Ensuring Reachability by Design*

Benoît Caillaud[1] and Jean-Baptiste Raclet[2]

[1] INRIA, Campus de Beaulieu, F-35042 Rennes cedex, France
Benoit.Caillaud@inria.fr
[2] IRIT/CNRS, 118 Route de Narbonne, F-31062 Toulouse cedex 9, France
Jean-Baptiste.Raclet@irit.fr

Abstract. This paper studies the independent implementability of reachability properties, which are in general not compositional. We consider modal specifications, which are widely acknowledged as suitable for abstracting implementation details of components while exposing to the environment relevant information about cross-component interactions. In order to obtain the required expressivity, we extend them with marked states to model states to be reached. We then develop an algebra with both logical and structural composition operators ensuring reachability properties by construction.

1 Introduction

In order to face the intrinsic complexity of automotive, aeronautic and consumer electronics embedded systems, but also of web-based service oriented architectures, modular design aims at organizing systems as a set of distinct components that can be developed independently and then assembled together. This is best achieved using interfaces which abstract superfluous implementation details of a component and expose cross-component protocol informations that are essential to a correct use of a component. Component reuse in different contexts is thus made possible, not only reducing design time, but also enabling the amortization of design costs over several different projects.

Component interoperability or compatibility is then a major issue: when can we safely compose two (or more) components? Compatibility is often considered at a signature level. In this simple case, interfaces consist in function or method types and compatibility consists in a type-checking, performed either at compile-time or at run-time. This paper deals with a richer notion of interfaces capable of capturing behavioral properties.

The first work on behavioral compatibility of interfaces has been proposed in [1]. This paper considers an automata-based formalism for interfaces in which transitions are labeled with output (produced by the component) or input (produced by the environment) actions. Then, a run-time error occurs whenever a component produces an output that is not accepted as input by one of its peers. The fact that a runtime error may occur does not necessarily lead to deem the

* A long version is available as a research report [5].

interfaces incompatible. Indeed, the authors promote an optimistic approach of composition in which two interfaces are compatible if there exists a restriction of the permitted actions of the environment in order to prevent the reachability of a runtime error. They show that this form of compatibility is preserved in the design flow provided alternating refinement [2] is used. More precisely, starting from initial interfaces whose product satisfies a particular *safety* property (i.e., a runtime error cannot be reached), they can be refined independently and then composed, their product will also satisfy the same safety property. This principle, called *independent implementability*, is of key importance [11] and enables the concurrent design of systems that are then assembled in a bottom-up manner.

This paper now studies the case of *reachability* properties and proposes results regarding their satisfaction by design. Basically, a reachability property states that some particular situation can be reached. Examples abound in practice. For instance, consider Service Oriented Architectures (SOA) formed of several interacting services; they should always have the possibility to reach a termination state, by delivering a response to all service activation. However, termination is in general not preserved by service composition. Although reachability properties are easy to verify in this context [4], model-checking may not be an appropriate solution. First, because it requires to construct the reachability graph of a system which may lead to a state explosion problem. Moreover, in case model-checking reveals a violation of the reachability property, designers have to iterate the design cycle by re-coding and re-validating their components, therefore extending time to market. The alternative approach advocated in this paper consists in controlling the design flow of components, that is, the evolution of interfaces through compositions and refinements, in order to ensure a reachability property by construction. Now, what specification formalism capturing some behavioral aspects of components is convenient for interface-based design? Modal specifications [16,14,3] are widely acknowledged as a suitable proposal [12,20,21]. Basically, they consist in labeling interface transitions with modalities, either *must* if the transition has to be enabled in any refinement, or *may* if the transition is allowed. In [12,20,21], modal specifications are shown to have many benefits comparing the specification formalism introduced in [1]; they are not only equipped with an optimistic composition operator and a refinement relation but also with a conjunction and a quotient operator. As reachability properties cannot be expressed, in general, with modal specifications, we first consider in this paper modal specifications enriched with marked states, in the same fashion as it is done in [6]. We show that, in this framework, we can develop a theory ensuring reachability properties by design.

2 Modeling with marked modal specifications

2.1 Background on automata

Let Σ be a finite alphabet of actions, a *deterministic* automaton over Σ is a tuple $\mathcal{M} = (R, r^0, \Sigma, \lambda, G)$ where R is a finite set of states, $r^0 \in R$ is the unique initial state, λ is a partial map from $R \times \Sigma$ to R called the *labeled transition*

map and $G \subseteq R$ is a non-empty set of *marked* states. The set of *firable* actions from $r \in R$ is $ready(r) = \{a \in \Sigma \mid \lambda(r, a) \text{ is defined}\}$.

