_\Sigma(K) \subseteq \Sigma^* :\downarrow \{w \in \Sigma^* \mid q_0 \stackrel{w}{\rightarrow}\}$ *the downwards closure of the possible traces of K in terms of interactions in Σ, where we use the extension of $<$ on Σ to traces.*
- $\text{acc}_\Sigma(K) \subseteq \text{traces}(K) \times 2^\Sigma$ *: $\{(w, \downarrow B) \in \Sigma^* \times 2^{\Sigma^*} \mid \exists q' \in Q \exists w' . w < w' \wedge q_0 \stackrel{w'}{\rightarrow} q' \wedge B = \{\sigma | q' \stackrel{\sigma}{\rightarrow}\}\}$. For each trace w, this defines the set of maximal downwards closed sets of interactions that may be enabled in K*

after some execution of an observable trace w. This is because internal transitions are under the control of the component and cannot be forbidden by a non cooperative environment.
- $\text{ref}_\Sigma(K) = \{(w, B') \setminus B) \,|\, B' \subseteq \Sigma, (w, B) \in \text{acc} \wedge B' \subseteq \Sigma \setminus B\}$ *is set defining all interaction sets that may be refused by K in some state after w; it is upwards closed with respect to $<$ and with respect to set inclusion.*
  $\text{REF}_\Sigma(K)$ *is the derived set of* refused traces *of the form $w; b$, where w is a trace of K and $b \in B$ where (w, B) is a refusal of K.*
- $\text{dead}_\Sigma(K) \subseteq \text{traces}_\Sigma(K) : \{w|(w, \emptyset) \subseteq \text{acc}(K)$. *These are deadlocks of K which can only be avoided by environments that avoid w.*

When TS on Σ defines the behaviour of K, we sometimes write $\text{traces}_\Sigma(TS)$ *instead of* $\text{traces}_\Sigma(K)$*, etc.*

We define traces to be downwards closed set and thus eliminate the effect of the application of the maximal progress rules. The maximal progress rule is useful for effective execution, whereas traces and refusals are used to define the satisfaction relation. Downwards closing traces normalises the behaviours, but does not change the properties satisfied by a component as *all* sequences must satisfy the property and smaller traces don't add inconsistencies.

For each trace w, exists a refusal set B if there exist in TS an execution for w', $w < w'$ to a state q in which a subset of B is refused. We consider traces, acceptance/refusal sets corresponding to an open semantics. E.g. $acc_\Sigma(K)$ contains any action that is accepted in K after w and that *may be accepted* in a system containing K. This open semantics is sufficient, as we want to verify contracts defining an assumption on the context of K, such that we always verify a closed system in which the open and the closed semantics coincide.

Definition 11 (Deadlock freedom of a specification). *Let K be a component. Then, K is (locally) deadlock free if* $\text{dead}(K) = \emptyset$ *that is, if there are no deadlocks in TS.*

Property 5. Let K be a component as above. Then, we have:
- $\underline{traces_\Sigma(K)}; \Sigma \cap \overline{traces_\Sigma(K)} \subseteq REF_\Sigma(K)$
- $\overline{traces_\Sigma(K)}; \Sigma \subseteq \overline{traces_\Sigma(K)}$
- the traces, acceptance, refusals of the internal and the external semantics are the same up to the relabelling (and the abstraction) defined by $\dashv$.

2.2 Comparison and Satisfaction Relations

We define first a comparison relation between behaviours, adequate for the intended property verification, in the sense that smaller models satisfy more properties and larger properties are satisfied by more models.

More precisely, define a preorder that only compares transition systems with respect to some given interface. This, because we are interested in the comparison between components that only differ by their behaviour. Comparing components by comparing their interfaces is an equally interesting problem but not addressed in this paper.

