Contract-Based Verification of Hierarchical Systems of Components

Sophie Quinton and Susanne Graf
Université Joseph Fourier, VERIMAG
{graf, quinton}@imag.fr

1 Introduction and related work

Our aim is to provide a framework for contract-based reasoning for the component-based design of complex hierarchically defined systems.

Compositional verification is necessary to scale to systems of arbitrary complexity. One well-studied approach to it is Assume/Guarantee reasoning ([6]) which proposes proof methods that allow to verify a system – defined as a composition of components – by (1) expressing the global property as a composition of local properties and (2) verifying the local properties on individual components, taking into account some assumptions on their environment that must be proven to hold.

In the context of system design, models of the components may not be available at early stages of development, and one may want to reason about assumptions and properties only. In [5], a notion of interface specification has been introduced in terms of I/O automata,where the constraint expressed on *inputs* is interpreted as the *assumption* and the constraint on outputs as the *guarantee* (under the given assumption). When dealing with such contracts, relevant notions are those of *composability* and *compatibility* of sets of contracts and of *refinement* between contracts. *Composability* is a purely syntactic criterion on interfaces and *compatibility* expresses the fact that there exists an environment preventing the composition to enter some predefined *error* state. Finally, in the context of contracts, *refinement* is sometimes also called *dominance* [3], and a refined contract is a contract that requires more from the implementation and less from the environment. Separation between assumption and guarantee has been proposed in [9] while interface automata have been enriched with modalities in [10]. In this paper we present a framework encompassing those interface theories and offering and heterogeneity of interaction thanks to the BIP framework.

In [7, 2, 4], the framework BIP (Behavior, Interaction, Priority) has been proposed for component-based design and verification. In BIP, the concept of input/output is replaced by the more general concept of multi-party interaction which allows each of the involved interaction partners to impose constraints on when the interaction may take place and thus does not require input completeness.

That means, contracts, as defined by interface specifications, are not needed to avoid *error* states. But it is still relevant to have a framework for assume/guarantee reasoning which allows to (1) characterize a set of environments E for which the composition $\gamma(K, E)$ is consistent, that is, deadlock-free, or (2) those for which $\gamma(K, E)$ guarantees that K behaves as specified by some property B where B may express a stronger property than a BIP behavior specification. This allows deriving global properties of a composed system in a compositional manner, as well as definition of frameworks for reusability and tailoring of components.

This paper is organized as follows: in section 2, we generalize the definitions of contract and dominance introduced in [3] by introducing a framework for contract-based reasoning parameterized by notions of *interface*, *composition*, *behavior semantics* and *refinement in a context*. Compositions of interfaces is defined within an abstract version of BIP without variables, values nor priorities. We then provide a proof rule for *dominance* that is the parameters.

In section 3, we apply the general framework to behaviors expressed as modal specifications [8], which are powerful enough to deal with loose specifications and properties implying some progress ensurance in absence of input enabledness: in a modal transition system, a *must*-transition represents a progress guarantee of all its implementations while *may*-transitions define safety properties as in usual transition systems. We define a notion of composition under context which takes into account (1) the composition operator γ and (2) which guarantees the soundness of apparent circular reasoning.

We finally present a possible use of this framework that is being implemented using the Maude rewrite engine [1]. This method addresses the problem of verifying that a global contract is satisfied by a given composition of behaviors. If the global guarantee G is "projectable" with respect to the interaction model defined by the hierarchical component structure, it automatically generates and discharges assumptions for the projection of G onto properties of components that must be verified in the context. The interested

reader is referred to [11] for a more comprehenseive description of the approach.

2 Frameworks for contract-based reasoning

2.1 The BIP framework

BIP [7, 2, 4] is a component framework for constructing systems by superposing three layers of modeling: Behavior, Interaction, and Priority. This implies a clear separation between behavior and structure as composition at the priority and interaction level does not depend on the behavior level. Besides, it is general enough to encompass heterogeneity of interaction, allowing description of synchronous as well as asynchrous and heterogeneous systems.