Transition map λ is extended to its transitive and reflexive closure: let ϵ denote the empty word, for all $r \in R$, $\lambda(r, \epsilon) = r$ and for all $u \in \Sigma^*$, $a \in \Sigma$, $r_1, r_2, r_3 \in R$, $\lambda(r_1, u) = r_2$ and $\lambda(r_2, a) = r_3$ imply $\lambda(r_1, u.a) = r_3$. Define $\mathcal{L}_\mathcal{M} = \{u \in \Sigma^* \mid \exists r' \in R,\ \lambda(r^0, u) = r'\}$ to be the *language* of $\mathcal{M}$. If $\lambda(r, u) = r'$ for some u then r' is said to be *reachable* from r.

Given $P \subseteq R$, define $pre^*(P)$ and $post^*(P)$ to be the set of states that are respectively coreachable and reachable from any state $r \in P$: it is the least set such that for $r \in P$, $r \in pre^*(P)$ and $r \in post^*(P)$ and for every $\lambda(r', a) = r''$, if $r'' \in pre^*(P)$ then $r' \in pre^*(P)$ and if $r' \in post^*(P)$ then $r'' \in post^*(P)$. With a slight abuse, we may write $pre^*(r)$ and $post^*(r)$ for $pre^*(\{r\})$ and $post^*(\{r\})$.

If modeling a service, it is desirable to set that a service session eventually terminates; this is often refered in SOC as weak termination. To capture this kind of requirement, we define *terminating* automata: an automaton $\mathcal{M}$ is said to be *terminating* whenever $R = pre^*(G)$ meaning that it is always possible to reach a marked state from any state of the automaton. In other words, $\mathcal{M}$ is terminating if and only if for any $u \in \mathcal{L}_\mathcal{M}$, there exists a v such that $uv \in \mathcal{L}_\mathcal{M}$ and $\lambda(r^0, uv) \in G$. In the temporal logic CTL, this property can be written $AG(EF\ G)$.

Given two automata $\mathcal{M}_1 = (R_1, r_1^0, \Sigma_1, \lambda_1, G_1)$ and $\mathcal{M}_2 = (R_2, r_2^0, \Sigma_2, \lambda_2, G_2)$, their product is the automaton $\mathcal{M}_1 \times \mathcal{M}_2 = (R_1 \times R_2, (r_1^0, r_2^0), \Sigma_1 \cup \Sigma_2, \lambda, G_1 \times G_2)$ where $\lambda((r_1, r_2), a)$ is defined as $(\lambda_1(r_1, a), r_2)$ for $a \in \Sigma_1 \setminus \Sigma_2$, $(r_1, \lambda_2(r_2, a))$ for $a \in \Sigma_2 \setminus \Sigma_1$ and $(\lambda_1(r_1, a), \lambda_2(r_2, a))$ for $a \in \Sigma_1 \cap \Sigma_2$.

2.2 Marked modal specifications

Following [6], we enrich modal specifications [16,14,3] with marked states in order to model *states to be reached*. For instance, if a designer specifies a service, this enables to represent session terminations. The obtained formalism allows to specify a (possibly infinite) set of automata called *implementations*.

Definition 1 (Marked Modal Specification). *A marked modal specification over Σ is a tuple $\mathcal{C} = (Q, q^0, \Sigma, \delta, must, may, F)$, where Q is a finite set of states, $q^0 \in Q$ is the unique initial state, $\delta : Q \times \Sigma \to Q$ is a partial labeled transition map; $must, may : Q \to 2^\Sigma$ map to each state q the set of required and allowed actions from q, $F \subseteq Q$ is a non-empty set of marked states.*

It is assumed that a transition is associated to any allowed action, that is for every state $q \in Q$ and every action $a \in \Sigma$, $a \in may(q)$ if and only if $\delta(q, a)$ is defined. The mapping $may : Q \to 2^\Sigma$ can thus be reconstructed from the transition relation δ. However, this distinction simplifies the definition of satisfaction and refinement relations and compositions operators.

In this paper, marked modal specifications are taken *deterministic*, that is: for any $a \in \Sigma$ and any state q there is at most one state q' such that $\delta(q, a) = q'$. The reason for this will be given later in Sec. 3.

The *underlying automata* associated to $\mathcal{C}$ is $\mathrm{Un}(\mathcal{C}) = (Q, q^0, \Sigma, \delta, F)$. The language $\mathcal{L}_\mathcal{C}$ is then $\mathcal{L}_{\mathrm{Un}(\mathcal{C})}$. As previously for automata, we extend δ to words by taking its transitive and reflexive closure. Moreover, we define $pre^*_M(P)$ and $pre^*_m(P)$ with $P \subseteq Q$ as the set of states that are coreachable from any state $q \in Q$ by following transitions labeled by required and allowed actions, respectively: $pre^*_m(P)$ corresponds to $pre^*(P)$ in $\mathrm{Un}(\mathcal{C})$; $pre^*_M(P)$ is the least set such that for $r \in P$, $r \in pre^*_M(P)$ and for every $\lambda(r', a) = r''$ with $a \in must(r')$ and $r'' \in pre^*_M(P)$ then $r' \in pre^*_M(P)$. Last, $post^*_m(P)$ is $post^*(P)$ in $\mathrm{Un}(\mathcal{C})$.

