How to Verify and Exploit a Refinement of Component-based Systems

Olga Kouchnarenko, Arnaud Lanoix

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

How to Verify and Exploit a Refinement of Component-based Systems

Olga Kouchnarenko — Arnaud Lanoix

N° 5898

Avril 2006

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--5898--FR+ENG

How to Verify and Exploit a Refinement of Component-based Systems

Olga Kouchnarenko* , Arnaud Lanoix†

Thème SYM — Systèmes symboliques
Projet Mosel

Rapport de recherche n 5898 — Avril 2006 — 17 pages

Abstract: In order to deal with the verification of large systems, compositional approaches postpone in part the problem of combinatorial explosion during model exploration. The purpose of the work we present in this paper is to establish a compositional framework in which the verification may proceed through a refinement-based specification and a component-based verification approaches.

A constraint synchronised product operator enables us 1) an automated compositional verification of a component-based system refinement relation, and 2) safety LTL properties of a whole system from local safety LTL properties of its components. The main advantage of our specification and verification approaches is that LTL properties are preserved through composition and refinement.

Key-words: component-based systems, modules, refinement, LTL properties, composition, verification

* LIFC, FRE 2661 CNRS - Université Franche-Comté, Besançon, France
† LORIA, Mosel & Dedale - CNRS, Nancy, France

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,

Comment vérifier et exploiter le raffinement des systèmes à composants

Résumé : Les approches compositionnelles permettent de résoudre en partie le problème de l'explosion combinatoire issu de l'exploration du modèle du système, lors de la vérification de systèmes de grande taille. L'objectif du travail présenté dans cet article est d'établir un cadre de vérification compositionnelle combinant une approche de spécification par raffinement avec une approche de vérification par composants.

Un opérateur de produit synchronisé contraint permet de vérifier automatiquement et de manière compositionnelle 1) une relation de raffinement au niveau du système à composants, 2) des propriétés LTL de sûreté du système complet à partir de propriétés locales de ses composants. L'avantage principal de notre approche de spécification et de vérification est que les propriétés LTL sont préservées autant par la composition que par le raffinement.

Mots-clés : systèmes à composants, modules, raffinement, propriétés LTL, composition, vérification

1 Introduction

Nowadays, formal methods are used in various areas, from avionics and automatic systems to telecommunication, transportation and manufacturing systems. However, the increasing size and complexity of these systems make their specification and verification difficult. Compositional reasoning is a way to master this problem.

The purpose of the work we present in this paper is to establish a compositional framework in which an algorithmic verification of a refinement of component-based systems by model exploration of components can be associated with the verification of LTL properties. In our compositional framework, we give ways (see Fig. 1) to preserve LTL properties through:

1. The composition operator for preserving safety LTL properties, meaning that a property satisfied by a separate component is also satisfied by a whole component-based system.

2. The refinement relation for preserving both safety and liveness LTL properties, meaning that a property established for an abstract system model is ensured when the system is refined to a richer level of details.

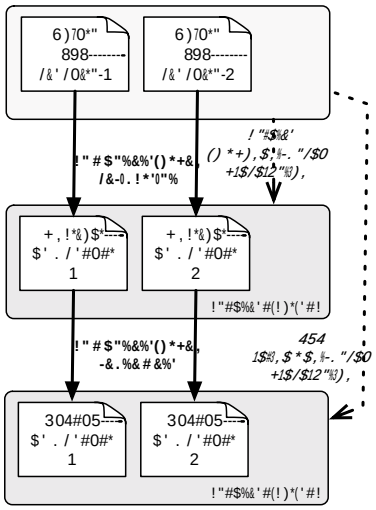


Figure 1: Verification Principle

To achieve the goal of compositional verification and to model synchronous and asynchronous behaviours of components, we define two operators: a composition of the modules and a constraint synchronised product of transition systems.

We show that the modules [12, 13, 2] – subsystems sharing variables – whose composition is often used in a concurrent setting, are suitable to compositionally verify a kind of τ-simulation, called the weak refinement. Unfortunately, this model does not allow analysing the strict refinement – a divergence-sensitive completed τ-simulation – from the separate refinements of its modules. That is why we introduce a constraint synchronised product operator. Moreover, the semantics of the component-based systems using this operator makes it possible to verify the strict refinement more efficiently.

The main result of this paper is the theorem claiming that the strict refinement of a component-based system can be established by checking the weak refinement of its expanded components viewed as the modules. The main advantage of the component-based refinement we have been developing is that it allows us to master the complexity of specifications and verifications with a step by step development process without building the whole system. All steps of our compositional approach have been implemented in an analysis tool called SynCo [9].

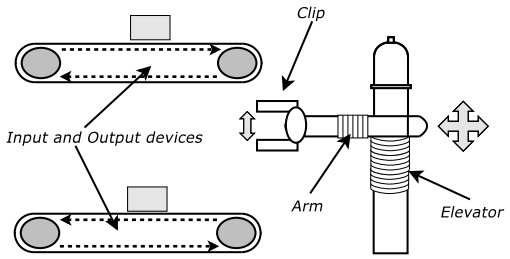


Figure 2: Production Cell

The main concepts of the paper are illustrated on an example of a simple controller of a production cell moving pieces from an input device to an output device. A pictorial representation of this running example is given in Fig.2. The cell is composed of an arm having horizontal moves, a clip, and an elevator moving vertically. Sensors notify the controller about the production cell changes.