In this paper, we build upon an abstract version of BIP without variables nor priorities. The interface of a component is a set of ports, and a composition operator γ is defined by a set of *legal* interactions (multi-partner rendezvous) given as a set of connectors of the algebra $\mathcal{AC}(P)$ introduced in [4]. In the following, when referring explicitely to the set of legal interactions of γ, we will denote it $\mathcal{L}(\gamma)$. Interactions are sets of ports. For readability's sake, we represent union of two interactions α_1 and α_2 by $\alpha_1|\alpha_2$.

BIP also defines how operators γ are composed. For example, in Figure 2.3, the composition of B_1, E_1 and E_2 is represented in (a) as $\gamma(B_2, \gamma_E(E_1, E_2))$ and in (b) as $\gamma_2(\gamma_1(B_2, E_1), E_2))$. As BIP ensures associativity and commutativity of composition, it is always possible to compute composition operators to compose a set of components in any order.

While in [7, 2] behaviors are expressed as LTSs, we will in section 3 we use MTSs. As the definition of framework for contract-based reasoning given in the next section is independent of the representation of behaviors, our results can easily be applied to the usual BIP behavior semantics.

2.2 A generic framework for contract-based reasoning

Definition 2.1 (Contract-based framework) *A contract-based verification framework is given by a tuple $(\mathcal{B}, \mathcal{P}, \Gamma, \|\,.\,\|, \theta)$, where:*

- *$\mathcal{B}$ is a set of* behaviors*; each behavior $B \in \mathcal{B}$ has as interface a set of* ports *denoted $\mathcal{P}_B$.*
- *$\mathcal{P} = \bigcup_{B\in\mathcal{B}} \mathcal{P}_\mathcal{B}$.*
- *Γ is a set of BIP* composition operators *on subsets of $\mathcal{P}$.*
- *$\|\,.\,\| : \Gamma \times 2^\mathcal{B} \longrightarrow \mathcal{B}$ is a partial function defining a* behavior semantics *for the composition of behaviors, ensuring associativity and commutativity of composition operators as defined in [7]. For $\gamma \in \Gamma$ and $B_1, \ldots, B_n \in \mathcal{B}$, $\|(\gamma, (B_1, \ldots, B_n))\|$, denoted $\gamma(B_1, \ldots, B_n)$, is defined iff γ is defined on $\bigsqcup_{i=1}^n \mathcal{P}_{B_i}$.*
- *$\theta : \mathcal{B} \times \Gamma \longrightarrow 2^{\mathcal{B}\times\mathcal{B}}$ is a partial function that is* monotonic w.r.t composition *as defined below. For each pair (E, γ) such that γ is a composition operator defined on $\mathcal{P}_E \sqcup \mathcal{P}_B$ for some set of ports $\mathcal{P}_B$, $\theta(E, \gamma)$, denoted $\sqsubseteq_{E,\gamma}$, is a* preorder *(a reflexive and transitive binary relation) over the set of behaviors with associated set of ports $\mathcal{P}_B$. θ is called* refinement in a context*.*

A behavior B is said to be *on* $\mathcal{P}$ iff $\mathcal{P}_B = \mathcal{P}$. In the following we suppose given a contract-based verification framework $(\mathcal{B}, \mathcal{P}, \Gamma, \|\,.\,\|, \theta)$.