Any terminating automaton can be seen as a marked modal specification with no design choice left open, that is, for any state r, the optional action set $may(r) \setminus must(r)$ is empty. More formally, the *embedding* of a terminating automaton $\mathcal{M} = (R, r^0, \Sigma, \lambda, G)$ into the class of the marked modal specifications is $\mathrm{Em}(\mathcal{M}) = (R, r^0, \Sigma, \lambda, must, may, G)$ with, for all $r \in R$, $may(r) = must(r) = ready(r)$. Now, the semantics of marked modal specifications is given in terms of terminating automata:

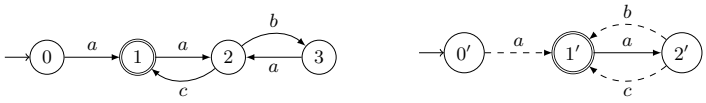
Definition 2 (Satisfaction). *A terminating automaton $\mathcal{M} = (R, r^0, \Sigma, \lambda, G)$ satisfies the marked modal specification $\mathcal{C} = (Q, q^0, \Sigma, \delta, must, may, F)$, denoted $\mathcal{M} \models \mathcal{C}$, if and only if there exists a simulation relation $\pi \subseteq R \times Q$ such that $(r^0, q^0) \in \pi$ and for all pairs $(r, q) \in \pi$:*
- *$must(q) \subseteq ready(r) \subseteq may(q)$;*
- *$r \in G$ implies $q \in F$;*
- *for every $a \in \Sigma$ and every $r' \in R$, $\lambda(r, a) = r'$ implies $\big(r', \delta(q, a)\big) \in \pi$.*

The set of models (or implementations) of $\mathcal{C}$ is denoted $[\![\mathcal{C}]\!]$. A marked modal specification is said *satisfiable* if and only if $[\![\mathcal{C}]\!] \neq \emptyset$. Two marked modal specifications $\mathcal{C}$ and $\mathcal{C}'$ are said *equivalent*, written $\mathcal{C} \equiv \mathcal{C}'$, if and only if they admit the same implementations: $[\![\mathcal{C}]\!] = [\![\mathcal{C}']\!]$. Any unsatisfiable specification is mapped on a special specification denoted $\mathcal{C}_\perp$, with $[\![\mathcal{C}_\perp]\!] = \emptyset$.

Example 1. Consider the terminating automaton $\mathcal{M}$ in Fig. 1(a) and the marked modal specification $\mathcal{C}$ in Fig. 1(b) where transitions from q labeled by a are dashed when $a \in may(q) \setminus must(q)$ and plain when $a \in must(q)$; marked states are double-circled. $\mathcal{M}$ satisfies $\mathcal{C}$ because of the simulation relation $\pi = \{(0, 0'), (1, 1'), (2, 2'), (3, 1')\}$.

Observe that, in state $2'$, although none of the two outgoing transition is *must*, at least one of the two has to be present in any model in order to preserve the reachability of a marked state. Such restricted disjunction cannot be expressed with traditional unmarked modal specifications. Observe also that, according to the second item of the Def. 2, the reachability of a marked state may be delayed: $1'$ is marked, $(3, 1') \in \pi$ but 3 is not marked; however, a marked state can be eventually reached from 3 thanks to the state 1.

According to Def. 2, only reachable states of $\mathcal{C}$ are semantically meaningful. We thus suppose from now on, and without loss of generality, that $\mathcal{C}$ is reachable, that is: $\forall q \in Q,\ q^0 \in pre^*(q)$.



(a) A terminating automaton $\mathcal{M}$ (b) A marked modal spec. $\mathcal{C}$

Fig. 1. $\mathcal{M}$ is a model of $\mathcal{C}$

A marked state $q \in F$ is said *delayable* if q can be reached again, that is, there exists a word $u \neq \epsilon$ such that $\delta(q, u) = q$; it is said *undelayable* otherwise. Denote by D the set of delayable states of a marked modal specification.

A marked state $q \in F$ is a bottleneck of $\mathcal{C}$ if it is the only marked state reachable from some state $q' \in Q$ that is, $post^*_m(q') \cap F = \{q\}$. Intuitively, this notion allows to identify the states that will be marked in any model of the specification.

Lemma 1. *Given a terminating automaton $\mathcal{M}$ and a marked modal specification $\mathcal{C}$ s.t. $\mathcal{M} \models \mathcal{C}$ then: $\mathcal{L}_\mathcal{M} \subseteq \mathcal{L}_\mathcal{C}$, and, for all $u \in \mathcal{L}_\mathcal{M}$, $\big(\lambda(r^0, u), \delta(q^0, u)\big) \in \pi$.*

The introduced semantics induces some simplifications in the structure of the marked modal specifications that we discuss now. At the end of this section, this will lead to the definition of an associated normal form.

