Priority Systems

Gregor Gössler[1] and Joseph Sifakis[2]

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

Abstract. We present a framework for the incremental construction of deadlock-free systems meeting given safety properties. The framework borrows concepts and basic results from the controller synthesis paradigm by considering a step in the construction process as a controller synthesis problem.
We show that priorities are expressive enough to represent restrictions induced by deadlock-free controllers preserving safety properties. We define a correspondence between such restrictions and priorities and provide compositionality results about the preservation of this correspondence by operations on safety properties and priorities. Finally, we provide an example illustrating an application of the results.

1 Introduction

A common idea for avoiding a posteriori verification and testing, is to use system design techniques that guarantee correctness by construction. Such techniques should allow to construct progressively from a given system S and a set of requirements $R_1, \ldots, R_n$, a sequence of systems $S_1, \ldots, S_n$, such that system S_i meets all the requirements R_j for $j \leq i$. That is, to allow incremental construction, requirements should be composable [2, 6] along the design process. In spite of their increasing importance, there is currently a tremendous lack of theory and methods, especially for requirements including progress properties which are essential for reactive systems. Most of the existing methodologies deal with construction of systems such that a set of state properties always hold. They are founded on the combined use of invariants and refinement relations. Composability is ensured by the fact that refinement relations preserve trace inclusion. We present a framework allowing to consider jointly state property invariance and deadlock-freedom.

Practice for building correct systems is often based on the idea of adding *enforcement* mechanisms to a given system S in order to obtain a system S' meeting a given requirement. These mechanisms can be implemented by instrumenting the code of S or by composing S with systems such as controllers or monitors that modify adequately the overall behavior.

An application of this principle is the enforcement of security policies which are safety properties described by automata [14]. A main requirement for the enforced system is that it safely terminates when it detects a deviation from

some nominal secure behavior. A more difficult problem is also to ensure system availability and preserve continuity of service [3, 10].

Another application of this principle is aspect oriented programming [8] used to build programs meeting (usually simple) requirements. Aspects can be considered as requirements from which code is generated and woven into a program intended to meet the requirements. In aspect oriented programming, aspect composition is identified as a central problem as it may cause unintentional interference and inconsistency [15].

Practice for building correct systems by using enforcement mechanisms raises some hard theoretical problems. For a sufficiently fine granularity of observation, it is relatively easy to enforce safety requirements (as non violations of given state properties) by stopping system progress. It is much harder to devise mechanisms that also guarantee system availability and avoid service interruption. Furthermore, composability of requirements e.g. security policies, aspects, is identified as a main obstacle to rigorous incremental system construction.

We propose a design framework for both safety and deadlock-freedom requirements. The framework consists of a model, *priority systems* and results concerning its basic properties including composability. A priority system is a transition system with a (dynamic) priority relation on its actions. A priority relation $\prec$ is a set of predicates of the form $a_i \prec C_{ij}.a_j$ meaning that action a_i has lower priority than action a_j at all states satisfying C_{ij}. At a given state of the transition system, only enabled actions with maximal priority can be executed. That is, in a priority system, a priority relation restricts the behavior of its transition system exactly as a scheduler restricts the behavior of a set of tasks. The remarkably nice property of priority systems is that they are deadlock-free if they are built from deadlock-free transition systems and from priority relations satisfying some easy-to-check consistency condition.

The proposed framework considers design as a *controller synthesis* [12] problem: from a given system S and requirement R, find a system S' meeting R. S' is the composition of S with a controller which monitors the state of S and restricts its behavior by adequately switching off a subset of *controllable* actions of S. The controller is usually specified as a solution of a fixpoint equation.

The simple case where R means that S' is deadlock-free and does not violate a state predicate U has been studied in various contexts e.g., in [11, 1]. The corresponding controller is specified as a *deadlock-free control invariant* which is a state predicate U', $U' \Rightarrow U$, such that

- it is preserved by the non controllable actions of S, that is if U' holds at some state then it remains true forever if only non controllable actions are executed;
- U' is *false* for all deadlock states of S.

Given U', the controlled (designed) system S' is obtained from S by conjuncting the guard of any controllable action a by the weakest precondition of U' under a.

In Section 2, we formalize the relationship between S and S', by introducing *restriction* operators. These are specified as tuples of state predicates in bijection with the set of actions of S. The application of a restriction operator to S is S', obtained from S by conjuncting the guards of its actions by the corresponding state predicates of the restriction. We study properties of deadlock-free control restrictions, that is restrictions corresponding to deadlock-free control invariants.

In Section 3, we show that under some consistency conditions, priorities can be used to represent deadlock-free restrictions. Thus, controlled systems S' can be represented as deadlock-free priority systems. Consistency checking boils down to computing a kind of transitive closure of the priority relation. We show that for static priorities consistency is equivalent to deadlock-freedom.

Composability in our framework means commutativity of application of priorities on a given system. As a rule, the result of the successive restriction of a system S by two priorities $\prec_1$ and $\prec_2$ depends on the order of application and we provide sufficient conditions for commutativity. This difficulty can be overcome by using a symmetric composition operator $\oplus$ for priorities which preserves safety and deadlock-freedom. The restriction of a system S by $\prec_1 \oplus \prec_2$ is a refinement of any other restriction of S obtained by application of $\prec_1$ and $\prec_2$.

An interesting question is whether priorities are expressive enough to represent restrictions induced by deadlock-free control invariants. We bring a positive answer by using a construction associating with a state predicate U a priority relation $\prec_U$. We show that if U is a deadlock-free control invariant then the controlled system S' is equivalent to the system S restricted by $\prec_U$. Furthermore, we provide results relating the controlled systems corresponding to U_1, U_2, $U_1 \wedge U_2$ to restrictions by $\prec_{U_1}$, $\prec_{U_2}$, $\prec_{U_1} \oplus \prec_{U_2}$.

Section 4 illustrates application of the results on an example.

Section 5 presents concluding remarks about the presented framework.

2 Deadlock-free Control Invariants

2.1 Definitions and basic properties

Definition 1 (Transition system). *A* transition system B *is a tuple* $(X, A, \{G^a\}_{a\in A}, \{F^a\}_{a\in A})$*, where*

- X *is a finite set of variables;*
- A *is a finite set of actions, union of two disjoint sets* A^u *and* A^c*, the sets of the* uncontrollable *and* controllable *interactions respectively;*
- G^a *is a* guard*, predicate on* X*;*
- $F^a : \mathbf{X} \to \mathbf{X}$ *is a* transition function*, where* $\mathbf{X}$ *is the set of valuations of* X*.*

Definition 2 (Semantics of a transition system). *A transition system* $(X, A, \{G^a\}_{a\in A}, \{F^a\}_{a\in A})$ *defines a* transition relation $\rightarrow: \mathbf{X} \times A \times \mathbf{X}$ *such that:* $\forall \mathbf{x}, \mathbf{x}' \in \mathbf{X}\ \forall a \in A\ .\ \mathbf{x} \xrightarrow{a} \mathbf{x}' \iff G^a(\mathbf{x}) \wedge \mathbf{x}' = F^a(\mathbf{x})$.