Definition 2.2 (Context for an interface) *Let $P \in 2^\mathcal{P}$ be an interface. A context for P is a pair (E, γ) where E is such that $P \cap \mathcal{P}_E = \emptyset$ and γ is a composition operator defined on $P \sqcup \mathcal{P}_E$.*

Definition 2.3 (Monotony) *θ is monotonic w.r.t. composition iff for any interface P, any context (E, γ) for P, the following implication always holds. If there exist γ_E on $\mathcal{P}_E$ and E_1, E_2 such that $\mathcal{P}_E = \mathcal{P}_{E_1} \sqcup \mathcal{P}_{E_2}$ and $E = \gamma_E(E_1, E_2)$, then for all B_1, B_2 behaviors on P: $B_1 \sqsubseteq_{\gamma_E(E_1,E_2),\gamma} B_2$ implies $\gamma_1(B_1, E_1) \sqsubseteq_{E_2,\gamma_2} \gamma_1(B_2, E_1)$, where γ_1 and γ_2 are calculated from γ and γ_E for respectively $P \sqcup \mathcal{P}_{E_1}$ and $P \sqcup \mathcal{P}_{E_1} \sqcup \mathcal{P}_{E_2}$.*

Figure 1 gives a graphic representation of the above implication. We expect any function defining refinement in a context in a contract-based verification framework to be monotonic w.r.t. composition. Indeed otherwise refinement in a context does not convey any information about the global system.

Definition 2.4 (Refinement) *If a behavior B_1 refines a behavior B_2 in all contexts, i.e. for all (E, γ), $B_1 \sqsubseteq_{E,\gamma} B_2$, we use the notation $B_1 \sqsubseteq B_2$.*

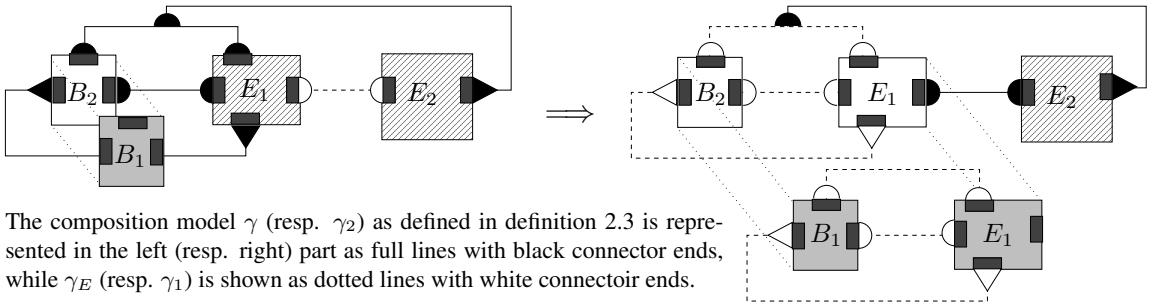
Definition 2.5 (Contract for an interface) *A contract $\mathcal{C}$ for an interface P consists of:*

- *a context $\mathcal{E} = (A, \gamma)$ for P, where A is called the* assumption
- *a behavior G on P called the* guarantee

We write $C = (A, \gamma, G)$ rather than $((A, \gamma), G)$.

Definition 2.6 (Satisfaction) *Let $P \in 2^\mathcal{P}$ be a set of ports. Let $\mathcal{C} = (A, \gamma, G)$ be a contract for P and B a behavior on P. We say that B* satisfies *$\mathcal{C}$, denoted $B \models \mathcal{C}$, iff $B \sqsubseteq_{A,\gamma} G$.*

The satisfaction relation depends on the verification framework. Intuitively, a component K satisfies a contract $\mathcal{C} = (A, \gamma, G)$ if K "behaves according to" G provided that the environment behaves according to A.



The composition model γ (resp. γ_2) as defined in definition 2.3 is represented in the left (resp. right) part as full lines with black connector ends, while γ_E (resp. γ_1) is shown as dotted lines with white connectoir ends.