This paper is organised as follows. After giving preliminary notions, we recall in Section 2, the semantics of our refinement relation and its properties. Then Section 3 presents the modules, their composition and the weak refinement of the composition of the modules, called modular refinement. In Section 4, the constraint synchronised product is introduced to specify component-based systems, and the modular refinement is used to establish the strict refinement of component-based systems more efficiently.

2 Preliminaries: LTS Refinement and LTL properties

We introduce labelled transition systems to specify component behaviours and properties. Transition systems we consider are interpreted over a finite set of variables V. Let AP_V be a set of atomic propositions over V.

Definition 1 (Labelled Transition System (LTS)) *A LTS S is a tuple (Q, Q_0, E, T, V, l) where*
– Q is a set of states,
– $Q_0 \subseteq Q$ is a set of initial states,
– E is a finite set of transition labels,
– $T \subseteq Q \times E \times Q$ is a labelled transition relation,
– V is a set of variables, and
– $l : Q \rightarrow 2^{AP_V}$ is a total function that labels each state with the set of atomic propositions true in that state; we call l the interpretation.

We consider a (finite or infinite) sequence of states $\sigma = q_0, q_1, \ldots$ [1] in Q. σ is a path of S iff $\forall i.(i \geq 0 \Rightarrow \exists e_i.(e_i \in E \wedge (q_i, e_i, q_{i+1}) \in T))$[2]. Given a path σ, we denote by $tr(\sigma)$ its trace $e_0, e_1, \ldots$. $\Sigma(S)$ designates the set of paths of S. A state q_i is reachable from q_0 iff there exists a path of the form $\sigma = q_0, q_1, \ldots, q_i$.

In this paper, dynamic properties of systems are expressed by formulae of propositional Linear Temporal Logic (LTL) [15] given by the following grammar: ϕ,$\phi' ::= ap \mid \phi \vee \phi' \mid \neg\phi \mid \bigcirc\phi \mid \phi \mathcal{U} \phi'$, where $ap \in AP_V$.

Definition 2 (LTL semantics) *Given LTL properties ϕ,ϕ' and a path σ, we define ϕ to be satisfied at $i \geq 0$ on a path σ, written $(\sigma, i) \models \phi$, as follows.*
– $(\sigma, i) \models ap$ iff $ap \in l((\sigma, i))$
– $(\sigma, i) \models \neg\phi$ iff it is not true that $(\sigma, i) \models \phi$
– $(\sigma, i) \models \phi \vee \phi'$ iff $(\sigma, i) \models \phi$ or $(\sigma, i) \models \phi'$
– $(\sigma, i) \models \bigcirc\phi$ iff $(\sigma, i+1) \models \phi$
– $(\sigma, i) \models \phi \mathcal{U} \phi'$ iff $\exists j.(j \geq i \wedge (\sigma, j) \models \phi' \wedge \forall k.(i \leq k < j \Rightarrow (\sigma, k) \models \phi))$

We also use the notations $\Diamond\phi \equiv true \mathcal{U} \phi$, $\Box\phi \equiv \neg\Diamond\neg\phi$, and $\phi \mathcal{W} \phi' \equiv \Box\phi \vee \phi \mathcal{U} \phi'$. A LTL property ϕ is satisfied by S when $\forall\sigma.(\sigma \in \Sigma(S) \wedge (\sigma, 0) \in Q_0 \Rightarrow (\sigma, 0) \models \phi)$.

Moreover, we often consider the set $SP_V \stackrel{\text{def}}{=} \{sp, sp_1, sp_2, \ldots\}$ of state propositions over V defined by $sp, sp' ::= ap \mid sp \vee sp' \mid \neg sp$, where $ap \in AP_V$. In this setting, an *invariance property* is a proposition $sp \in SP_V$ satisfied by every state of S, i.e. $\forall q.\ (q \in Q \Rightarrow q \models sp)$, written $S \models sp$.

To handle a product of LTSs, the satisfaction of a state proposition sp by a state is extended to tuples of states. For example, a formula $sp \in SP_{V_1 \cup V_2}$ is satisfied by (q_1, q_2) iff either $sp \in SP_{V_1}$ and $q_1 \models sp$, or $sp \in SP_{V_2}$ and $q_2 \models sp$.

[1] (σ, i) designates the i^{th} state of σ.
[2] An element of T is also denoted $q \stackrel{e}{\rightarrow} q'$.

In this paper, we exploit the system top-down refinement relation we have introduced in [5]. Let $SA = (Q_A, Q_{0A}, E_A, T_A, V_A, l_A)$ be an abstract LTS and $SR = (Q_R, Q_{0R}, E_R, T_R, V_R, l_R)$ a more detailed LTS. The syntactic features of the refinement are as follows. First, refinement introduces new actions, so $E_A \subseteq E_R$. Second, some new variables can be introduced and the old ones are renamed: $V_A \cap V_R = \varnothing$. Third, a propositional calculus formula gp over $V_A \cup V_R$, called *gluing predicate*, links variables of both LTSs.

Definition 3 (Glued states) *Let $gp \in SP_{V_A \cup V_R}$ be a gluing predicate. The state $q_R \in Q_R$ is* glued *to $q_A \in Q_A$ w.r.t. gp, written $q_R \ \mu \ q_A$, iff $(q_A, q_R) \models gp$.*

The refinement relation with the semantic features below is a restriction of μ.

1. The transitions of SR, the labels of which are in E_A (i.e. labelled by the "old" labels) are kept.
2. New transitions introduced during the refinement design (i.e. labelled in $E_R \setminus E_A$) are considered as being non-observable; they are labelled by τ and called τ-transitions in the system SR.
3. Moreover, new transitions should not introduce new deadlocks.
4. Finally, new transitions should not take control forever. So, infinite sequences of τ-transitions, i.e. τ-cycles, are forbidden.