We introduce the following notations:

- Given two transition systems B_1, B_2 with disjoint action vocabularies such that $B_i = (X_i, A_i, \{G^a\}_{a\in A_i}, \{F^a\}_{a\in A_i})$, for $i = 1, 2$, their *union* is the transition system $B_1 \cup B_2 = (X_1 \cup X_2, A_1 \cup A_2, \{G^a\}_{a\in A_1\cup A_2}, \{F^a\}_{a\in A_1\cup A_2})$.
- Given a transition system B, we represent by B^u (respectively B^c) the transition system consisting of the uncontrollable (respectively controllable) actions of B. Clearly $B = B^u \cup B^c$.
- Given a transition system B, we represent by $G(B)$ the disjunction of its guards, that is $G(B) = \bigvee_{a\in A} G^a$ where A is the set of the actions of B.

Definition 3 (Predecessors). *Given $B = (X, A, \{G^a\}_{a\in A}, \{F^a\}_{a\in A})$ and a predicate U on X, the* predecessors *of U by action a is the predicate $pre_a(U) = G^a \wedge U([F^a(X)/X])$ where $U[F^a(X)/X]$ is obtained from U by uniform substitution of X by $F^a(X)$.*

Clearly, $pre_a(U)$ characterizes all the states from which execution of a leads to some state satisfying U.

Definition 4 (Invariants and control invariants). *Given a transition system B and a predicate U,*

- *U is an* invariant *of B if $U \implies \bigwedge_{a\in A} \neg pre_a(\neg U) = \bigwedge_{a\in A}(\neg G^a \vee U([F^a(X)/X])$. An invariant U, $U \neq$ false, is called* deadlock-free *if $U \Rightarrow G(B)$.*
- *U is a* control invariant *of B if $U \Rightarrow \bigwedge_{a\in A^u} \neg pre_a(\neg U)$. A control invariant U, $U \neq$ false, is called* deadlock-free *if $U \Rightarrow \bigvee_{a\in A} pre_a(U)$.*

We write $inv[B](U)$ to express the fact that U is an invariant of B. Notice that invariants are control invariants of systems that have only uncontrollable actions.

Proposition 1. *If U is a control invariant of $B = (X, A, \{G^a\}_{a\in A}, \{F^a\}_{a\in A})$ then U is an invariant of $B' = (X, A, \{(G^a)'\}_{a\in A}, \{F^a\}_{a\in A})$ where $(G^a)' = G^a \wedge U[F^a(X)/X]$ for $a \in A^c$ and $(G^a)' = G^a$ otherwise. Furthermore, if U is a deadlock-free control invariant of B then it is a deadlock-free invariant of B'.*

This result allows to build from a given system B and a safety requirement of the form "always U_0" a deadlock-free system B' meeting this requirement, provided there exists a deadlock-free control invariant U of B such that $U \Rightarrow U_0$. The following simple example illustrates this fact.

Example 1. In a Readers/Writers system, we use two counters, non negative integers, r and w to represent respectively, the number of readers and writers using a common resource. The counters are modified by actions of a transition system B specified as a set of guarded commands:

$$a_1 : true \to r := r + 1 \qquad a_2 : r > 0 \to r := r - 1$$
$$b_1 : true \to w := w + 1 \qquad b_2 : w > 0 \to w := w - 1$$

where a_1 and b_1 are respectively, the actions of granting the resource to a reader and a writer and a_2 and b_2 are respectively, the actions of releasing the resource by a reader and a writer.

We assume that the actions a_1 and b_1 are controllable and we want to enforce the requirement "always U" for $U = (w \leq 1) \wedge (w = 0 \vee r = 0)$. This prevents concurrent access among writers, as well as between readers and writers. It is easy to check that U is a deadlock-free control invariant. In fact, it is easy to check that U is preserved by the uncontrollable actions a_2 and b_2:

$(r > 0) \wedge U \Rightarrow U[r - 1/r]$ and $(w > 0) \wedge U \Rightarrow U[w - 1/w]$.

Furthermore, it is easy to check that $U \Rightarrow pre_{a_1} \vee pre_{a_2} \vee pre_{b_1} \vee pre_{b_2}$.

As $pre_{a_1}(U) \equiv w = 0$ and $pre_{b_1}(U) \equiv (w = 0) \wedge (r = 0)$, we have $inv[B'](U)$ where B' is the controlled transition system:

$$a_1 : w = 0 \rightarrow r := r + 1 \qquad a_2 : r > 0 \rightarrow r := r - 1$$
$$b_1 : (r = 0) \wedge (w = 0) \rightarrow w := w + 1 \qquad b_2 : w > 0 \rightarrow w := w - 1$$

The notion of *restriction* defined below allows a formalization of the relationship between the initial and the controlled system.

Definition 5 (Restriction). *Given a transition system $B = (X, A, \{G^a\}_{a\in A}, \{F^a\}_{a\in A})$, a* restriction *is a tuple of predicates $V = (U^a)_{a\in A}$. B/V denotes the transition system B* restricted by *V, $B/V = (X, A, \{G^a \wedge U^a\}_{a\in A}, \{F^a\}_{a\in A})$.*

$V = (U^a)_{a\in A}$ is a control restriction *for B if $\bigwedge_{a\in A^u}(\neg G^a \vee U^a) = true$.*

$V = (U^a)_{a\in A}$ is a deadlock-free restriction *for B if $\bigvee_{a\in A} G^a \wedge U^a = \bigvee_{a\in A} G^a$.*

We simply say that V is a control restriction *or a* deadlock-free restriction *if the corresponding equation holds for any transition system B with vocabulary of actions $A = A^c \cup A^u$ (independently of the interpretation of the guards).*

Definition 6 (U^A, $V(U)$). *Given a predicate U, we denote by U^A the restriction $U^A = (U)_{a\in A}$, and by $V(U)$ the restriction $V(U) = (U[F^a(X)/X])_{a\in A}$.*

If V_1, V_2 are two restrictions, $V_j = (U_j^{a_i})_{a_i\in A}$ for $j = 1, 2$, we write $V_1 \wedge V_2$ for the restriction $(U_1^{a_i} \wedge U_2^{a_i})_{a_i\in A}$.

Proposition 2 (Control invariants and restrictions). *Given a transition system B and a predicate U,*

a) *If U is a control invariant of B then $V(U)$ is a control restriction of B;*
b) *If U is a deadlock-free invariant of B then $V(U)$ is a deadlock-free restriction of B;*
c) *If U is a deadlock-free control invariant of B then $V(U)$ is a deadlock-free control restriction of B.*

We need the following definitions for the comparison of transition systems.