Figure 1. Monotony of refinement

Definition 2.7 (Dominance) *Let $\{P_i\}_{i=1}^n \in 2^{\mathcal{P}}$ be a family of pairwise disjoint sets of ports, with $P = \bigsqcup_{i=1}^n P_i$. Let $\mathcal{C}$ be a contract for P and for each $i = 1..n$, $\mathcal{C}_i$ be a contract for P_i. Let γ be a composition operator on P. We say that $\mathcal{C}$ dominates $\{\mathcal{C}_i\}_{i=1}^n$ w.r.t. γ iff $\forall B_1, \dots, B_n \in \mathcal{B}$:*

$$\forall i, P_{B_i} = P_i \ \wedge \ B_i \models \mathcal{C}_i \implies \gamma(B_1, \dots, B_n) \models \mathcal{C}$$

A contract $\mathcal{C}$ dominates a set of contracts $\{\mathcal{C}_i\}_{i=1}^n$ w.r.t a composition operator γ if any set of behaviors satisfying the "subcontracts", when composed using γ, yields a component satisfying $\mathcal{C}$.

2.3 A generic condition for apparent circular reasoning and dominance

Definition 2.8 (Apparent circular reasoning) *A framework $(\mathcal{B}, \mathcal{P}, \Gamma, \|\,.\,\|, \theta)$ allows apparent circular reasoning iff for any given interface P, behavior B on P, context (E, γ) for P and contract $\mathcal{C} = (A, \gamma, G)$ for P we have:*

$$B \sqsubseteq_{A,\gamma} G \wedge E \sqsubseteq_{G,\gamma} A \implies B \sqsubseteq_{E,\gamma} G$$

This means that every component with a contract (A, γ, G), if integrated into an environment that is compatible with γ and that ensures A, can be replaced in the system by a component with the same interface and with G as behavior. Everything that can be proved in the new system can also be proved in the original system.

Theorem 2.9 *Let $(\mathcal{B}, \mathcal{P}, \Gamma, \|\,.\,\|, \theta)$ be a framework allowing circular reasoning. Let $\{P_i\}_{i=1}^n \in 2^{\mathcal{P}}$ be a family of pairwise disjoint interfaces, with $P = \bigsqcup_{i=1}^n P_i$. Let $\mathcal{C} = (A, \gamma, G)$ be a contract for P, and for each $i = 1..n$, $\mathcal{C}_i = (A_i, \gamma_i, G_i)$ be a contract for P_i. Let γ_I be a composition operator on P. Then the following conditions are sufficient to prove that $\mathcal{C}$ dominates $\{\mathcal{C}_i\}_{i=1}^n$ w.r.t. γ:*

$$\begin{cases} \gamma_I(G_1, \dots, G_n) \models \mathcal{C} \\ \forall i, \gamma(A, \gamma_{I\setminus i}(G_1, \dots, G_{i-1}, G_{i+1}, \dots, G_n)) \models \mathcal{C}^{-1} \end{cases}$$

with $\gamma_{I\setminus i}$ standing for the restriction of γ_I to $P \setminus P_i$ and $\mathcal{C}^{-1} = (G_i, \gamma_i, A_i)$.

This proof rule can thus be applied in any framework allowing apparent circular reasoning.

3 Application to a framework based on modal transition systems

3.1 A framework based on MTSs

Modal transition systems ([8]) are labeled transition systems where transitions have in addition a modality, either *must* or *may*.We suppose a set of ports $\mathcal{P}$.

Definition 3.1 (Modal Transition System) *A modal transition system (MTS) on an interface P is a structure $B = (Q, q^0, P, \delta)$, where Q is a set of states, $q^0 \in Q$ is an initial state, $P \in 2^{\mathcal{P}}$ is an interface, and δ consists of two relations $\longrightarrow$ and $\dashrightarrow$ on $Q \times 2^P \times Q$ such that:*

$$\forall q^1, q^2 \in Q, \forall \alpha \in 2^P, \ q^1 \xrightarrow{\alpha} q^2 \implies q^1 \overset{\alpha}{\dashrightarrow} q^2$$

The condition that *must*-transitions may also be interpreted as *may*-transitions ensures *modal consistency* of the MTSs ([8]). Notice that transitions are labeled by sets of ports and not by ports. Indeed, if an MTS represents the behavior of a component K, it may have to deal with interactions involving several ports of K.