Definition 12 (Preorder and equivalence on behaviours). *Let K be a component and $(\Sigma, <, \#)$ its internal or external interaction set. Let TS, TS' be transition systems on Σ.*

We define the preorder relation $\preceq$ on transition systems with respect to Σ:

- $TS \preceq_\Sigma TS'$, *iff*
  1. $\text{traces}_\Sigma(TS) \subseteq \text{traces}_\Sigma(TS')$ *and*
  2. $\text{ref}_\Sigma(TS) \supseteq \text{ref}_\Sigma(TS')_{|\text{traces}_\Sigma(TS)}$ *where* $\text{ref}_\Sigma(TS')_{|\text{traces}(TS)} = \{(w, B) \in \text{ref}_\Sigma(TS') \mid w \in \text{traces}_\Sigma(TS)\}$
- $TS \approx_\Sigma TS'$ *iff* $TS \preceq_\Sigma TS'$ *and* $TS' \preceq_\Sigma TS$
- *The preorder and equivalence on components K and K' with interaction set Σ and behaviour defined by TS, respectively TS' is straightforward:* $K \preceq K'$ *iff* $TS \preceq_\Sigma TS'$ *and* $K \approx K'$ *iff* $TS \approx_\Sigma TS'$.

Property 6 (Minimal and Maximal behaviours for an interface). Under the same conditions as previously, That is the interaction set $(\Sigma, <, \#)$ for one of the interfaces of a component

- the smallest component, called $dead_\Sigma$ is defined by any transition system TS which has $\{\epsilon\}$ as its set of traces and refuses everything after ϵ. This means *dead* is locally deadlocking
- the largest component, called $true_\Sigma$ is defined by any transition system TS which has Σ^* as its set of traces and an epty refusal set. Thus, *true* has no local deadlock but if no interaction in Σ is complete, then *true* may deadlock in a non cooperative environment
- For K defined by any behaviour TS
  - $dead_\Sigma \preceq K$
  - $K \preceq true_\Sigma$

The satisfaction relation expresses that a component K with behaviour TS_K has a property expressed by a transition system TS where TS_K is defined on $(\Sigma, <, \#)$ and TS on $\Sigma' \subseteq \Sigma$.

Definition 13 (Property for an interaction model). *Let $(\Sigma, <, \#)$ be an interaction set and let TS be a transition system on Σ', a subset of Σ that is downwards closed in Σ. That is $(\Sigma', <, \#)$ is a sub interaction set of $(\Sigma, <, \#)$. Then, TS represents a property for Σ, respectively for a component with an interface having $(\Sigma, <, \#)$ as its interaction set.*

We compare now a behaviour TS defined on Σ with a property for Σ, TS' on Σ'. In order to do so, we simply project the traces and refusals of TS on Σ'.

Definition 14 (Projection). *Let be $(\Sigma, <, \#)$ an interaction set and $(\Sigma', <', \#')$ a sub interaction set. We define the projection $proj(TS, \Sigma')$ of TS to Σ', by*

- $\text{traces}_{\Sigma'}(TS) = \text{traces}_\Sigma(TS)_{|\Sigma'}$
- $\text{ref}_{\Sigma'}(TS) = \text{ref}_\Sigma(TS)_{|\Sigma'}$

We do not redefine the other semantic sets as they are derived from the set of traces and refusals.

Definition 15 (Satisfaction relation). *Let TS on $(\Sigma, <, \#)$ be the behaviour of a component K and TS_P on $(\Sigma', <, \#)$ a property P for Σ. Then,*

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

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

Definition 16 (Composition of properties). *Let TS_i be transition systems on $(\Sigma'_i, <, \#)$ defining properties for $(\Sigma_i, <, \#)$.*

- *the product $TS_1 \|_{CM} TS_2$ obtained by a composition model yielding synchronisation on related actions and interleaving on others ($a \in \Sigma_1$ such that exists $b \in \Sigma_2$ and $a < b$ or $b < a$ typed* sync *and a and b are connected by a connector), where for $a < b$, the interaction $a \cdot b$ is then mapped by $\dashv$ to a in the external view, such that the product is a transition system on $\Sigma_1 \cup \Sigma_2$ which is a subinteraction set of Σ. We denote the product $TS_1 \wedge TS_2$.*
- *If TS_i are deterministic the product $TS_1 \|_{CM'} TS_2$ obtained by a composition model connecting as for $\wedge$ all related actions by a connector, but where all individual interactions are considered of type* trig*, and where $a < b$, the interaction $a \cdot b$ is then mapped by $\dashv$ to b in the external view. We denote the product $TS_1 \vee TS_2$.*

Notice that constructing $TS_1 \vee TS_2$ yields exactly the external choice $TS_1 \uplus TS_2$ of CSP [Hoa84] in the case that Σ_i are unstructured. Here we componentise properties for composition of properties. For effectively verifying properties we will also represent the satisfaction relation as a composition with a particular composition model.