Must-saturation. Observe that any terminating automaton model of the marked modal specification in Fig. 1 includes the starting transition labeled by a stemming from the initial state. It is thus a required transition that can be assigned a must modality in the specification. We therefore introduce the *must-saturation* of marked modal specifications.

Definition 3 (Must-saturation). *A marked modal specification is* must-saturated *if for all $q \notin F$ such that there is a unique $a \in may(q)$, we have $a \in must(q)$. Such a must-mapping is then said to be* saturated.

Lemma 2. *Any must-mapping can be saturated without changing the set of marked implementations.*

Consistency and attractability. Given a marked modal specification $\mathcal{C} = (Q, q^0, \Sigma, \delta, must, may, F)$ and a state $q \in Q$, $\mathcal{C}$ is said *consistent* in q if and only if $must(q) \subseteq may(q)$. $\mathcal{C}$ is said *attracted* in q if and only if $q \in pre^*_m(F)$.

Lemma 3. *If $\mathcal{M} \models \mathcal{C}$ then $\mathcal{C}$ is consistent and attracted in every state $\delta(q^0, u)$ with $u \in \mathcal{L}_\mathcal{M}$.*

As a consequence, only consistent and attracting states of $\mathcal{C}$ are semantically meaningful. This now leads us to define a reduced form:

Definition 4 (Reduced marked modal specification). *$\mathcal{C}$ is* reduced *iff every state is reachable and it is consistent and attracted in every state $q \in Q$.*

Proposition 1 (Reducibility). *Every satisfiable marked modal specification is equivalent to a reduced marked modal specification.*

Proof of this proposition is by construction of a reduced marked specification $\rho\mathcal{C}$ and then proving that $\mathcal{C}$ and $\rho\mathcal{C}$ are equivalent. This construction makes use of a pruning operation. We denote by $Q_\Psi \subseteq Q$ the set of all states $q \in Q$ such that q is inconsistent or unattracting, that is: $must(q) \not\subseteq may(q)$ or $q \notin pre^*(F)$.

Definition 5 (Reduction operation). *Given a marked modal specification $\mathcal{C} = (Q, q^0, \Sigma, \delta, must, may, F)$: if $q^0 \in pre^*_M(Q_\Psi)$ then the reduction of $\mathcal{C}$ is $\mathcal{C}_\perp$; otherwise, it is the marked modal specification $(Q \setminus pre^*_M(Q_\Psi), q^0, \Sigma, \delta', must', may', F \setminus pre^*_M(Q_\Psi))$ where: $\delta'(r, a) = r'$ if and only if $\delta(r, a) = r'$ and $r, r' \notin pre^*_M(Q_\Psi)$; as indicated in right after Def. 1, may' can be recovered from δ' whereas $must'$ is the restriction of must to the domain $Q \setminus pre^*_M(Q_\Psi)$.*

Normal form. This now leads us to define the normal form of any marked modal specification:

Definition 6 (Normal form). *A marked modal specification is in* normal form *if it is both must-saturated and reduced.*

According to Lem. 2 and Prop. 1, any marked modal specification $\mathcal{C}$ can be put in normal form ηC without altering its set of models. As a result, from now on, we always suppose that marked modal specifications are in normal form.

At this point, the reader may wonder why must-saturation, consistency and attractability are not fully part of the definition of marked modal specification (as it is the case for the consistency requirement in the original papers on unmarked modal specifications [16,14]). The reason for this is because, in what follows, we propose composition operators on marked modal specifications and it is easier to define these constructions without trying to preserve these different requirements. Now if the combination of two marked modal specifications (which are now implicitly supposed to be in normal form) gives rise to a specification violating one of the above requirements then a step of normalization has to be applied on the result in order to have an iterative process.

3 Refinement of marked modal specifications

A refinement relation aims at relating interfaces at different stages of their design. Basically, it should correspond to refine the set of allowed implementations of an interface. Moreover, we shall see later that refinement should entail *substitutability*, meaning that the substitution of an interface $\mathcal{C}_2$ by a refined version $\mathcal{C}_1$ must not impact the possible and actual cooperation with other components, that have been previously declared as legal for $\mathcal{C}_2$.