Definition 7 (Refinement and equivalence). *Given $B_i = (X_i, A, \{G_i^a\}_{a\in A}, \{F_i^a\}_{a\in A})$ for $i = 1, 2$, two transition systems and a predicate U we say that*

- *B_1* refines *B_2 under U, denoted by $B_1 \sqsubseteq_U B_2$, if $\forall a \in A\ .\ F_1^a = F_2^a$ and $U \wedge G_1^a \Rightarrow U \wedge G_2^a$;*

– B_1 is equivalent to B_2 *under* U, *denoted by* $B_1 \simeq_U B_2$, *if* $B_1 \sqsubseteq_U B_2$ *and* $B_2 \sqsubseteq_U B_1$.

We write $B_1 \sqsubseteq B_2$ *and* $B_1 \simeq B_2$ *for* $B_1 \sqsubseteq_{true} B_2$ *and* $B_1 \simeq_{true} B_2$, *respectively.*

Property 1. Given transition systems B, B_1, B_2 and restrictions V, V_1, V_2,

1a $B/V \sqsubseteq B$;
1b $(B_1 \cup B_2)/V \simeq B_1/V \cup B_2/V$;
1c $(B/V_1)/V_2 \simeq B/(V_1 \wedge V_2)$;
1d $B_1 \sqsubseteq B_2 \Rightarrow (inv[B_2](U) \Rightarrow inv[B_1](U))$ for any predicate U.

Notice that if the conjunction of control invariants is a control invariant, the conjunction of deadlock-free control invariants is not in general, a deadlock-free control invariant. We investigate conditions for composability.

3 Priority Systems

We define *priority systems* which are transition systems restricted with priorities. Priorities provide a general mechanism for generating deadlock-free restrictions.

3.1 Deadlock-free restrictions and priorities

Priorities

Definition 8 (Priority). *A* priority *on a transition system* B *with set of actions* A *is a set of predicates* $\prec = \{C_{ij}\}_{a_i,a_j \in A}$. *The restriction defined by* $\prec$, $V(B,\prec) = (U^a)_{a\in A}$ *is* $U^{a_i} = \bigwedge_{a_j \in A} \neg(C_{ij} \wedge G^{a_j})$.

The predicates C_{ij} specify priority between actions a_i and a_j. If C_{ij} is true at some state, then in the system restricted by $V(B,\prec)$ the action a_i cannot be executed if a_j is enabled. We write $a_i \prec C_{ij}.a_j$ to express the fact that a_i is dominated by a_j when C_{ij} holds. A priority is *irreflexive* if $C_{ij} \Rightarrow \neg C_{ji}$ for all $a_i, a_j \in A$.

Definition 9 (Transitive closure). *Given a priority* $\prec$ *we denote by* $\prec^+$ *the least priority such that* $\prec \subseteq \prec^+$, *obtained by the rule:*
$a_i \prec^+ C_{ij}.a_j$ *and* $a_j \prec^+ C_{jk}.a_k$ *implies* $a_i \prec^+ (C_{jk} \wedge C_{jk}).a_k$.

Proposition 3 (Activity preservation for priorities). *A priority* $\prec$ *defines a deadlock-free restriction if* $\prec^+$ *is irreflexive.*

Proof. Suppose that $\prec^+$ is irreflexive. Consider some transition system $B = (X, A, \{G^a\}_{a\in A}, \{F^a\}_{a\in A})$, and let $G = \bigvee_{a\in A} G^a$, and $V(B,\prec) = (U^a)_{a\in A}$. Let $\mathbf{x}$ be a state of B such that $G(\mathbf{x})$, let $A' = \{a \in A \mid G^a(\mathbf{x})\}$, and define a relation $\prec'$ on A' such that $\forall a_i, a_j \in A' \,.\, a_i \prec' a_j \iff C_{ij}(\mathbf{x})$. As $\prec^+$ is irreflexive, $\prec'$ is a partial order on A', and thus acyclic. If $A' \neq \emptyset$ then $\max A'$ exists and is non-empty. Thus, $\big(\bigvee_{a\in A} G^a \wedge U^a\big)(\mathbf{x}) = \big(\bigvee_{a\in A'} G^a\big)(\mathbf{x}) = \big(\bigvee_{a\in A} G^a\big)(\mathbf{x})$. ■

The above proposition motivates the definition of priority systems which are transition systems restricted by priorities.

Definition 10 (Priority system). *A* priority system *is a pair* $(B, \prec)$ *where* B *is a transition system and* $\prec = \{C_{ij}\}_{a_i,a_j \in A}$ *is a priority on* B *such that* $C_{ij} = false$ *for all* $(a_i, a_j) \in A^u \times A$.

The priority system $(B, \prec)$ *represents the transition system* $B/V(B, \prec)$.

The following propositions give properties of priority systems.

Proposition 4. *If* $(B, \prec)$ *is a priority system, then* $V(B, \prec)$ *is a control restriction for* B.

Proof. If $V(B, \prec) = (U^{a_i})_{a_i \in A}$ then for all uncontrollable actions a_i, $U^{a_i} = true$ because $C_{ij} = false$. ■

Corollary 1. *If* U *is a control invariant of* B *then* U *is a control invariant of* $(B, \prec)$.

Proposition 5. *If* U *is a deadlock-free control invariant of a transition system* B *then for any priority* $\prec$ *such that* $\prec^+$ *is irreflexive,* U *is a deadlock-free invariant of* $(B/V(U), \prec)$.

Proof. If U is a deadlock-free control invariant of B then $U \Rightarrow G(B/V(U))$ and $inv[B^u](U)$. As $\prec$ defines deadlock-free restrictions, $(B/V(U), \prec)^u = B^u$ and $G(B/V(U)) = G(B/V(U), \prec)$. ■

Static priorities

Definition 11 (Static priority). *A* static priority *is a priority* $\prec = \{C_{ij}\}_{a_i,a_j \in A}$ *such that the predicates* C_{ij} *are positive boolean expressions on guards. We call* static *restrictions the corresponding restrictions* $V(B, \prec) = (U^a)_{a \in A}$, *that is restrictions which are tuples of negative boolean expressions on guards.*

It is easy to check that any static restriction defines a static priority. Notice that in a priority system with static priorities, the choice of the action to be executed at some state depends only on the set of the actions enabled at this state. For example, a restriction with $U^{a_1} = \neg G^{a_2} \wedge (\neg G^{a_3} \vee \neg G^{a_4})$ means that in the restricted system the action a_1 can be executed only if a_2 is disabled and a_3 or a_4 is disabled. The corresponding the priority relation is: $a_1 \prec true.a_2, a_1 \prec G^{a_3}.a_4, a_1 \prec G^{a_4}.a_3$

Notation: For a static priorities the notation can be drastically simplified.