Definition 4 (Refinement Relation) *Let SA and SR be two LTSs, and $e \in E_A$. Let μ be the gluing relation. The refinement relation $\eta \subseteq Q_R \times Q_A$ is defined as the greatest binary relation included in μ and satisfying the following clauses:*
1) strict transition refinement $(q_R \ \eta \ q_A \ \wedge \ q_R \xrightarrow{e} q'_R \in T_R) \Rightarrow \exists q'_A.\ (q_A \xrightarrow{e} q'_A \in T_A \ \wedge \ q'_R \ \eta \ q'_A)$,
2) stuttering transition refinement $(q_R \ \eta \ q_A \ \wedge \ q_R \xrightarrow{\tau} q'_R \in T_R) \Rightarrow (q'_R \ \eta \ q_A)$,
3) lack of new deadlocks $(q_R \ \eta \ q_A \ \wedge \ q_R \not\rightarrow) \Rightarrow (q_A \not\rightarrow)$[3],
4) lack of τ-cycles $q_R \ \eta \ q_A \Rightarrow (\forall \sigma.\ \sigma \in \Sigma(q_R) \ \Rightarrow \ tr(\sigma) \neq \tau^\omega)$.

The relation η is a partial order. From now on, we say that SR *refines* SA, written $SR \sqsubseteq_\eta SA$, when $\forall q_R.(q_R \in Q_R \Rightarrow \exists q_A.\ (q_A \in Q_A \ \wedge \ q_R \ \eta \ q_A))$.

It has been shown in [5] that the refinement relation can be classified as a *divergence stability respecting completed simulation* in the van Glabbeek' spectrum [18]. Consequently, it is more expressive than the trace inclusion, and hence the closed systems refinement [17] by Shankar. An algorithmic verification of the relation η can be done by model exploration of the refined system which complexity order is $O(|SR|)$ where $|SR| = |Q_R| + |T_R|$.

Figure 3 gives a refinement of a part of the controller. In the abstract $Clip_A$ system, the sensor has two positions, *open* and *close*, whereas in the refined $Clip_R$ system, there

[3] We note $q \not\rightarrow$ when $\forall q', e.\ (q' \in Q \wedge e \in E \Rightarrow q \xrightarrow{e} q' \notin T)$.

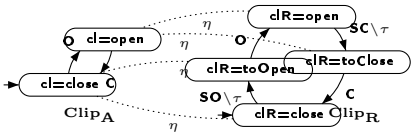


Figure 3: $Clip_R \sqsubseteq_\eta Clip_A$

are two more positions, *toOpen* and *toClose*. The relation η between $Clip_R$ and $Clip_A$ is established, so $Clip_R \sqsubseteq_\eta Clip_A$.

The refinement relation η being a kind of τ-simulation, it preserves safety properties. Moreover, we have shown in [8] that the refinement relation η preserves LTL – safety and liveness – properties. In other words, any abstract LTL property satisfied by an abstract system SA is, modulo gp, satisfied by a corresponding refined system SR. Here the satisfaction relation $\models_{gp}$ taking gp into account, can be defined by induction on the structure of a formula ϕ like $\models$ in Definition 2. For example, $sp_A \in SP_{V_A}$ is satisfied by a state $q_R \in Q_R$, modulo gp, written $q_r \models_{gp} sp_A$, iff $\bigwedge_{ap \in l_R(q_R)} ap \wedge gp \Rightarrow sp_A$.

Theorem 1 (LTL Component-based Preservation [8]) *Let SA and SR be two LTSs. Let gp be their gluing predicate, and ϕ_A an abstract LTL property. If T_A is total then*

$$\frac{SR \sqsubseteq_\eta SA,\ SA \models \phi_A}{SR \models_{gp} \phi_A}$$

otherwise, only safety LTL properties are preserved.

To handle a product of LTSs which brings about new deadlocks and cycles of τ-transitions, the weak refinement relation is defined by

Definition 5 (Weak Refinement Relation) *Let SA and SR be two LTSs, μ the gluing relation. Let $D \subseteq Q_R$ be the deadlock set. The weak refinement $\rho \subseteq Q_R \times Q_A$ is the greatest binary relation included in μ and satisfying the following clauses:*
***1) strict refinement** and **2) stuttering refinement** from definition 4,*
3') old or new deadlocks $((q_R\ \rho\ q_A \wedge\ q_R \nrightarrow) \Rightarrow ((q_A \nrightarrow \wedge q_R \notin D) \vee (q_A \rightarrow \wedge q_R \in D)))$

We say that SR *weakly refines* SA, written $SR \sqsubseteq_\rho^D SA$, when $\forall q_R.(q_R \in Q_R \Rightarrow \exists q_A.\ (q_A \in Q_A \wedge\ q_R\ \rho\ q_A))$. Like η, the weak refinement relation ρ is a kind of τ-simulation too. Consequently, ρ preserves safety LTL properties.