Definition 7 (Refinement). *Given two marked modal specifications $\mathcal{C}_1 = (Q_1, q_1^0, \Sigma, \delta_1, must_1, may_1, F_1)$ and $\mathcal{C}_2 = (Q_2, q_2^0, \Sigma, \delta_2, must_2, may_2, F_2)$, $\mathcal{C}_1$ is a refinement of $\mathcal{C}_2$, noted $\mathcal{C}_1 \leq \mathcal{C}_2$, if and only if there exists a simulation relation $\Pi \subseteq Q_1 \times Q_2$ such that $(q_1^0, q_2^0) \in \Pi$ and, for all pairs $(q_1, q_2) \in \Pi$:*

– $may_1(q_1) \subseteq may_2(q_2)$ *and* $must_1(q_1) \supseteq must_2(q_2)$*;*
– $q_1 \in F_1$ *implies* $q_2 \in F_2$*;*
– *for every* $a \in may_1(q_1)$*, we have:* $\big(\delta_1(q_1, a), \delta_2(q_2, a)\big) \in \Pi$*.*

Intuitively, refining an interface corresponds to possibly changing a transition with a *may* modality into either a required or a proscribed transition while potentially delaying the reachability of a marked state. This relation is reflexive and transitive and is thus a preorder.

Theorem 1. *Given two marked modal specifications* $\mathcal{C}_1$ *and* $\mathcal{C}_2$*,* $\mathcal{C}_1 \leq \mathcal{C}_2$ *if and only if,* $[\![\mathcal{C}_1]\!] \subseteq [\![\mathcal{C}_2]\!]$*.*

Theorem 1 holds provided the marked modal specifications are deterministic. If nondeterminism is allowed, refinement becomes correct but not fully abstract (the implication from right to left in Theorem 1 is not true in general). This is discussed for *unmarked* modal specifications in [15]; their counterexample can be immediately adapted to our context. Moreover, as argued in [8], nondeterministic modal specifications are not really suitable to characterize a set of *deterministic* automata.

When the left counterpart is ultimately refined, the refinement relation coincide with the implementation relation: given a terminating automaton $\mathcal{M}$ and a marked modal specification $\mathcal{C}$, $\mathcal{M} \models \mathcal{C}$ if and only if $\mathrm{E}m(\mathcal{M}) \leq \mathcal{C}$.

4 Conjunction of marked modal specification

It is a current practice, when modeling complex systems, to associate several specifications with a same system, sub-system, or component, each of them describing a different aspect of it. These so-called *viewpoints* may be engineered independently, and possibly by different teams. It is then natural to question whether different viewpoints are not contradictory and how to realize all of them. This leads to define a conjunction operator. Moreover in [7], the authors point out that, during the design cycle, a designer may be tempted to merge two interfaces which share some similarities in order to use a same implementation for the two interfaces. More formally, this corresponds to look for a shared refinement of the interfaces, if it exists.

We now define a conjunction operator which enjoy the expected properties to solve the two above problems.

Definition 8 (Conjunction). *Given two marked modal specifications* $\mathcal{C}_1 = (Q_1, q_1^0, \Sigma, \delta_1, must_1, may_1, F_1)$ *and* $\mathcal{C}_2 = (Q_2, q_2^0, \Sigma, \delta_2, must_2, may_2, F_2)$*, the conjunction of* $\mathcal{C}_1$ *and* C_2*, noted* $\mathcal{C}_1 \wedge \mathcal{C}_2$*, is the normal form* $\eta(\mathcal{C}_1 \,\&\, \mathcal{C}_2)$ *of* $\mathcal{C}_1 \,\&\, \mathcal{C}_2 = (Q, q^0, \Sigma, \delta, must, may, F)$ *with:*
– $Q = Q_1 \times Q_2$ *and* $q^0 = (q_1^0, q_2^0)$*;*
– *for any* $q_1 \in Q_1$*,* $q_2 \in Q_2$ *and* $a \in \Sigma$*,* $\delta\big((q_1, q_2), a\big) = (q_1', q_2')$ *if and only if* $\delta_1(q_1, a) = q_1'$ *and* $\delta_2(q_2, a) = q_2'$*;*
– $may(q_1, q_2) = may_1(q_1) \cap may_2(q_2)$ *and* $must(q_1, q_2) = must_1(q_1) \cup must_2(q_2)$*;*

– $(q_1, q_2) \in F$ *if and only if* $q_1 \in F_1$ *and* $q_2 \in F_2$.

Considering the manipulations done on the may/must-maps and on the transition map to obtain $\mathcal{C}_1 \,\&\, \mathcal{C}_2$, the must-saturation and the consistency may not be respected. We thus impose a normalization step in order to have an iterative process as explained at the end of Sec. 2.

Theorem 2. *Given some marked modal specifications* $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ *and* $\mathcal{C}$*:*
– $[\![\mathcal{C}_1 \wedge \mathcal{C}_2]\!] = [\![\mathcal{C}_1]\!] \cap [\![\mathcal{C}_2]\!]$*;*
– $\mathcal{C}_1 \wedge \mathcal{C}_2$ *is the greatest lower bound of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *for the refinement relation:* $\mathcal{C} \leq \mathcal{C}_1$ *and* $\mathcal{C} \leq \mathcal{C}_2$ *iff* $\mathcal{C} \leq \mathcal{C}_1 \wedge \mathcal{C}_2$*;*
– $\wedge$ *is associative:* $\mathcal{C}_1 \wedge (\mathcal{C}_2 \wedge \mathcal{C}_3) \equiv (\mathcal{C}_1 \wedge \mathcal{C}_2) \wedge \mathcal{C}_3$.