Definition 3.2 (Composition of MTSs) *Let $\{P_i\}_{i=1}^n$ be a family of pairwise disjoint interfaces and $P = \bigsqcup_{i=1}^n P_i$. Let γ be a composition operator on P with $\mathcal{L}(\gamma)$ the set of* legal *interactions of γ. For $i = 1..n$, let be the MTSs $B_i = (Q_i, q_i^0, P_i, \delta^i)$. The composition of $B_1, \dots, B_n$ with γ, denoted $\|_\gamma\ (B_1, \dots, B_n)$, is an MTS (Q, q^0, P, δ) such that $Q = \prod_{i=1}^n Q_i$, $q^0 = (q_1^0, \dots, q_n^0)$ and $\longrightarrow$ and $\dashrightarrow$ are defined by: $\forall \alpha \in \mathcal{L}(\gamma)$, $\forall q^1 = (q_1^1, \dots, q_n^1), q^2 = (q_1^2, \dots, q_n^2) \in Q$,*

- $q^1 \xrightarrow{\alpha} q^2$ *iff* $\forall i, q_i^1 \xrightarrow{\alpha_i} q_i^2$
- $q^1 \overset{\alpha}{\dashrightarrow} q^2$ *iff* $\forall i, q_i^1 \xrightarrow{\alpha_i} q_i^2$ *or* $q_i^1 \overset{\alpha_i}{\dashrightarrow} q_i^2$

where $\alpha_i = \alpha \cap P_i$ and with the convention that $\forall q, q \xrightarrow{\emptyset} q$ and $q \overset{\emptyset}{\dashrightarrow} q$.

Definition 3.3 (Refinement of MTS) *Refinement $\preceq$ is a binary relation defined on MTS with the same interface. An MTS $(Q_1, q_1^0, P, \delta^1)$ refines an MTS $(Q_2, q_2^0, P, \delta^2)$ if there*

exists a relation $\mathcal{R} \subseteq Q_1 \times Q_2$ such that $\mathcal{R}$ is a simulation for $(Q_1, q_1^0, P, \dashrightarrow^1)$ and $(Q_2, q_2^0, P, \dashrightarrow^2)$, and $\mathcal{R}^{-1}$ is a simulation for $(Q_2, q_2^0, P, \longrightarrow^2)$ and $(Q_1, q_1^0, P, \longrightarrow^1)$.

This is the usual definition of refinement for MTSs ([8]). We introduce now a refinement in a context allowing apparent circular reasoning for MTSs.

Definition 3.4 (Enabled interaction) *Let B be an MTS with $B = (Q, q^0, P, \delta)$. We say that an interaction $\alpha \in 2^P$* must *(resp.* may*) be enabled in a state $q \in Q$ if there exists $q' \in Q$ s.t. $q \xrightarrow{\alpha} q'$ (resp. $q \overset{\alpha}{\dashrightarrow} q'$). The function $en_\Box$ (resp. $en_\Diamond$) : $Q \longrightarrow 2^{2^P}$ returns for any state $q \in Q$ the set of interactions that* must *(resp.* may*) be enabled in q.*

Definition 3.5 (Refinement in a context) *Let P be an interface, (E, γ) a context for P with $E = (Q_E, q_E^0, P, \delta^E)$. Let $B_1 = (Q_1, q_1^0, P, \delta^1)$ and $B_2 = (Q_2, q_2^0, P, \delta^2)$ be MTSs on P. B_1 refines B_2 in the context of E, denoted $B_1 \sqsubseteq_{E,\gamma} B_2$, iff there exists a relation $\mathcal{R} \subseteq (Q_1 \times Q_E) \times Q_2$ such that $(q_1^0, q_E^0)\,\mathcal{R}\,q_2^0$ and for all $q_1 \in Q_1, q_2 \in Q_2, q_E \in Q_E$, $(q_1, q_E)\,\mathcal{R}\,q_2$ implies:*