If $(U^{a_i})_{a_i \in A}$ is a static restriction then it is of the form, $U^{a_i} = \bigwedge_{k \in K_i} \neg M_i^k$ where M_i^k is a monomial on guards $M_i^k = \bigwedge_{k_w \in W} G^{a_{k_w}}$. Each monomial M_i^k, corresponds to the set of priorities $\{a_i \prec \bigwedge_{k_w \in W \setminus \{j\}} G^{a_{k_w}}.a_j\}_{j \in W}$. This set can be canonically represented by simply writing $a_i \prec \bigwedge_{k_w \in W} a_{k_w}$.

For example if $M_i^k = G^{a_1} \wedge G^{a_2} \wedge G^{a_3}$ instead of writing $a_i \prec (G^{a_1} \wedge G^{a_2}).a_3$, $a_i \prec (G^{a_1} \wedge G^{a_3}).a_2$, $a_i \prec (G^{a_2} \wedge G^{a_3}).a_1$, we write $a_i \prec a_1 a_2 a_3$. We propose the following definition for static priorities.

Definition 12 (Static priority – simplified definition). *A* monomial m *on a vocabulary of actions* A *is any term* $m = a_{j_1} \dots a_{j_n}$ *obtained by using an associative, commutative and idempotent product operation. Let* $\mathbf{1}$ *denote its neutral element, and* $\mathrm{M}(A)$ *the set of the monomials on* A.

A static priority $\prec$ *on* A *is a relation* $\prec \subseteq A \times \mathrm{M}(A)$.

Example 2. The static priority $\prec$ corresponding to the static restriction $U^{a_1} = true, U^{a_2} = true, U^{a_3} = \neg G^{a_1} \vee \neg G^{a_2}, U^{a_4} = \neg G^{a_1} \wedge \neg G^{a_2}, U^{a_5} = \neg G^{a_1} \wedge \neg G^{a_3} \vee \neg G^{a_2} \wedge \neg G^{a_4} \equiv \neg(G^{a_1} \wedge G^{a_2}) \wedge \neg(G^{a_1} \wedge G^{a_4}) \wedge \neg(G^{a_3} \wedge G^{a_2}) \wedge \neg(G^{a_3} \wedge G^{a_4})$ is: $a_3 \prec a_1a_2$, $a_4 \prec a_1$, $a_4 \prec a_2$, $a_5 \prec a_1a_2$, $a_5 \prec a_1a_4$, $a_5 \prec a_3a_2$, $a_5 \prec a_3a_4$.

Definition 13 (Closure). *Let* $\prec$ *be a static priority. The* closure *of* $\prec$ *is the least static priority* $\prec^{\mp}$ *containing* $\prec$ *such that*

- *if* $a_1 \prec^{\mp} a_2m_2$ *and* $a_2 \prec^{\mp} m_3$ *then* $a_1 \prec^{\mp} m_2m_3$*;*
- *if* $a \prec^{\mp} am$*, then* $a \prec^{\mp} m$ *for* $m \neq \mathbf{1}$.

Example 3. For $\prec = \{a \prec bc, b \prec ad\}$, $\prec^{\mp} = \{a \prec^{\mp} bc, b \prec^{\mp} ad, a \prec^{\mp} acd, a \prec^{\mp} cd, b \prec^{\mp} bcd, b \prec^{\mp} cd\}$.

Lemma 1. *If for any* $a_i \in A$, $a_i \prec \mathbf{m}_i$ *with* $\mathbf{m}_i$ *a monomial on* A*, then* $a_i \prec^{\mp} a_i$ *for some* $a_i \in A$.

Proof. Omitted.

Proposition 6 (Activity preservation for static priorities). *A static priority* $\prec$ *defines a deadlock-free restriction if and only if* $\prec^{\mp}$ *is irreflexive.*

Proof. "if": suppose that $\prec^{\mp}$ is irreflexive. By definition, only top elements in $\prec$ can be non-trivial monomials. Thus, $\prec$ is acyclic, and all ascending chains in $\prec$ are finite. Consider some deadlock-free transition system $B = (X, A, \{G^a\}_{a \in A}, \{F^a\}_{a \in A})$, and let $G = \bigvee_{a \in A} G^a$. Let $\mathbf{x}$ be a state of B such that $G(\mathbf{x})$, and let $A' = \{a \in A \mid G^a(\mathbf{x})\}$. As $\prec$ is acyclic, $\max A'$ exists and is non-empty. It remains to show that some element of $\max A'$ is not dominated by any monomial in $2^{A'}$. Suppose that for any $a_i \in A'$ there is some $\mathbf{m}_i \in 2^{A'}$, $a_i \prec \mathbf{m}_i$. In that case, $\prec^{\mp}$ has a circuit by lemma 1, which contradicts the hypothesis. Thus, at least one action in $\max A'$ is maximal in $\prec$.

"only if": suppose that $a \prec^{\mp} a$ for some $a \in A$. By construction of $\prec^{\mp}$, this means that $(A \cup \mathrm{M}(A), \prec)$ contains a tree $(A' \cup \mathrm{M}(A'), \prec')$ with root a such that all leaves are monomials consisting only of the action a. Take $B = (\emptyset, A, \{G^a\}_{a \in A}, \{\emptyset\}_{a \in A})$ with $G^{(a')} = true$ if $a' \in A'$, and $G^{(a')} = false$ otherwise. By definition of $/$, all guards in $B/V(B, \prec)$ are *false*, whereas B is clearly deadlock-free. ∎

Example 4. Consider the static priority $\prec$ on the actions a_1, a_2, a_3, a_4 such that, $a_2 \prec a_3a_4$, $a_3 \prec a_2a_4$, $a_4 \prec a_2a_3$. It is easy to see that $\prec^{\mp}$ is not irreflexive, thus $\prec$ does not define a deadlock-free restriction. By elimination of a_4, as in the proof of Lemma 1, we get: $a_2 \prec^{\mp} a_2a_3$, $a_3 \prec^{\mp} a_2a_3$ which gives by application of the second closure rule, $a_2 \prec^{\mp} a_2$, $a_3 \prec^{\mp} a_3$. Thus $\prec^{\mp}$ is not irreflexive.

Consider the slightly different static priority $\prec_1$ on the actions a_1, a_2, a_3, a_4 such that, $a_2 \prec_1 a_1a_3a_4$, $a_3 \prec_1 a_2a_4$, $a_4 \prec_1 a_2a_3$. It can be checked that $\prec_1^{\mp}$ is irreflexive and thus deadlock-free and contains the chain $a_4 \prec_1^{\mp} a_3 \prec_1^{\mp} a_2 \prec_1^{\mp} a_1$.

Clearly, $\prec_1^{+}$ is not irreflexive as $a_3 \prec_1 G^{a_2}.a_4, a_4 \prec_1 G^{a_2}.a_3$. This example shows that for static priorities the use of the specific closure gives finer results than by using Proposition 3.