5 Product of marked modal specifications

Reachability is not preserved by product in general. Fig. 2 shows a simple example: $\mathcal{M}_1 \models \mathcal{C}_1$ and $\mathcal{M}_2 \models \mathcal{C}_2$; however the product of $\mathcal{M}_1 \times \mathcal{M}_2$ is a single non-marked state, hence the reachability of a marked state is not possible.

(a) $\mathcal{M}_1$ (b) $\mathcal{C}_1$ (c) $\mathcal{M}_2$ (d) $\mathcal{C}_2$

Fig. 2. Reachability is not compositional

This leads us to consider the following problem: given two marked modal specifications, can they be implemented concurrently i.e., such that the product of any model of the first specification with any model of the second one will always have the ability to reach a marked state of the product?

Similarly to [1], we distinguish a pessimistic from an optimistic view of composition and solve the previous problem in this two contexts.

First, in order to represent the cooperation between subsystems, a signature over Σ is now associated to any terminating automaton or marked modal specification over Σ:

Definition 9 (Signature). *Given a set of actions* Σ*, a signature over* Σ *is a mapping* $\mu : \Sigma \to \{?, !\}$ *which associates to any action either* ? *when the action is an input or* ! *when it is an output.*

Now, transitions are either labeled $!a$ (for $\mu(a) = !$) when the entity responsible for the occurrence of a is the system, or $?a$ (for $\mu(a) = ?$) if a stems from the environment of the system. The resulting formalism is thus suited to model protocols between a system and an unknown partner belonging to the system

environment. Contrarily to the input/output automata of [18] and following the interface automata of [1], terminating automata and marked modal specifications are not required to be input-enabled, meaning that some actions $?b$ of the environment may not be permitted in some state q. More formally, this situation occurs in state q if there is no outgoing transition from q labeled by $?b$. This allows to restrict, from the point of view of the system, the behavior of its environment.

Example 2. Fig. 3(a) depicts the specification of a service which can receive requests $?r$ from an unidentified subsystem in its environment and then answers by producing $!a$ until it is stopped with $?f$. It may also produce $!b$ when set in an enhanced mode by $?e$. Fig. 3(b) depicts the specification of a client which expects to receive $?a$ as an answer to any request $!r$ and is ready to receive remitted $?a$. Although $?b$ is in the signature of $\mathcal{C}_2$, there is no transition labeled $?b$ meaning that the client rejects this inputs.

We write $\Sigma^?$ and $\Sigma^!$ for the set of input and output actions, respectively, thus forming a partition of Σ. A system is then *closed* if its associated signature is such that $\Sigma^? = \emptyset$ and *open* otherwise. In this paper, we assume that if $\mathcal{M} \models \mathcal{C}$ then the signature associated to $\mathcal{M}$ and $\mathcal{C}$ is identical. Similarly, if $\mathcal{C}_1 \leq \mathcal{C}_2$ then $\mathcal{C}_1$ and $\mathcal{C}_2$ have the same signature[3].

A first condition to product is the *composability* of signatures. Given two signatures μ_1 and μ_2 over Σ_1 and Σ_2 respectively, defining two partitions $\left(\Sigma_1^?, \Sigma_1^!\right)$ and $\left(\Sigma_2^?, \Sigma_2^!\right)$, they are *composable* if no output actions is shared: $\Sigma_1^! \cap \Sigma_2^! = \emptyset$. For composable signatures, we let the *communication* actions be the set $\Sigma_{co}(\mu_1, \mu_2) = (\Sigma_1^? \cap \Sigma_2^!) \cup (\Sigma_2^? \cap \Sigma_1^!)$ which corresponds to the shared actions on which a synchronization will be possible. The set of *private* actions is $\Sigma_{pr}(\mu_1, \mu_2) = (\Sigma_1 \cup \Sigma_2) \setminus (\Sigma_1 \cap \Sigma_2)$.

Definition 10 (Product of signatures). *The product of two composable signatures μ_1 and μ_2 is $\mu = \mu_1 \times \mu_2$ defined over $\Sigma_1 \cup \Sigma_2$ such that: $\Sigma^? = (\Sigma_1^? \cup \Sigma_2^?) \setminus \Sigma_{co}(\mu_1, \mu_2)$ and $\Sigma^! = \Sigma_1^! \cup \Sigma_2^!$.*

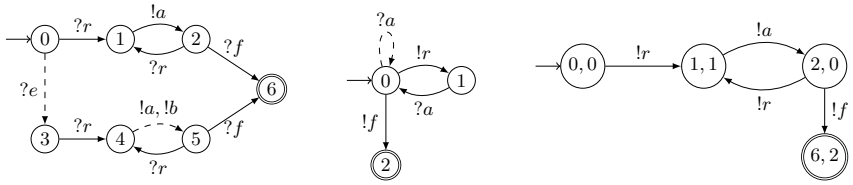
The product of two terminating automata $\mathcal{M}_1$ and $\mathcal{M}_2$ with respective composable signatures μ_1 and μ_2 is then $\mathcal{M}_1 \times \mathcal{M}_2$ as defined in Sec. 2.1 with signature $\mu_1 \times \mu_2$.