The definition above does not mention the τ-cycles. Let $div^\tau(SR, SA)$ be a predicate meaning that SR contains some τ-cycles w.r.t. SA. It is easy to see that the refinement and the weak refinement are linked by

Property 1 (Refinement vs. Weak Refinement)

$$SR \sqsubseteq_\eta SA \textit{ iff } SR \sqsubseteq_\rho^D SA \textit{ et } D = \varnothing \textit{ et } \neg\ div^\tau(SR, SA)$$

3 Modular Refinement

In a concurrent setting, systems are often modelled using a parallel composition of subsystems sharing variables, called *modules* in [12, 13, 2]. We consider the modules sharing global variables from the set V. Let M^1 and M^2 be two modules. We introduce the parallel composition of M^1 and M^2, denoted $M^1 \| M^2$, which is a module that has exactly the behaviours of M^1 and M^2. *Local* behaviours of M^1 (resp. M^2) are the transitions labelled in $E^1 \setminus E^2$ (resp. $E^2 \setminus E^1$), whereas *global* behaviours are the transitions labelled in $E^1 \cup E^2$.

Definition 6 (Parallel Composition) *Let* $M^1 = (Q^1, Q_0^1, E^1, T^1, V, l^1)$ *and* $M^2 = (Q^2, Q_0^2, E^2, T^2, V, l^2)$ *be two modules. Their composition is defined by* $M^1 \| M^2 = (Q, Q_0, E, T, V, l)$, *where*
- $Q = Q^1 \cup Q^2$,
- $Q_0 = Q_0^1 \cup Q_0^2$,
- $E = E^1 \cup E^2$,
- $T = T^1 \cup T^2$,
- $\forall q \in Q^1 \cup Q^2, l(q) = \begin{cases} l^1(q), \text{ if } q \in Q^1 \\ l^2(q), \text{ if } q \in Q^2 \end{cases}$

Figure 4 gives two modules M_{Clip} and M_{Arm} of the controller. They interact by modifying the global variables cl and ar to give the parallel composition $M_{Clip} \| M_{Arm}$.

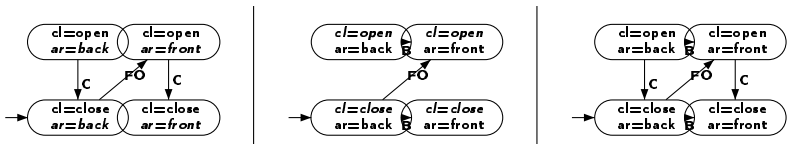


Figure 4: Parallel composition of $M_{Clip} \| M_{Arm}$

Property 2 (Commutativity of $\|$) $\forall M^1, M^2.\ M^1 \| M^2 = M^2 \| M^1$.

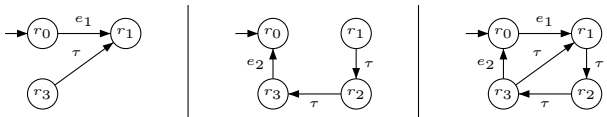
Property 3 (Associativity of $\|$) $\forall M^1, M^2, M^3. (M^1 \| M^2) \| M^3 = M^1 \| (M^2 \| M^3)$.

Property 4 (Invariance Preservation of $\|$) *Given* $sp \in SP_V$, *if* $M^1 \models sp$ *and* $M^2 \models sp$ *then* $M^1 \| M^2 \models sp$.

The above property can be extended to dynamic invariants, that are LTL properties of the form $\Box(sp_1 \Rightarrow sp_2)$.

Instead of verifying the refinement of the composition of the modules, we propose verifying the refinement of each module separately reaching a conclusion about the refinement of

their parallel composition automatically. Unfortunately, the strict refinement relation cannot be compositionally established because of interleaving of τ-transitions in the modules composition as illustrated in Fig. 5.



(a) M^1 (b) M^2 (c) $M^1 \| M^2$

Figure 5: Interleaving of τ-transitions

Let us examine the weak refinement relation clauses. It is easy to see that the τ-simulation (the strict transition refinement and the stuttering transition refinement) can be compositionally verified since modules only use shared global variables in V.

For compositional deadlock checking, the idea is as follows. Suppose that $MR^1 \sqsubseteq_\rho^{D_1} MA^1$ and $MR^2 \sqsubseteq_\rho^{D_2} MA^2$. A state in $D_1 \cup D_2$ is a deadlock in $MR^1 \| MR^2$ iff either it is a deadlock in both modules, or a deadlock in a module and not a state in the other one. This deadlock reduction, denoted $D_1 \triangle D_2$, can be computed by

$$D_1 \triangle D_2 = (D_1 \cap D_2) \cup (D_1 \setminus Q_R^2) \cup (D_2 \setminus Q_R^1) \tag{1}$$

Property 5 (Associativity of $\triangle$) $(D_1 \triangle D_2) \triangle D_3 = D_1 \triangle (D_2 \triangle D_3)$.

Now we are ready to establish – in a compositional manner – the weak refinement of the composition of the modules.

Theorem 2 (Modular Refinement) *Let $MA^1 \| MA^2$ and $MR^1 \| MR^2$ be modules compositions. One has*

$$\frac{MR^1 \sqsubseteq_\rho^{D_1} MA^1, \quad MR^2 \sqsubseteq_\rho^{D_2} MA^2}{MR^1 \| MR^2 \sqsubseteq_\rho^{D_1 \triangle D_2} MA^1 \| MA^2}$$

Proof is in Appendix A.1.

Theorem 2 can be generalised to n modules thanks to Property 3 and Property 5.

4 Component-based Refinement

The model of the modules is not well-adapted to verify the strict refinement in a compositional way. To this end, a constraint synchronised product is introduced allowing specifying the synchronised behaviours of components.

Let consider independent interacting components. The whole component-based system is a rearrangement of its separate part, i.e. the components and their interactions. Let '$-$' denote the fictive action "skip". To specify interactions between components, a synchronisation set syn is defined.