3.2 Composition of priorities

Notice that given B and $\prec$, the predicate $V(B, \prec)$ depends on B. The property $((B, \prec^1), \prec^2) = ((B, \prec^2), \prec^1)$ does not hold in general. For instance, consider a system B and priorities $\prec$ and $\prec'$ such that $a_1 \prec a_2$ and $a_2 \prec' a_3$ where a_1, a_2, a_3 are actions of B. If from some state of B the three actions are enabled then in $((B, \prec), \prec')$ only a_3 is enabled while in $((B, \prec'), \prec)$ both a_1 and a_3 are enabled.

We define two composition operations on priorities and study composability results.

Definition 14 (Composition of priorities). *Given two priorities $\prec^1$ and $\prec^2$ their composition is the operation $\oplus$ such that $\prec^1 \oplus \prec^2 = (\prec^1 \cup \prec^2)^+$. Furthermore, if $\prec^1$ and $\prec^2$ are static priorities we define another composition operation, $\bar{\oplus}$ such that $\prec^1 \bar{\oplus} \prec^2 = (\prec^1 \cup \prec^2)^{\mp}$.*

Proposition 7. *The operations $\oplus$ and $\bar{\oplus}$ are associative and commutative.*

Lemma 2. *Let $\prec = \prec^{\mp}$ be an irreflexive closed static priority. Then, any non maximal action a is dominated by some monomial m on maximal actions.*

Proof. Omitted.

Proposition 8 (Composability for static priorities). *Given a transition system B and two static priorities $\prec^1$ and $\prec^2$, if $\prec^1 \cup \prec^2 = \prec^1 \bar{\oplus} \prec^2$ then $((B, \prec^1), \prec^2) \simeq (B, \prec^1 \bar{\oplus} \prec^2)$.*

Proof. Let G^a, $(G^a)'$, $(G^a)''$, and $(G^a)'''$ be the guards of action a in B, $B/\prec^1$, $(B/\prec^1)/\prec^2$, and $B/(\prec^1 \bar{\oplus} \prec^2)$, respectively. For some state $\mathbf{x}$, let $A_0 = \{a \in A \mid G^a(\mathbf{x})\}$, $A_1 = \{a \in A \mid (G^a)'(\mathbf{x})\}$, $A_2 = \{a \in A \mid (G^a)''(\mathbf{x})\}$, and $A_3 = \{a \in A \mid (G^a)'''(\mathbf{x})\}$, respectively. Notice that $A_2 \cup A_3 \subseteq A_1 \subseteq A_0$. We show that $A_2 = A_3$.

If $a \in A_2$ then there is no monomial on A_0 dominating a in $\prec^1$, and there is no monomial on A_1 dominating a in $\prec^2$. Thus, either there is no monomial on A_0 dominating a in $\prec^1 \cup \prec^2 = \prec^1 \bar{\oplus} \prec^2$, and $a \in A_3$, or there is a monomial m on A_0 such that $a \prec^2 m$. In the latter case, $m = m'm''$ with m' a non-empty monomial on $A_0 \setminus A_1$, and m'' a monomial on A_1 (i.e., product of actions that are maximal in $\prec^1$). Thus, for any factor a_i of m' there is a monomial m_i on A_0 (and by lemma 2, on A_1) such that $a_i(\prec^1)^+ m_i$. Since $(\prec^1)^+ \subseteq \prec^1 \cup \prec^2 = \prec^1 \bar{\oplus} \prec^2$, we have $a(\prec^1 \cup \prec^2)m_1 \cdots m_k m''$, and $a \notin A_2$, which is in contradiction to the assumption. Thus, $a \in A_3$.

Conversely, if $a \in A_3$, then a is not dominated by any monomial on A_0 in $\prec^1 \cup \prec^2$. Thus, a is maximal among A_0 and A_1 in both priorities, and $a \in A_2$. ■

Proposition 9 (Composability for priorities). *Given a transition system B and two priorities $\prec^1$, $\prec^2$, if $\prec^1 \cup \prec^2 = \prec^1 \oplus \prec^2$ then $((B, \prec^1), \prec^2) \simeq (B, \prec^1 \oplus \prec^2)$.*

Proof. Consider some state $\mathbf{x}$, and let $\prec^i_{\mathbf{x}}$ be the static priority defined by $\prec^i$ at state $\mathbf{x}$: $a_i \prec^i_{\mathbf{x}} a_j \iff C^i_{ij}(\mathbf{x})$, $i = 1, 2$. Notice that $\prec^1_{\mathbf{x}} \bar{\oplus} \prec^2_{\mathbf{x}}$ is irreflexive whenever $\prec^1 \oplus \prec^2$ is irreflexive. The proof follows that of proposition 8 for the static priorities $\prec^1_{\mathbf{x}}$ and $\prec^2_{\mathbf{x}}$ at state $\mathbf{x}$. ■

Propositions 8 and 9 provide composability conditions, that is conditions guaranteeing commutativity of two restrictions defined by priorities. The following proposition is easy to prove by using monotonicity properties $\sqsubseteq$ and the definitions of composition operations. It shows that the successive application of priority restrictions can be safely replaced by their composition.

Proposition 10. *For any transition system B and priorities $\prec^1$, $\prec^2$ we have*

- *if $\prec^1 \Rightarrow \prec^2$ then $(B, \prec^2) \sqsubseteq (B, \prec^1)$;*
- *$(B, \prec^1 \oplus \prec^2) \sqsubseteq (B, \prec^1 \cup \prec^2) \sqsubseteq ((B, \prec^1), \prec^2)$. Furthermore, for static priorities, $(B, \prec^1 \bar{\oplus} \prec^2) \simeq (B, \prec^1 \oplus \prec^2)$.*

3.3 Safety and deadlock-freedom

We present results relating deadlock-free control invariants to priorities. We show that priorities can be used to define any restriction corresponding to a deadlock-free control invariant.

Given a transition system B and a predicate U, the restriction $V(U)$ guarantees the invariance (safety) for U in $B/V(U)$, that is $inv[B/V(U)](U)$. Furthermore, if U is a control invariant then $V(U)$ is a control restriction, that is a restriction that does not affect the guards of uncontrollable actions. As a rule, $V(U)$ is not deadlock-free. We define for a predicate U, a priority $\prec_U$ and study relationships between its restrictions and $V(U)$.

Definition 15. *Given a state predicate U on a transition system, the associated priority $\prec_U$ is defined by $\prec_U = \{pre_a(\neg U) \wedge pre_{a'}(U)\}_{(a,a') \in A^c \times A}$.*

Property 2. The priority $\prec_U$ is transitively closed and irreflexive and thus it defines a deadlock-free restriction.