5.1 Pessimistic composition of marked modal specifications

We first consider the case of *pessimistic*[4] composition; we define a sufficient and necessary condition such that two marked modal specification can be independently implemented, the product of any of their implementations being terminating.

[3] This assumption is taken to simplify the presentation. Refinement of signature as defined in [21] can be handled in the presented theory.

[4] The pessimistic view of this approach will be made clearer in the next section.



(a) $\mathcal{C}_1$ over $\{?r, ?e, ?f, !a, !b\}$ (b) $\mathcal{C}_2$ over $\{?a, ?b, !r, !f\}$ (c) $\mathcal{C}_1 \parallel \mathcal{C}_2$ over $\{?e, !r, !f, !a, !b\}$

Fig. 3. Example of composition

This condition corresponds to the existence of a joint path to a marked state, for every reachable state of the product of arbitrary implementations. We then consider the less cooperative situation in which any optional behavior is disabled and check if such paths exist. However, the minimal behavior associated to a state of a marked modal specification is not unique in general. Consider $\mathcal{C}_1$ in Fig. 2(b), the minimal number of outgoing transition stemming from the initial state among all the models of $\mathcal{C}_1$ is 1 and can be either a transition label by a or by b. To represent the different minimal possibilities, we thus use an intermediate structure called *minimal constraint automaton*. First we define the set of minimal constraints associated to a state:

Definition 11 (Minimal constraints). *For any state q of a marked modal specification $\mathcal{C}$ defined over Σ, we associate the set $\zeta(q) \in 2^{2^\Sigma}$ defined by:*

$$\zeta(q) = \begin{cases} \{\ must(q)\ \} & \text{if } must(q) \neq \emptyset \\ \{\ \{a\} \mid a \in may(q)\ \} & \text{if } must(q) = \emptyset \text{ and } q \notin F \\ \{\ \emptyset\ \} & \text{if } must(q) = \emptyset \text{ and } q \in F \end{cases}$$

Definition 12 (Minimal constraints automaton). *Given a state q of a marked modal specification $\mathcal{C}$ over Σ, the minimal constraints automaton $\mathrm{Min}(\mathcal{C}, q)$ is the automaton over Σ whose initial state is q; its labeled transition map is λ_{Min} such that $\lambda_{\mathrm{Min}}(q', a) = q''$ if and only if $a \in X$ with $X \in \zeta(q')$ and $\delta(q', a) = q''$; its set of final states G_{Min} is the set of undelayable bottlenecks of $\mathcal{C}$.*

We identify potential dead-ends, that is pairs of states of two marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$ to be composed from which no outgoing transition may be available in a product of two respective implementations:

Definition 13 (Dead-end). *Given q_1 and q_2 two states respectively from the marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$ defined over Σ_1 and Σ_2, the pair (q_1, q_2) is a dead-end if:*

- *$q_1 \notin (F_1 \setminus D_1)$ or $q_2 \notin (F_2 \setminus D_2)$ and,*
- *there exists $X_1 \in \zeta_1(q_1)$ and $X_2 \in \zeta_2(q_2)$ such that: $(X_1 \cup (\Sigma_2 \setminus \Sigma_1)) \cap (X_2 \cup (\Sigma_1 \setminus \Sigma_2) = \emptyset$.*

Example 3. The minimal constraints associated to the initial states of $\mathcal{C}_1$ and $\mathcal{C}_2$ from Fig. 2 and defined over the same alphabet of actions $\{a, b\}$ are respectively $\{\{a\}, \{b\}\}$ and $\{\{a\}\}$. The pair formed by this two states is thus a dead-end as for $X_1 = \{b\}$ and $X_2 = \{a\}$, we have $X_1 \cap X_2 = \emptyset$.