Definition 7 (Synchronisation Set) *Let S^1 and S^2 be two components. A synchronisation set syn is a subset of $\{(e_1, e_2)/sp |\ e_1 \in E^1 \cup \{-\} \wedge e_2 \in E^2 \cup \{-\} \wedge sp \in SP_{V_1 \cup V_2}\}$.*

In words, the set syn contains tuples of labels (e_1, e_2) with feasibility conditions sp scheduling the behaviours of components. Given the whole system (S^1, S^2, syn), the rearrangement of its parts is described by

Definition 8 (Constraint Synchronised Product) *Let (S^1, S^2, syn) be a component-based system. The constraint synchronised product $S^1 \times_{syn} S^2$, is the tuple (Q, Q_0, E, T, V, l) where*
- $Q \subseteq Q^1 \times Q^2$,
- $Q_0 \subseteq Q_0^1 \times Q_0^2$,
- $E = \{(e_1, e_2) \mid (e_1, e_2)/sp \in syn\}$,
- $V = V^1 \cup V^2$,
- $l((q_1, q_2)) = l^1(q_1) \cup l^2(q_2)$,
- $T \subseteq Q \times E \times Q$ *is obtained by :*

[8.1] $(q_1, q_2) \xrightarrow{(e_1,-)} (q_1', q_2) \in T$ *if* $(e_1, -)/sp \in syn$, $q_1 \xrightarrow{e_1} q_1' \in T^1$ *and* $(q_1, q_2) \models sp$,
[8.2] $(q_1, q_2) \xrightarrow{(-,e_2)} (q_1, q_2') \in T$ *if* $(-, e_2)/sp \in syn$, $q_2 \xrightarrow{e_2} q_2' \in T^2$ *and* $(q_1, q_2) \models sp$, *or*
[8.3] $(q_1, q_2) \xrightarrow{(e_1,e_2)} (q_1', q_2') \in T$ *if* $(e_1, e_2)/sp \in syn$, $q_1 \xrightarrow{e_1} q_1' \in T^1$, $q_2 \xrightarrow{e_2} q_2' \in T^2$ *and* $(q_1, q_2) \models sp$

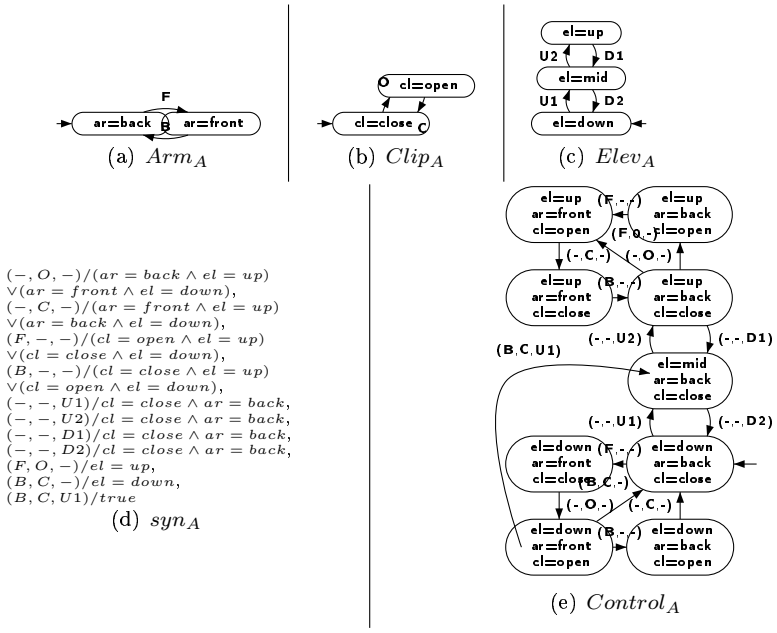
Definition 8 can be easily extended to n components. Notice that the synchronised product above is more expressive than the well-known synchronised product by Arnold and Nivat [4, 3] because of feasibility conditions. Indeed, each transition of our product operator can involve either joint transitions of components or single transition of one component.

Notice that there is a τ-simulation between the whole system and a component. Consequently, the constraint synchronised product preserves safety LTL properties from local components to the component-based system. Furthermore, the conjunction of local safety LTL properties is ensured for the entire system.

Property 6 (Safety LTL Component-based Preservation) *Let ϕ_1 and ϕ_2 be safety LTL properties. If $S^1 \models \phi_1$ and $S^2 \models \phi_2$ then $(S^1, S^2, syn) \models \phi_1 \wedge \phi_2$*

In addition, every LTL property being the conjunction of a safety property and a liveness property [1], its safety part is preserved by our product operator.

For our running example, Fig. 6 presents the components Arm_A, $Clip_A$ and $Elev_A$, the synchronisation set syn_A, and the computed entire system $Control_A = (Arm_A, Clip_A, Elev_A, syn_A)$.



(a) Arm_A (b) $Clip_A$ (c) $Elev_A$

$(-, O, -)/(ar = back \wedge el = up)$
$\vee(ar = front \wedge el = down)$,
$(-, C, -)/(ar = front \wedge el = up)$
$\vee(ar = back \wedge el = down)$,
$(F, -, -)/(cl = open \wedge el = up)$
$\vee(cl = close \wedge el = down)$,
$(B, -, -)/(cl = close \wedge el = up)$
$\vee(cl = open \wedge el = down)$,
$(-, -, U1)/cl = close \wedge ar = back$,
$(-, -, U2)/cl = close \wedge ar = back$,
$(-, -, D1)/cl = close \wedge ar = back$,
$(-, -, D2)/cl = close \wedge ar = back$,
$(F, O, -)/el = up$,
$(B, C, -)/el = down$,
$(B, C, U1)/true$

(d) syn_A (e) $Control_A$

Figure 6: $(Arm_A, Clip_A, Elev_A, syn_A)$: Components and Synchronisation Set

Each component is a context-free component. However, for the compositional refinement verification, its environment has to be taken into account. For that purpose, we define an *expanded* component, that is a component in the context of the other components.