Property 7. Let TS, TS' be transition systems on $(\Sigma, <, \#)$ defining components K, K' and $TS_P, TS_{P'}$ on $(\Sigma', <, \#)$ defining properties P and P' for Σ. Then,

- $K \models P$ implies $traces_{\Sigma'}(TS) \subseteq traces_{\Sigma'}(P)$ and $ref_{\Sigma'}(P) \subseteq ref_{\Sigma'}(TS)$, more precisely, every trace that may be refused by P must be refused by K.
- Call a component K deterministic if for each $w \in traces_{\Sigma}(K)$, $w \notin REF_{\Sigma}(K)$, which means that K has a deterministic transition relation.

  If TS_P is deterministic, then $K \models P$ *if and only if* $traces_{\Sigma'}(TS) \subseteq traces_{\Sigma'}(P)$.
- if $P \preceq P'$ and $K \models P$, then $K \models P'$, that is, larger properties are satisfied by more components.
- if $K' \preceq K$ and $K \models P$, then $K' \models P'$, that is, smaller components satisfy more properties, in particular, more deterministic components satisfy more properties.
- $K \models P$ 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.

Now, let TS_i, TS_i' on $(\Sigma_i, <_i, \#_i)$ define the behaviour of components K_i, K_i' with interfaces $Int_i = (\mathcal{P}_i, <_i, \#_i)$, TS_P^i on $(\Sigma_1^i, <, \#)$ define properties P^i for Σ_1. Let CM be a composition model on Int_i and P a property on the interaction set Σ defined by CM. Then,

- if $K_1 \preceq K_1'$ then $K_1 \|_{CM} K_2 \preceq K_1' \|_{CM} K_2$, that is, $\preceq$ is preserved by composition.
- $K_1 \models P$, then $K_1 \|_{CM} K_2 \models P$ where $K_1 \|_{CM} K_2$ represents the internal view of the behaviour. On the external view of $K_1 \|_{CM} K_2$ holds the property P' obtained by mapping interactions in Σ onto external ports which is more abstract.
- $K_1 \models P^1$ and $K_1 \models P^2$ iff $K_1 \models P^1 \wedge P^2$, that is $\wedge$ represents indead conjunction on properties (for a common trace of P^1 and P^2, $P^1 \wedge P^2$ may refuse exactly those traces that may be refused by at least one of P^1 or P^2)
- $K_1 \models P^1$ or $K_1 \models P^2$ iff $K_1 \models P^1 \vee P^2$, that is, $\vee$ represents indead disjunction on properties

2.3 Decomposition and Recomposition of Components

A composition model is not a unique representation for an interaction set. As it is shown in [BS07], there are generally alternative ways of defining connectors on a set of ports for obtaining a given product interaction set Σ.

We have defined in addition the ports and connectors also the notion of interaction model of BIP, as it is simple and contains enough information for deriving several useful properties (it defines the semantics). A components may be defined by providing just an interaction model and a behaviour. The ports are only used for defining the way in which component may be composed.

[BS07] provides the following useful theorem allowing for a component K which is a composition of components K_i to construct the composition models allowing to represent K as a composition of one of the components K_i with a component K' grouping all the other subcomponents.

Consequently, this allows us obtaining for any component the composition model relating K_i to its environment.

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 any set of ports

This allows the decomposition of a global interaction model on an interaction model of on each of its parts. It yields then a composition model for each part and a global composition models composing the parts. This is enough for defining in a closed system the interaction model between any component in the component hierarchy and "the rest of the system".