Proposition 11. *For any transition system B and predicate U, $B/V(U) \sqsubseteq_U (B, \prec_U)$. Furthermore, if U is a deadlock-free invariant of B, $B/V(U) \simeq_U (B, \prec_U)$.*

Proof. As we consider B with initial set of states satisfying U we assume that all the guards G^a of its actions are such that $G^a \Rightarrow U$. Let's verify that if $(G^{a_i})'$ is the restricted guard of action a_i in $(B, \prec_U)$, then $G^{a_i} \wedge pre_{a_i}(U) \Rightarrow (G^{a_i})'$.

We find $(G^{a_i})' = G^{a_i} \wedge \bigwedge_{a_j \in A} \neg(pre_{a_i}(\neg U) \wedge pre_{a_j}(U) \wedge G^{a_j}) = G^{a_i} \wedge \bigwedge_{a_j \in A}(\neg pre_{a_i}(\neg U) \vee \neg pre_{a_j}(U)) = G^{a_i} \wedge \neg pre_{a_i}(\neg U) \vee G^{a_i} \wedge \bigwedge_{a_j \in A} \neg pre_{a_j}(U)$.

Given that $G^{a_i} \wedge \neg pre_{a_i}(\neg U) = G^{a_i} \wedge pre_{a_i}(U)$, we have

$(G^{a_i})' = G^{a_i} \wedge pre_{a_i}(U) \vee G^{a_i} \wedge \bigwedge_{a_j \in A} \neg pre_{a_j}(U)$.

From this follows that $B/V(U) \sqsubseteq_U (B, \prec_U)$.

If U is a deadlock-free invariant then for any guard G^{a_i}, $G^{a_i} \Rightarrow U \Rightarrow \bigvee_{a_j \in A} pre_{a_j}(U)$. Thus, we have $G^a \wedge \bigwedge_{a_j \in A} \neg pre_{a_j}(U) = \mathit{false}$. Consequently, $(G^{a_i})' = G^{a_i} \wedge pre_{a_i}(U)$ which completes the proof. ■

A direct consequence of this proposition is that for any deadlock-free control invariant U, $B/V(U) \simeq_U (B, \prec_U)$. That is the effect of deadlock-free controllers can be modeled by restrictions induced by priorities.

From this proof it also follows that the guards of $B/V(U)$ and $(B, \prec_U)$ agree at deadlock-free states of $B/V(U)$ in U. They may differ at deadlock states of $B/V(U)$ where B is deadlock-free. In other words $(B, \prec_U)$ is a kind of "best deadlock-free abstraction" of $B/V(U)$ under U.

Example 5. We apply the previous proposition for B and U of Example 1. We show that $(B, \prec_U)$ behaves exactly as $B' = B/V(U)$ from any state satisfying the deadlock-free control invariant U.

We have $pre_{a_1}(\neg U) \wedge pre_{b_2}(U) \equiv w > 0$, $pre_{b_1}(\neg U) \wedge pre_{b_2}(U) \equiv w \geq 1$, $pre_{b_1}(\neg U) \wedge pre_{a_2}(U) \equiv r > 0$ and $pre_{a_1}(\neg U) \wedge pre_{b_1}(U) \equiv pre_{b_1}(\neg U) \wedge pre_{a_1}(U) \equiv \mathit{false}$. This gives the priority

$\prec_U = \{a_1 \prec_U (w > 0).b_2, b_1 \prec_U (w \geq 1).b_2), b_1 \prec_U (r > 0).a_2)\}$. It can be checked that $(B, \prec_U)$ is indeed equivalent to $(B/V(U))$. The computation of the restricted guards $(G^{a_1})'$ and $(G^{b_1})'$ gives

$(G^{a_1})' = G^{a_1} \wedge (\neg w > 0 \vee \neg G^{b_2}) \equiv w = 0$ and

$(G^{b_1})' = G^{b_1} \wedge (\neg w \geq 1 \vee \neg G^{b_2}) \wedge (\neg r > 0 \vee \neg G^{a_2}) \equiv (w = 0) \wedge (r = 0)$.

The following propositions study relationships between safety and deadlock-free restrictions.

Proposition 12. *If U_1, U_2 are two state predicates and $\prec_{U_1}$, $\prec_{U_2}$ the corresponding priorities, then $B/V(U_1 \wedge U_2) \sqsubseteq_{U_1 \wedge U_2} (B, \prec_{U_1} \oplus \prec_{U_2}) \sqsubseteq_{U_1 \wedge U_2} (B, \prec_{U_1 \wedge U_2})$.*

Furthermore, if $U_1 \wedge U_2$ is a deadlock-free invariant then $B/V(U_1 \wedge U_2) \simeq_{U_1 \wedge U_2} (B, \prec_{U_1} \oplus \prec_{U_2}) \simeq_{U_1 \wedge U_2} (B, \prec_{U_1 \wedge U_2})$.

Proof. Omitted.

This proposition says that $(B, \prec_{U_1} \oplus \prec_{U_2})$ is an upper approximation of $B/V(U_1 \wedge U_2)$. The following proposition shows an even stronger relationship between the two priority systems.

Proposition 13. *If U_1, U_2 are two deadlock-free invariants of B and $\prec_{U_1} \oplus \prec_{U_2}$ is irreflexive then $B/V(U_1 \wedge U_2) \simeq_{U_1 \wedge U_2} (B, \prec_{U_1} \oplus \prec_{U_2})$ is deadlock-free.*

Proof. We have from $B/V(U_1) \simeq_{U_1} (B, \prec_{U_1})$ and $B/V(U_2) \simeq_{U_2} (B, \prec_{U_2})$, $(B, \prec_{U_1} \oplus \prec_{U_2}) \sqsubseteq_{U_1} (B, \prec_{U_1} \cup \prec_{U_2}) \sqsubseteq_{U_1} B/V(U_1)$ and $(B, \prec_{U_1} \oplus \prec_{U_2}) \sqsubseteq_{U_2} (B, \prec_{U_1} \cup \prec_{U_2}) \sqsubseteq_{U_2} B/V(U_2)$. This gives, $(B, \prec_{U_1} \oplus \prec_{U_2}) \sqsubseteq_{U_1 \wedge U_2} B/V(U_1) \wedge V(U_2) \simeq B/V(U_1 \wedge U_2)$. From the previous proposition we get the result. ■

The following proposition provides for static priorities, a result similar to Proposition 11. It is very useful for establishing safety by using static priorities.