1. $en_\Diamond(q_1) \cap \gamma \subseteq en_\Diamond(q_2) \cap \gamma$
2. $\alpha_P|\alpha_E \in \gamma \wedge q_1 \overset{\alpha_P}{\dashrightarrow} q_1' \wedge \alpha_E \in en_\Diamond(q_E)$
   $\implies \exists q_2'$ *s.t. a)* $q_2 \overset{\alpha_P}{\dashrightarrow} q_2'$
   b) $\forall q_E',\ q_E \overset{\alpha_E}{\dashrightarrow} q_E' \implies (q_1', q_E')\,\mathcal{R}\,q_2'$
3. $en_\Box(q_2) \cap \gamma \subseteq en_\Box(q_1) \cap \gamma$
4. $\alpha_P|\alpha_E \in \gamma \wedge q_2 \xrightarrow{\alpha_P} q_2' \wedge \alpha_E \in en_\Box(q_E)$
   $\implies \exists q_1'$ *s.t. a)* $q_1 \xrightarrow{\alpha_P} q_1'$
   b) $\forall q_E',\ q_E \xrightarrow{\alpha_E} q_E' \implies (q_1', q_E')\,\mathcal{R}\,q_2'$

Theorem 3.6 *The usual refinement relation $\preceq$ for MTSs is equal to refinement in all contexts $\sqsubseteq$.*

Theorem 3.7 *Let $\mathcal{B}$ be the set of MTSs on $\mathcal{P}$, Γ the set of composition operators defined on all subsets of $\mathcal{P}$, $\|$ and θ respectively the composition for MTSs and the refinement in a context defined above. Then $(\mathcal{B}, \mathcal{P}, \Gamma, \|, \theta)$ is a contract-based verification framework allowing apparent circular reasoning.*

3.2 Consistency of a behavior with a composition operator and a partition

As will be shown, not all properties can be ensured by a composition of behaviors using a given composition operator if the partition of the set of ports is fixed. It can be useful to detect such cases as early as possible in the design flow. Hence the notion of consistency of a behavior with a composition operator and a partition.

In the following definitions, P always denotes an interface, $B = (Q, q^0, P, \delta)$ an MTS on P, γ a composition operator on P and $\{P_i\}_{i=1}^n$ a partition of P.

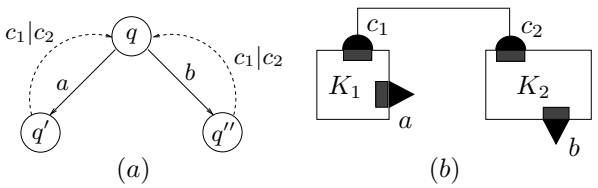


Figure 2. Consistency of a behavior with a composition operator and a partition.

Definition 3.8 (Consistency and decomposability) *B is* consistent *with (resp.* decomposable *w.r.t.) γ and $\{P_i\}$ if there exist some MTSs $B_1, \ldots, B_n$ on respectively $P_1, \ldots, P_n$ such that $\gamma(B_1, \ldots, B_n) \sqsubseteq B$ (resp. $\gamma(B_1, \ldots, B_n) = B$).*

Figure 2 shows a behavior (a) defined on $P = \{a, b, c_1, c_2\}$ that is not consistent with the composition operator $\gamma = \{\{a\}, \{b\}, \{c_1|c_2\}\}$ and a partition $\{\{a, c_1\}, \{b, c_2\}\}$ shown in (b). Indeed, as a must be enabled in q, then considering that q is of the form (q_1, q_2), with q_1 (resp. q_2) a state of some behavior B_1 (resp. B_2), then a must be enabled in q_1. Thus in q'', after b has been fired, B_1 is still in q_1, so a must still be enabled, which is not allowed in (a). Inconsistency here is due to some properties of the composition of behaviors. We develop this idea in the sequel.