Definition 17 (A component and its environment). *Let* $Int(K) = (\mathcal{P}, <, \#)$ *be the external interface of a component* K*.*

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

Then, the internal structure of the component S_K *obtained simply by composing* K *with its environment* K_E *according to definition 7 as* $CM_{EK} = (\{K, E_K\}, (\Sigma, <, \#), Ports \cup \mathcal{P}_E \cup Con)$*.*

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 the given composition models.*

As the internal structure of K *is of the form* $(\{K_i\}, CM, \dashv, \mathcal{P})$*, where the internal composition model is* $CM = (\{Int_i\}, (\Sigma, <, \#), \bigcup_i \mathcal{P}_i \cup \mathcal{P}_{CON})$*.*

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

- *the set of ports* $\mathcal{P}_{CM'}$ *is a set of ports defined by a union of 3 sets of connectors:* $\mathcal{P}_i$*,* $CON_{Ei} \subseteq \bigcup_{j \neq i} \mathcal{P}_j \cup \mathcal{P}_E$ *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* Σ' *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 again straightforward.*

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* $CM_{K_i} = (\{K_i, K_{Ei}\}, (\Sigma_{Ei}, <_{Ei}, \#_{Ei}), Ports_i \cup \mathcal{P}_{Ei} \cup CON_{i-Ei})$ *such that* Σ_{Ei} *is obtained by renaming interactions of* Σ *in terms of* $Ports_{Ei}$*.*

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 [BC07$^+$,BB$^+$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 17 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 leaf component or implicitly defined by the set of leaf components. In the context of contract based reasoning, we want to be able to do some reasoning without having already defined all the leaf components and/or their behaviour.

Definition 18 (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 19 (Rich component (HRC)). *A rich component is of the form* $((\{K_i\}, CM, \dashv, \mathcal{P}_K), (\mathcal{P}_E, CM_{EK}), CONTR)$ *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 the composition model connecting* K *to its environment as defined in Definition 17; 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* CM_{EK} *is given.*
- $\{K_i\}$ *are HRC and* $(\{Int_i\}, CM, \dashv, \mathcal{P})$ *is defined like an internal interface of a hierarchical component or alternatively,* $((\mathcal{P}, <, \#), TS)$ *defines an atomic component without a defined substructure.*
- $CONTR$ *is a set of contracts of the form* (A_i, G_i) *where* A_i *is defined on* $\mathcal{P}_E$ *and* G_i *is defined on* $\mathcal{P}$ *.*

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 explicitly.

Given the structure of an HRC K we can now consider the environment K_E of K like any other component. How a valid K_E can be constructed is defined in Section 2.3.

3.2 Compositional Verification of HRC

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 consider here the case of the satisfaction of a single contract. Multiple contracts can be validated independently of each other.

Definition 20 (Satisfaction of contracts). *Let K be a rich component with an external interface $Int_K = (\mathcal{P}, <, \#)$ and K_E an environment with $Int_E = (\mathcal{P}_E, <, \#)$ and composition model CM_{EK} between K and E 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 behaviour A.

According to the satisfaction relation of definition 15, as G is defined as a property on the interaction set of K, and the behaviour of S_K is of the form $A\|_{CON}TS$ on the composition of the interaction set of K and E_K, $A\|_{CON}TS \models G$ means that the projection of $A\|_{CON}TS$ onto the interaction set 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 $Int_E = (\mathcal{P}_E, <, \#)$ and composition model CM_{EK} between K and E. Suppose that the transition system TS on $\mathcal{P}$ represents the implementation of K.*

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

$$A\|_{CM_{EK}}TS \preceq_{\mathcal{P}} A\|_{CM_{EK}}G$$

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 $A\|_{CM_{EK}}TS$, we can conclude that if the system defined by $A\|_{CM_{EK}}TS$ satisfies property G then $A\|_{CM_{EK}}TS \preceq_{Int_K} G$ by property 7; together with $S_K \preceq_{Int_E} A$ which is due to monotonicity, this allows to derive $A\|_{CM_{EK}}TS \preceq A\|_{CM_{EK}}G$. Using the property saying that $G \models G$ for deterministic G and the fact that $A\|TS \preceq_{Int_K} G$ one obtains the second assertion.*

This important property expresses the fact that G defines an upper bound on all components that satisfy G in any environment satisfying A; and it allows the use of a simple proof rule for verifying contract dominance.