Definition 9 (Expanded component) *Let (S^1, S^2, syn) be a component-based system. The expanded component $[S^1]$ corresponding to S^1 is defined by*

$$[S^1] \stackrel{def}{=} S^1 \times_{[syn]_{S^1}} S^2$$

where $[syn]_{S^1} \stackrel{def}{=} \{(e_1, e_2)/sp \mid ((e_1, e_2)/sp) \in syn \wedge e_1 \in E^1 \wedge e_2 \in E^2 \cup \{-\}\}$

In the previous definition, the synchronisation set is restricted to conserve only behaviours involving the considered component. The expanded component $[S^2]$ is similarly defined. Notice that both expanded components are modules (cf. Section 3) defined over the same set of global variables $V^1 \cup V^2$. The parallel composition of these modules gives rise to the whole component-based system.

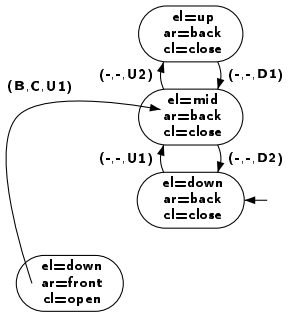


Figure 7: $[Elev_A]$

Property 7 (Component-based System vs. Modules)

$$(S^1, S^2, syn) \;=\; S^1 \times_{syn} S^2 \;=\; [S_1] \parallel [S_2]$$

Figure 7 gives the expanded component $[Eleve_A]$ computed from $(Arm_A, Clip_A, Elev_A, syn_A)$. To illustrate Property 7, the other expanded components $[Arm_A]$ and $[Clip_A]$ can be built, and we have the whole system $[Arm_A] \parallel [Clip_A] \parallel [Elev_A]$ without building it.

We show now that the constraint synchronised product semantics makes it possible to compositionally verify the strict refinement relation; notably, by resolving the interleaving of τ-transitions and reducing deadlocks more efficiently.

Let (SA^1, SA^2, syn_A) and (SR^1, SR^2, syn_R) be two component-based systems. The lack of τ-cycles in the whole component-based system can be derived from its local checking for each component separately.

Property 8 (Lack of τ-cycles)

$$\frac{\neg\ div^\tau(SR^1, SA^1), \quad \neg\ div^\tau(SR^2,\ SA^2)}{\neg\ div^\tau((SR^1, SR^2, syn_R), (SA^1, SA^2, syn_A))}$$

The deadlock reduction can be done more efficiently too. Intuitively, a state inducing a new deadlock in an expanded component does not induce a deadlock in the whole system if there exists an expanded component where this state is not a deadlock state. The checking of whether a state is in an expanded component state space can be done by studying the synchronisation set. Suppose $[SR^1] \sqsubseteq_\rho^{D_1} [SA^1]$. The reduced deadlock set RD_1 is defined by

$$RD_1 \stackrel{\text{def}}{=} D_1 \setminus \{q \mid q \in D_1 \wedge \exists e_2.\ (e_2 \in E_R^2 \wedge (-, e_2)/sp \in syn_R \wedge q \models sp)\} \qquad (2)$$

We want to emphasise that for an expanded component, the deadlock reduction is independent from the other expanded components. Property 7 allows us to apply the modular refinement to component-based systems using expanded components as modules.

Theorem 3 (Component-based Refinement) *Let* (SA^1, SA^2, syn_A) *and* (SR^1, SR^2, syn_R) *be two component-based systems. Then*

$$\frac{\neg\ div^\tau(SR^1, SA^1),\ [SR^1] \sqsubseteq_\rho^{D_1} [SA^1],\ RD_1 = \varnothing,\quad \neg\ div^\tau(SR^2, SA^2),\ [SR^2] \sqsubseteq_\rho^{D_2} [SA^2],\ RD_2 = \varnothing}{(SR^1, SR^2, syn_R) \sqsubseteq_\eta (SA^1, SA^2, syn_A)}$$

Proof is in Appendix A.2. Theorem 3 can be given for n components. It provides a compositional refinement verification algorithm based on the computation, for each refined expanded component $[SR^i]$ separately, of the relation ρ. The complexity order of this refinement verification algorithm is $O(\sum_{i=1}^{n} |[SR^i]|)$. However, the greatest memory space used is $max_{i=1}^{n}|[SR^i]|$, at most: the expanded component building, the weak refinement verification and the deadlock reduction can be done sequentially.

5 Conlusion

In areas like telecommunication or manufacturing, complex systems may be derived from initial models by composition and refinement. Composition combines separate parts and refinements add new details for systematically deriving component-based systems where safety properties are preserved. Furthermore, the state explosion can be alleviated by considering the components one by one.

Our compositional framework is well-adapted for studying components instead of the whole system. Indeed, the weak refinement verification of the composition of the modules can be reduced to the weak refinements of its modules. Furthermore, the constraint synchronised product advocated in this paper allows us to establish the strict refinement of the component-based systems by checking the weak refinements of its expanded components viewed as the modules.