Proposition 14. *Given a state predicate U on a transition system $B = (X, A, \{G^a\}_{a \in A}, \{F^a\}_{a \in A})$, let $\prec_U$ be a static priority such that $\forall a \in A \,.\, pre_a(\neg U) \implies \bigvee_{m \,.\, a \prec_U m} \bigwedge_{a_i \in m} G^{a_i}$. Then, $inv[(B, \prec_U)](U)$.*

Proof. By Definition 10 of the semantics of $(B, \prec_U)$. ■

As shown in the following example, this proposition provides a means to ensure invariance of an arbitrary predicate U by static priorities. The choice of $\prec_U$ is a trade-off between completeness and efficiency. Extreme choices are given by

- $a \prec_U a' \iff pre_a(\neg U) \wedge pre_{a'}(U) \neq false$, which is a priority with singleton monomials only; the closure of this priority may easily be not irreflexive.
- $a \prec_U m \iff \exists \mathbf{x} \,.\, \big(pre_a(\neg U)\big)(\mathbf{x}) \;\wedge\; m = \{a' \mid G^{a'}(\mathbf{x})\}$ which is the most fine-grained static priority ensuring invariance of U.

4 Example

We consider a robotic system controlled by the following processes:

- 3 trajectory control processes TC_1, TC_2, TC_man. TC_2 is more precise and needs more resources than TC_1; TC_man is the process for manual operation.
- 2 motion planners, MP_1, MP_2; MP_2 is more precise and needs more resources than MP_1.

We consider for each process P predicates *P.halted* and *P.running* such that *P.halted* $\equiv \neg$*P.running*. Each process P can leave states of *P.halted* (resp. *P.running*) by action *P.start* (resp. *P.stop*), as in figure 1. The robotic system must satisfy forever the following constraints:

1. In order to ensure permanent control of the position and movements of the robot, at least one trajectory control process and at least one motion planner must be running at any time: $(TC_1.\mathsf{running} \vee TC_2.\mathsf{running} \vee TC\_man.\mathsf{running}) \wedge (MP_1.\mathsf{running} \vee MP_2.\mathsf{running})$.

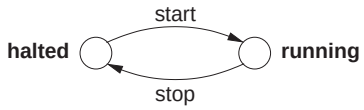

Fig. 1: Transition system of a process.

2. In order to meet the process deadlines, the CPU load must be kept below a threshold, which excludes that both high-precision processes can be simultaneously active: $TC_2.\mathsf{halted} \vee MP_2.\mathsf{halted}$.

The problem is to find a deadlock-free controller which restricts the behavior of the system so that the above requirement is met. A similar problem has been solved in [13] by using controller synthesis [12]. We propose a solution by finding an appropriate static priority.

We put the global constraint to be satisfied as a predicate U in conjunctive form: $U = (TC_1.\mathsf{running} \vee TC_2.\mathsf{running} \vee TC\_man.\mathsf{running}) \wedge (MP_1.\mathsf{running} \vee MP_2.\mathsf{running}) \wedge (TC_2.\mathsf{halted} \vee MP_2.\mathsf{halted})$.

Invariance of U requires the invariance of each one of the three conjuncts, disjunctions of predicates. We define the static priority $\prec_U$ in the following manner.

For each conjunct D consider the critical configurations where only one literal of the disjunction is *true*. The priority $\prec_U$ prevents critical actions, that is actions that can turn this term to *false*, by keeping them dominated by safe actions enabled in the considered configuration. More formally, for each disjunction D, each critical action a (for which $D \wedge pre_a(\neg D) \neq false$) is dominated by the monomial consisting of the safe actions enabled in D.

For example, take $D = TC_1.\mathsf{running} \vee TC_2.\mathsf{running} \vee TC\_man.\mathsf{running}$. Consider the critical configuration where $TC_1.\mathsf{running} = true$, $TC_2.\mathsf{running} = false$, and $TC\_man.\mathsf{running} = false$. Clearly, $TC_1.stop$ is a critical action for this configuration. Its occurrence can be prevented by the static priority $TC_1.\mathsf{stop} \prec TC_2.\mathsf{start} \cdot TC\_man.\mathsf{start}$. The monomial $TC_2.\mathsf{start} \cdot TC\_man.\mathsf{start}$ is the product of the safe actions enabled at this configuration. In this way, we compute the static priority $\prec_U$ which guarantees invariance of U:

$$
\begin{aligned}
TC_1.\mathsf{stop} &\prec_U TC_2.\mathsf{start} \cdot TC\_man.\mathsf{start} \\
TC_2.\mathsf{stop} &\prec_U TC_1.\mathsf{start} \cdot TC\_man.\mathsf{start} \\
TC\_man.\mathsf{stop} &\prec_U TC_1.\mathsf{start} \cdot TC_2.\mathsf{start} \\
MP_1.\mathsf{stop} &\prec_U MP_2.\mathsf{start} \\
MP_2.\mathsf{stop} &\prec_U MP_1.\mathsf{start} \\
TC_2.\mathsf{start} &\prec_U MP_2.\mathsf{stop} \\
MP_2.\mathsf{start} &\prec_U TC_2.\mathsf{stop}
\end{aligned}
$$

It is easy to see that $\prec_U^{\dagger}$ is irreflexive. By Proposition 6, $\prec_U$ is a deadlock-free restriction. By Proposition 14, U is an invariant of $(TC_1 \cup TC_2 \cup TC\_man \cup MP_1 \cup MP_2, \prec_U)$.

This approach can be applied to find deadlock-free control restrictions of arbitrary systems of processes $\{B_1, \ldots, B_n\}$ abstractly modeled as the deadlock-free transition system of figure 1, preserving a predicate U, boolean expression on atomic predicates $B_i.\mathsf{running}$ and $B_i.\mathsf{halted}$. For example, U can express requirements on the global system state such as:

- a safety-critical process must not run unless a failure-handling process is running;
- mutual exclusion between concurrently running processes, e.g., between a safety-critical and an untrusted process.

We suppose U to be written as a conjunction of disjunctions

$$U = \bigwedge_{i \in I} \Big(\bigvee_{j \in J_i} B_j.\mathsf{running} \vee \bigvee_{j \in J'_i} B_j.\mathsf{halted} \Big)$$

where I, J_i and J'_i are index sets such that any conjunct has at least two atoms that are predicates on two different processes (this is always possible for any predicate U if we have at least two processes).

Invariance of U is equivalent to invariance of all of its conjuncts D_i. Consider the conjunct $\bigvee_{l \in J_i} B_l.\mathsf{running} \vee \bigvee_{l \in J'_i} B_l.\mathsf{halted}$. As in the previous example, consider a critical configuration, that is, a configuration where only one literal is true. We distinguish two cases:

- if that literal is $B_j.\mathsf{running}$ (thus $j \in J_i$), then $B_j.\mathsf{stop}$ violates U from this configuration characterized by $\bigwedge_{l \in J_i \setminus \{j\}} B_l.\mathsf{halted} \wedge \bigwedge_{l \in J'_i \cup \{j\}} B_l.\mathsf{running}$. This action can be prevented by the static priority

$$B_j.\mathsf{stop} \prec_U \prod_{l \in J_i \setminus \{j\}} B_l.\mathsf{start} \cdot \prod_{l \in J'_i} B_l.\mathsf{stop}$$

  In this relation, $B_j.\mathsf{stop}$ is dominated by the monomial consisting of the actions of the other processes involved in this configuration.
- if the literal is $B_j.\mathsf{halted}$ (thus $j \in J'_i$), then $B_j.\mathsf{start}$ violates U, and we apply a similar reasoning and get $B_j.\mathsf{start} \prec_U \prod_{l \in J_i} B_l.\mathsf{start} \cdot \prod_{l \in J'_i \setminus \{j\}} B_l.\mathsf{stop}$.