Definition 21. *Let K be a hierarchical rich component with a structure of the form $((str(K_i), (CM, \dashv \mathcal{P}_K), (\mathcal{P}_E, CM_{EK}), (A, G))$ such that each $str(K_i)$ defines a contract (A_i, G_i). Then (A, G) dominates the set of contracts $\{(A_i, G_i)\}$ in the context of K iff*

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

Remember 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 [BB+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. There, the semantics is defined in terms of sets of traces and the contract (A', G') is defined using negations (complements of trace sets); here, we show the soundness of a similar proof rule, without using negation.

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

Then (A, G) dominates the set of contracts $\{(A_i, G_i)\}$ in the context of K 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 i, the component S_K defined as a composition of K_i with K_{Ei} obtained from $\mathcal{P}_{Ei}$ and Definition 17, 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 to environments satisfying A_i can be eliminated as A_i is already guaranteed by A and by G_j; indeed, the second item implies that $A \| (A_1 \| G_1)_p \| ... \| (A_n \| G_n)_p$ 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 interpretation of the result in the interface of K_i.

4 Handling Verification Conditions Contructively

We have defined a framework for architecture and system modelling based on the BIP framework 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 appropriate 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,GG$^+$07] 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 will hopefully lead to interesting results.

We have defined components which have in their interface not only the possible interactions 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 means to explore asynchronous transition systems by using either partial order reduction or maximal progress rules. We also envisage to use a Petrinet like representation of asynchronous transition systems, for example UML activity diagrams to represent concurrency in a more explicit manner.

The abstraction defined by the use of typed connectors is particularly interstsing if we succeed to construct on-the-fly reductions of composed behaviours. But we envisage also an approach based on incremantal contruction and abstraction as in [GLS96].

References

BB$^+$07. Badouel, E., Benveniste, A., Bozga, M., Caillaud, B., Constant, O., Josko, B., Ma, Q., Passerone, R., Skipper, M.: SPEEDS meta-model syntax and draft semantics. Deliverable D2.1c (February 2007)

BBS06. Basu, A., Bozga, M., Sifakis, J.: 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)

BC07$^+$. Bozga, M., Constant, O., Skipper, M., Ma, Q.: SPEEDS meta-model syntax and static semantics. Deliverable D2.1b (January 2007)

BS07. Bliudze, S., Sifakis, J.: The algebra of connectors structuring interaction in BIP. Techreport, Verimag (February 2007)

RB$^+$01. de Roever, W.P., de Boer, F., Hannemann, U., Hooman, J., Lakhnech, Y., Poel, M., Zwiers, J.: Concurrency Verification: Introduction to Compositional and Noncompositional Methods. In: Nr 54 in Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge (2001)

GG+07. Gößler, G., Graf, S., Majster-Cederbaum, M., Martens, M., Sifakis, J.: An approach to modeling and verification of component based systems. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, Springer, Heidelberg (2007)

GS03. Gößler, G., Sifakis, J.: Component-based construction of deadlock-free systems. In: Pandya, P.K., Radhakrishnan, J. (eds.) FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science. LNCS, vol. 2914, Springer, Heidelberg (2003)

GS05. Goessler, G., Sifakis, J.: Composition for component-based modeling. Science of Computer Programming, pp. 161–183 (March 2005)

GLS96. Graf, S., Lüttgen, G., Steffen, B.: Compositional Minimisation of Finite State Systems using Interface Specifications. In: Formal Aspects of Computation, vol. 8, Appeared as Passauer Informatik Bericht MIP-9505 (1996)

Hoa84. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs (1984)

WN95. Winskel, G., Nielsen, M.: Models for concurrency, vol. 4. Oxford Univ. Press, Oxford (1995)