Finally, our compositional framework gives ways to postpone the model-checking blow-up. Actually, a safety LTL property is 1) preserved from an abstract component to a whole abstract component-based system, and 2) preserved from that system to a whole refined component-based system (see Fig. 1). The usefulness of our approach is illustrated by our previous results [10, 11] and confirmed by the experimental results [14].

Future work concerns reactive systems. We are going to investigate what model of open systems can be used to obtain a closed system model by adding an environment specification. At that stage, simulation relations should be established again.

Related works. The *assume-guarantee* paradigm (see for instance [6, 7, 16]) is a well-studied framework for compositional verification. The assume-guarantee paradigm requires

the hypotheses on the component environment strong enough to imply any potential constraint. The way out is the *lazy composition* approach by Shankar [17] which works at the level of the specification of component behaviour and discharges proof obligations lazily.

In our approach, the component environment is taken into account by increasing a component model to automatically build an expanded component. Like the lazy composition, the constraint synchronised product allows us to proceed through a refinement-based specification and a component-based verification approaches. The strict refinement relation being a divergence-sensitive completed τ-simulation, we anticipate that it is more expressive than the closed systems refinement by Shankar.

References

[1] B. Alpern and F.B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2:117–126, 1987.

[2] R. Alur and T.A. Henzinger. Reactive modules. *Formal Methods in System Design (FMSD)*, 15(1):7–48, July 1999.

[3] A. Arnold. *Systèmes de transitions finis et sémantique des processus communicants.* Collection Etudes et Recherches en Informatiques. Masson, Paris, 1992.

[4] A. Arnold and M. Nivat. Comportements de processus. In *Actes du Colloque AFCET - Les Mathématiques de l'Informatique*, pages 35–68, 1982.

[5] F. Bellegarde, J. Julliand, and O. Kouchnarenko. Ready-simulation is not ready to express a modular refinement relation. In *Fundamental Aspects of Software Engineering (FASE'00)*, volume 1783 of *LNCS*, pages 266–283. Springer Verlag, April 2000.

[6] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* The MIT Press, 2000.

[7] J.-M. Cobleigh, D. Giannakopoulou, and C. Pasareanu. Learning assumptions for compositional verification. In *9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, volume 2619 of *LNCS*, Warsaw, Poland, April 2003. Springer-Verlag.

[8] C. Darlot, J. Julliand, and O. Kouchnarenko. Refinement preserves PLTL properties. In D. Bert, J. P. Bowen, S. C. King, and M. Walden, editors, *Formal Specification and Development in Z and B (ZB'2003)*, volume 2651 of *LNCS*, Turku, Finland, June 2003. Springer Verlag.

[9] O. Kouchnarenko and A. Lanoix. SynCo: a refinement analysis tool for synchronized component-based systems. In *Tool Exhibition Notes, Formal Methods (FM'03).*

[10] O. Kouchnarenko and A. Lanoix. Refinement and verification of synchronized component-based systems. In K. Araki, S. Gnesi, and Mandrioli D., editors, *Formal*

Methods (FM'03), volume 2805 of *LNCS*, pages 341–358, Pisa, Italy, September 2003. Springer Verlag.

[11] O. Kouchnarenko and A. Lanoix. Verifying invariants of component-based systems through refinement. In C. Rattray, S. Maharaj, and C. Shankland, editors, *Algebraic Methodology and Software Technology (AMAST'04)*, volume 3116 of *LNCS*, pages 289–303, Stirling, Scotland, July 2004. Springer Verlag.

[12] O. Kupferman and M. Y. Vardi. Module checking. In Rajeev Alur and T.A. Henzinger, editors, *Eighth International Conference on Computer Aided Verification CAV*, volume 1102, pages 75–86, New Brunswick, NJ, USA, 1996. Springer Verlag.

[13] O. Kupferman and M. Y. Vardi. Module checking revisited. In *9th International Computer Aided Verification Conference*, pages 36–47, 1997.

[14] A. Lanoix. *Systèmes à composants synchronisés : contributions à la vérification compositionnelle du raffinement et des propriétés.* PhD thesis, Université de Franche-comté, Septembre 2005.

[15] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specifications.* Springer Verlag, 1992.

[16] K.L. McMillan. A methodology for hardware verification using compositional model-checking. *Science of Computer Programming*, 37:279–309, 2000.

[17] N. Shankar. Lazy compositional verification. In *COMPOS'97: Revised Lectures from the International Symposium on Compositionality: The Significant Difference*, pages 541–564, London, UK, 1998. Springer-Verlag.

[18] R.J. van Glabbeek. The linear time-branching time spectrum (extended abstract). In *CONCUR'90*, pages 278–297. Springer-Verlag, 1990.

A Proofs

A.1 Proof of Theorem 2

Let $M \stackrel{\text{def}}{=} \{(q_R, q_A) \mid q_R \in Q_R^1 \cup Q_R^2 \wedge q_A \in Q_A^1 \cup Q_A^2\}$. We show that M verifies all conditions of definition 5 of ρ.

1. *Strict refinement:* suppose $(q_R, e, q'_R) \in T_R$ and $e \in E_A$.

   We must prove there exists $q_A \in Q_A$ and $q'_A \in Q_A$ such that $(q_A, e, q'_A) \in T_A$, $q_R \rho q_A$ and $q'_R \rho q'_A$.

   By definition $E_A = E_A^1 \cup E_A^2$. Proof depends of e.

   $e \in E_A^1$. We have $(q_R, e, q'_R) \in T_R^1$. Since $MR^1 \sqsubseteq_\rho^{D_1} MA^1$, there exists $q_A \in Q_A^1$ and $q'_A \in Q_A^1$ such that $q_R \rho q_A$, $(q_A, e, q'_A) \in T_A^1$ and $q'_R \rho q'_A$. q_A and q'_A belong to $Q_A = Q_A^1 \cup Q_A^2$.