Let $\prec_U$ be the union of the so defined priorities for all $i \in I$.

By definition of $\prec_U$, for any disjunct D_i of U, any critical action a is dominated by at least one monomial $m(a, D_i) = \prod B_l.\mathsf{start} \cdot \prod B_l.\mathsf{stop}$ consisting of safe actions enabled in D_i. Thus, $pre_a(\neg D_i) \implies \bigwedge_{a_i \in m(a,D_i)} G^{a_i}$, and $pre_a(\neg U) = pre_a(\neg \bigwedge_{i \in I} D_i) = pre_a\big(\bigvee_{i \in I} \neg D_i\big) = \bigvee_{i \in I} pre_a(\neg D_i) \implies \bigvee_{i \in I} \bigwedge_{a_i \in m(a,D_i)} G^{a_i}$. By proposition 14, U is an invariant of $\big(\bigcup_i B_i, \prec_U\big)$. Notice that $\prec_U$ is minimally restrictive, that is, only transitions violating the invariance of U are inhibited.

Deadlock-freedom of $\big(\bigcup_i B_i, \prec_U\big)$ is established by Proposition 14 if $\prec_U^{\dagger}$ is irreflexive, which depends on the actual predicate U.

5 Discussion

We present a framework for the incremental construction of deadlock-free systems meeting given safety properties. The framework borrows concepts and basic results from the controller synthesis paradigm by considering a step in the construction process as a controller synthesis problem. Nevertheless, it does not directly address controller synthesis and other related computationally hard problems. Instead, it is based on the abstraction that the effect of the controller corresponding to a deadlock-free control invariant can be modeled by deadlock-free control restrictions.

Priorities play a central role in our framework. They can represent any deadlock-free control restriction. They can be naturally used to model mutual exclusion constraints and scheduling policies [4, 2]. They are equipped with very simple and natural composition operations and criteria for composability. We provide an equational characterization of priorities and a sufficient condition for representing deadlock-free restrictions. Static priorities are solutions expressed as boolean expressions on guards for which a necessary and sufficient condition for deadlock-freedom is provided.

The use of priority systems instead of simple transition systems is a key idea in our approach. Of course, any priority system is, by its semantics, equivalent to a transition system. Nevertheless, using such layered models offers numerous advantages of composability and compositionality:

- The separation between transition system (behavior) and priorities allows reducing global deadlock-freedom to deadlock-freedom of the transition system and a condition on the composed priorities.
- The use of priorities to model mutual exclusion and scheduling policies instead of using transition systems leads to more readable and compositional descriptions [2].
- In [6, 5] priority systems are used to define a general framework for component-based modeling. This framework uses a single associative parallel composition operator for layered components, encompassing heterogeneous interaction. Priorities are used to express *interaction constraints*. For systems of interacting components, we have proposed sufficient conditions for global and individual deadlock-freedom, based on the separation between behavior and priorities.

Our work on priorities found application in generating schedulers for real-time Java applications [9]. This paper uses a scheduler synthesis algorithm that generates directly (dynamic) priorities. Another interesting application is the use of priorities in the IF toolset to implement efficiently run-to-completion semantics of the RT-UML profile [7].

Priority systems combine behavior with priorities, a very simple enforcement mechanism for safety and deadlock-freedom. This mechanism is powerful enough to model the effect of controllers ensuring such properties. They offer both abstraction and analysis for incremental system construction. Our theo-

retical framework can be a basis for the various approaches and practices using enforcement mechanisms in a more or less ad-hoc manner.

References

1. K. Altisen, G. Gössler, A. Pnueli, J. Sifakis, S. Tripakis, and S. Yovine. A framework for scheduler synthesis. In *Proc. RTSS'99*, pages 154–163. IEEE Computer Society Press, 1999.
2. K. Altisen, G. Gössler, and J. Sifakis. Scheduler modeling based on the controller synthesis paradigm. *Journal of Real-Time Systems, special issue on "control-theoretical approaches to real-time computing"*, 23(1/2):55–84, 2002.
3. L. Bauer, J. Ligatti, and D. Walker. A calculus for composing security policies. Technical Report TR-655-02, Princeton University, 2002.
4. S. Bornot, G. Gössler, and J. Sifakis. On the construction of live timed systems. In S. Graf and M. Schwartzbach, editors, *Proc. TACAS'00*, volume 1785 of *LNCS*, pages 109–126. Springer-Verlag, 2000.
5. G. Gössler and J. Sifakis. Component-based construction of deadlock-free systems (extended abstract). In *proc. FSTTCS'03*, volume 2914 of *LNCS*. Springer-Verlag, 2003.
6. G. Gössler and J. Sifakis. Composition for component-based modeling. In *proc. FMCO'02*, volume 2852 of *LNCS*. Springer-Verlag, 2003.
7. S. Graf, I. Ober, and I. Ober. Model checking of uml models via a mapping to communicating extended timed automata. In S. Graf and L. Mounier, editors, *Proc. SPIN'04*, volume 2989 of *LNCS*. Springer-Verlag, 2004.
8. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In *Proc. ECOOP '97*, volume 1241 of *LNCS*, page 220ff. Springer-Verlag, 1997.
9. C. Kloukinas, C. Nakhli, and S. Yovine. A methodology and tool support for generating scheduled native code for real-time java applications. In R. Alur and I. Lee, editors, *Proc. EMSOFT'03*, volume 2855 of *LNCS*, pages 274–289, 2003.
10. J. Ligatti, L. Bauer, and D. Walker. Edit automata: Enforcement mechanisms for run-time security policies. Technical Report TR-681-03, Princeton University, 2003.
11. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In E.W. Mayr and C. Puech, editors, *STACS'95*, volume 900 of *LNCS*, pages 229–242. Springer-Verlag, 1995.
12. P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1), 1987.
13. E. Rutten and H. Marchand. Task-level programming for control systems using discrete control synthesis. Technical Report 4389, INRIA, 2002.
14. F. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, 2000.
15. P. Tarr, M. D'Hondt, L. Bergmans, and C. V. Lopes. Workshop on aspects and dimensions of concern: Requirements on, challenge problems for, advanced separation of concerns. In *ECOOP 2000 Workshop Proceedings, Springer Verlag*, 2000.