Definition 3.9 (Ports affected by an interaction) *Let $\alpha \in 2^P$ be an interaction. We fix $P_\alpha = \{P_i \mid \alpha \cap P_i \neq \emptyset\}$. The set of ports affected by α is* $\mathrm{aff}(\alpha) = \bigcup_{I \in P_\alpha} I$.

Property 3.10 (Decomposability conditions) *If B is decomposable w.r.t. γ and $\{P_i\}$, then the following properties hold for all $q, q' \in Q$ and $\alpha \in 2^P$:*

1. $q \overset{\alpha}{\dashrightarrow} q' \implies \forall a \in en_\Diamond(q) \cup en_\Diamond(q'),\ a \cap P \backslash \mathrm{aff}(\alpha) \in en_\Diamond(q) \cap en_\Diamond(q')$
2. $q \xrightarrow{\alpha} q' \implies \forall a \in en_\Box(q) \cup en_\Box(q'),\ a \cap P \backslash \mathrm{aff}(\alpha) \in en_\Box(q) \cap en_\Box(q')$
3. $\forall \alpha_1, \alpha_2 \in en_\Box(q), \alpha_1|\alpha_2 \in \mathcal{L}(\gamma) \Longrightarrow \alpha_1|\alpha_2 \in en_\Box(q)$

Conditions 1 and 2 come from the fact that firing a transition labeled by α does not affect the sets of enabled interactions of the components that are not involved in α, since they remain in the same local state as a property of the composition defined in 3.2.Condition 3 is also a consequence of composition: if α_1 and α_2 are enabled, then so is $\alpha_1|\alpha_2$.

A behavior B is consistent with a composition operator γ and a partition $\{P_i\}_{i=1}^n$ iff there exists a behavior B' that is decomposable w.r.t. γ and $\{P_i\}_{i=1}^n$ and such that $B' \sqsubseteq B$. Thus, deciding consistency of B boils down to deciding if there exists such a B'. As any such B' satisfies the decomposability conditions, there are syntactic constraints

that it must satisfy. Due to lack of space they are not described here (see [11]). We use them in an algorithm that eliminates some states that are unreachable in a well-formed refinement as well as *may*-transitions that will always disappear during refinement while others are transformed into *must*-transitions. This allows detecting some inconsistencies such as in Figure 2. If the set of behaviors B' on P that are decomposable w.r.t. γ and $\{P_i\}_{i=1}^n$ and such that $B' \sqsubseteq B$ has a greatest element, then we can find it and it is complete as described below.

Definition 3.11 (Completeness) *B is complete w.r.t. γ and $\{P_i\}_{i=1}^n$ iff the decomposability conditions hold and $\forall q \in Q, \forall \alpha_1, \alpha_2 \in en_\diamond(q), \alpha_1|\alpha_2 \in \mathcal{L}(\gamma) \Longrightarrow \alpha_1|\alpha_2 \in en_\diamond(q)$.*

Behaviors that are complete w.r.t a composition operator and a partition have the nice property that they are decomposable and that a decomposition can easily be computed, as explained below. Let us underline here the role of composition operators in completeness.

Definition 3.12 (Projection) *Let $P' \subseteq P$ be an interface and $B = (Q, q^0, P, \delta)$ an MTS. The projection of B on P', denoted $\Pi_{P'}(B)$ is $(Q, q^0, \{\tau\} \cup P', \delta')$ where δ' is defined as δ except that labels have been projected on $\{\tau\} \cup P'$.*

Theorem 3.13 *If B is complete w.r.t. γ and $\{P_i\}_{i=1}^n$, then B is also decomposable w.r.t. γ and $\{P_i\}_{i=1}^n$ and $B = \gamma(\Pi_{P_1}(B), \dots, \Pi_{P_n}(B))$.*

3.3 Verifying a global contract directly from behaviors

In section 2.3, we have given a sufficient condition to prove dominance between contracts. We sketch (see [11] for more) here a methodology to generate such contracts from a global contract and the behaviors (or specifications) of the atomic components. This method only applies to behaviors that are *complete* w.r.t the global composition operator, as it is based on projection of the global property. Due to the expressivity of the interaction layer of BIP, this still represents an interesting class of systems.