   $e \in E_A^2$. We have $(q_R, e, q'_R) \in T_R^2$. Since $MR^2 \sqsubseteq_\rho^{D_2} MA^2$, there exists $q_A \in Q_A^2$ and $q'_A \in Q_A^2$ such that $q_R \rho q_A$, $(q_A, e, q'_A) \in T_A^2$ and $q'_R \rho q'_A$. q_A and q'_A belong to $Q_A = Q_A^1 \cup Q_A^2$.

2. *stuttering refinement:* suppose $(q_R, \tau, q'_R) \in T_R$.

   We must prove there exists $q_A \in Q_A$ such that $q_R \rho q_A$ and $q'_R \rho q_A$.

   By definition $E_A = E_A^1 \cup E_A^2$ and $E_R = E_R^1 \cup E_R^2$. Proof depends of $e \backslash \tau \in E_R \setminus E_A$.

   $e \in E_R^1 \setminus E_A^1$. We have $(q_R, e\backslash\tau, q'_R) \in T_R^1$. Since $MR^1 \sqsubseteq_\rho^{D_1} MA^1$, there exists $q_A \in Q_A^1$ such that $q_R \rho q_A$ and $q'_R \rho q_A$. q_A belongs to $Q_A = Q_A^1 \cup Q_A^2$.

   $e \in E_R^2 \setminus E_A^2$. We have $(q_R, e\backslash\tau, q'_R) \in T_R^2$. Since $MR^2 \sqsubseteq_\rho^{D_2} MA^2$, there existe $q_A \in Q_A^2$ such that $q_R \rho q_A$ and $q'_R \rho q_A$. q_A belongs to $Q_A = Q_A^1 \cup Q_A^2$.

3. *old or new deadlocks:* suppose $q_R \nrightarrow \in T_R$.

   We must prove there exists $q_A \in Q_A$ such that $q_R \rho_w q_A$ and $q_A \nrightarrow$ or $q_R \in D_1 \triangle D_2$.

   By definition $Q_R = Q_R^1 \cup Q_R^2 = (Q_R^1 \cap Q_R^2) \cup (Q_R^1 \setminus Q_R^2) \cup (Q_R^1 \setminus Q_R^2)$. Proof depends of q_R.

   $q_R \in Q_R^1 \setminus Q_R^2$. Since $MR^1 \sqsubseteq_\rho^{D_1} MA^1$, either there exists $q_A \in Q_A^1$ such that $q_A \nrightarrow$ and $q_R \rho q_A$. q_A belongs to Q_A ; or $q_R \in D_1$ and $q_R \rho q_A$. As $q_R \in Q_R^1 \setminus Q_R^2$ and $q_R \in D_1$, $q_R \in D_1 \setminus Q_R^2$. q_R belongs to $D_1 \triangle D_2$.

   $q_R \in Q_R^2 \setminus Q_R^1$. Since $MR^2 \sqsubseteq_\rho^{D_2} MA^2$, either there exists $q_A \in Q_A^2$ such that $q_A \nrightarrow$ and $q_R \rho q_A$. q_A belongs to Q_A ; or $q_R \in D_2$ and $q_R \rho q_A$. As $q_R \in Q_R^2 \setminus Q_R^1$ and $q_R \in D_2$, $q_R \in D_2 \setminus Q_R^1$. q_R belongs to $D_1 \triangle D_2$.

   $q_R \in Q_R^1 \cap Q_R^2$. Since $MR^1 \sqsubseteq_\rho^{D_1} MA^1$, there exists $q_A \in Q_A^1$ such that either $q_A \nrightarrow$ and $q_R \rho q_A$ (1) or $q_R \rho q_A$ and $q_R \in D_1$ (2).

Since $MR^2 \sqsubseteq_\rho^{D_2} MA^2$, there exists $q_A \in Q_A^2$ such that either $q_A \nrightarrow$ and $q_R \rho q_A$ (3) or $q_R \rho q_A$ and $q_R \in D_2$ (4).
If (1) and (3), there exists $q_A \in Q_A^1 \cup Q_A^2$ such that $q_A \nrightarrow$.
If (1) and (4), there exists $q_A \in Q_A^1$ such that $q_A \nrightarrow$.
If (2) and (3), there exists $q_A \in Q_A^2$ such that $q_A \nrightarrow$.
If (2) and (4), $q_R \in D_1 \cap D_2$: q_R belongs to $D_1 \triangle D_2$.

A.2 Proof of Theorem 3

$[SR^1] \sqsubseteq_\rho^{D_1} [SA^1]$, $[SR^2] \sqsubseteq_\rho^{D_2} [SA^2]$ and Property 7 imply $(SR^1, SR^2, syn_R) \sqsubseteq_\rho^{D_1 \triangle D_2} (SA^1, SA^2, syn_A)$ by Theorem 2.

$RD_1 = \varnothing$ and $RD_2 = \varnothing$ imply $D_1 \triangle D_2 = \varnothing$.

$\neg\ div^\tau(SR^1, SA^1)$ and $\neg\ div^\tau(SR^2, SA^2)$ imply $\neg\ div^\tau((SR^1, SR^2, syn_R), (SA^1, SA^2, syn_A))$ by Property 8.

Then we have $(SR^1, SR^2, syn_R) \sqsubseteq_\eta (SA^1, SA^2, syn_A)$ by Property 1.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399