Our verification problem is the following: let P be an interface, $\mathcal{C} = (A, \gamma, G)$ a contract for P, γ_I a composition operator on P and $\{P_i\}_{i=1}^n$ a partition of P. For all $i = 1..n$, let B_i be a behavior on P_i and $B = \gamma_I(B_1, \dots, B_n)$. We want to know if $\gamma_I(B_1, \dots, B_n) \models \mathcal{C}$. If G is complete w.r.t. γ and $\{P_i\}_{i=1}^n$, then it is sufficient that all B_i refine $G_i = \Pi_{P_i}(G)$ in the context of A and the other components.

From that statement we build for all B_i an assumption A_i on its environment such that $B_i \sqsubseteq_{A_i,\gamma_i} G_i$. This construction is adapted from [12] but our approach is different here in so far it does not involve any global guarantee. The A_i are deterministic. Furthermore, if control is not distributed, they are minimal in the sense that if the environment does not verify them, then B will in some refinement violate G. If the A_i are complete w.r.t. their context, it is possible to iterate the process by projecting them as done for G.

4 Conclusion and future work

In this paper, we add to the usual notion of contract a structural part specifying the composition operator used to compose the component and its environment. We provide a framework for compositional verification including a proof rule for dominance between contracts based on apparent circular reasoning. We also briefly describe a consistency condition and a method based on assumption generation to generate or refine contracts.

An implementation of this contract-based verification framework, using the rewriting tool Maude [1] is currently under development. Plain MTSs do not allow to express such properties as "one of the following interactions must be enabled". It is therefore not always possible to prove deadlock-freedom of a component-based system. Using disjunctive modal transition systems could solve this issue. Besides, we would like to investigate how such a methodology applies to the full BIP framework with priorities and data transfer through connectors.

References

[1] Maude website. http://maude.cs.uiuc.edu/.

[2] A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in bip. In *SEFM*, pages 3–12, 2006.

[3] A. Benveniste, B. Caillaud, and R. Passerone. A generic model of contracts for embedded systems. *CoRR*, abs/0706.1456, 2007.

[4] S. Bliudze and J. Sifakis. The algebra of connectors: structuring interaction in bip. In *EMSOFT*, pages 11–20, 2007.

[5] L. de Alfaro and T. A. Henzinger. Interface automata. In *ESEC / SIGSOFT FSE*, pages 109–120, 2001.

[6] W. P. de Roever, F. S. de Boer, U. Hannemann, J. Hooman, Y. Lakhnech, M. Poel, and J. Zwiers. *Concurrency Verification: Introduction to Compositional and Noncompositional Methods*, volume 54 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[7] G. Gößler and J. Sifakis. Composition for component-based modeling. *Sci. Comput. Program.*, 55(1-3):161–183, 2005.

[8] K. G. Larsen. Modal specifications. In *Automatic Verification Methods for Finite State Systems*, pages 232–246, 1989.

[9] K. G. Larsen, U. Nyman, and A. Wasowski. Interface input/output automata. In *FM*, pages 82–97, 2006.

[10] K. G. Larsen, U. Nyman, and A. Wasowski. Modal I/O automata for interface and product line theories. In *ESOP*, pages 64–79, 2007.

[11] S. Quinton and S. Graf. Contract-based verification of hierarchical systems of components. Technical report, Verimag, 2008.

[12] J.-B. Raclet. Residual for component specifications. Technical Report 6196, INRIA, May 2007.