This now leads us to a definition of *exception state pairs* from which a joint path to a marked state pair cannot be ensured independently of the implementation choices to be made:

Definition 14 (Exception state pair). *Given q_1 and q_2 two states respectively from two marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$, the pair (q_1, q_2) is an exception if:*
- *$\mathrm{M}in(\mathcal{C}_1, q_1) \times \mathrm{M}in(\mathcal{C}_2, q_2)$ is not terminating or,*
- *there exists a reachable dead-end*[5] *(q'_1, q'_2) in $\mathrm{M}in(\mathcal{C}_1, q_1) \times \mathrm{M}in(\mathcal{C}_2, q_2)$.*

We denote by $\mathrm{E}x(\mathcal{C}_1, \mathcal{C}_2)$ the set of exception state pairs from $\mathcal{C}_1$ and $\mathcal{C}_2$. Then we can define the following criterion characterizing marked modal specifications having compatible reachability:

Definition 15 (Compatible reachability). *Two marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$ have a compatible reachability, noted $\mathcal{C}_1 \sim_{\mathcal{T}} \mathcal{C}_2$, if there is no exception state pair that is reachable in $\mathrm{U}n(\mathcal{C}_1) \times \mathrm{U}n(\mathcal{C}_2)$.*

The soundness and the completeness of the previous definition are then stated by the following Theorem:

Theorem 3 (Independent implementability). *Given two marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$, $\mathcal{C}_1 \sim_{\mathcal{T}} \mathcal{C}_2$ if and only if for any $\mathcal{M}_1 \models \mathcal{C}_1$ and $\mathcal{M}_2 \models \mathcal{C}_2$, the product $\mathcal{M}_1 \times \mathcal{M}_2$ is terminating.*

We now define the product of two marked modal specifications with compatible reachability.

Definition 16 (Pessimistic product). *Given two marked modal specifications $\mathcal{C}_1 = (Q_1, q_1^0, \Sigma_1, \delta_1, must_1, may_1, F_1)$ and $\mathcal{C}_2 = (Q_2, q_2^0, \Sigma_2, \delta_2, must_2, may_2, F_2)$ with compatible reachability, the product $\mathcal{C}_1 \otimes \mathcal{C}_2$ is the marked modal specification $(Q, q^0, \Sigma_1 \cup \Sigma_2, \delta, must, may, F)$ with:*
- *$Q = Q_1 \times Q_2$ and $q^0 = (q_1^0, q_2^0)$;*
- *for any $q_1 \in Q_1$, $q_2 \in Q_2$ and $a \in \Sigma_1 \cup \Sigma_2$, $\delta\big((q_1, q_2), a\big)$ is defined as $(\delta_1(q_1, a), \delta_2(q_2, a))$ for $a \in \Sigma_1 \cap \Sigma_2$, $(\delta_1(q_1, a), q_2)$ for $a \in \Sigma_1 \setminus \Sigma_2$ and $(q_1, \delta_2(q_2, a))$ for $a \in \Sigma_2 \setminus \Sigma_1$;*
- *$may((q_1, q_2)) = \big(may_1(q_1) \cup (\Sigma_2 \setminus \Sigma_1)\big) \cap \big(may_2(q_2) \cup (\Sigma_1 \setminus \Sigma_2)\big)$;*
- *$must((q_1, q_2)) = \big(must_1(q_1) \cup (\Sigma_2 \setminus \Sigma_1)\big) \cap \big(must_2(q_2) \cup (\Sigma_1 \setminus \Sigma_2)\big)$;*
- *$(q_1, q_2) \in F$ if and only if $q_1 \in F_1$ and $q_2 \in F_2$.*

[5] As the set of states of $\mathrm{M}in(\mathcal{C}_i, q_i)$ is a subset of these of $\mathcal{C}_i$, we can refer to (q'_1, q'_2) in $\mathrm{M}in(\mathcal{C}_1, q_1) \times \mathrm{M}in(\mathcal{C}_2, q_2)$ as a pair of states of $\mathcal{C}_1$ and $\mathcal{C}_2$ and then test if it is a dead-end in the sense of Def. 13.

Now, the product of any models $\mathcal{M}_1$ of $\mathcal{C}_1$ and $\mathcal{M}_2$ of $\mathcal{C}_2$ is model of $\mathcal{C}_1 \otimes \mathcal{C}_2$:

Proposition 2. *Given two marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$, if $\mathcal{C}_1 \sim_{\mathcal{T}} \mathcal{C}_2$ then for any $\mathcal{M}_1 \models \mathcal{C}_1$ and $\mathcal{M}_2 \models \mathcal{C}_2$, $\mathcal{M}_1 \times \mathcal{M}_2 \models \mathcal{C}_1 \otimes \mathcal{C}_2$.*

Moreover, $\mathcal{C}_1 \otimes \mathcal{C}_2$ gives the most precise characterization of the behavior of the product of any models $\mathcal{M}_1$ of $\mathcal{C}_1$ and $\mathcal{M}_2$ of $\mathcal{C}_2$:

Proposition 3. *Given two marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$, if $\mathcal{C}_1 \sim_{\mathcal{T}} \mathcal{C}_2$ and if there exists a marked modal specification $\mathcal{C}$ such that for any $\mathcal{M}_1 \models \mathcal{C}_1$ and $\mathcal{M}_2 \models \mathcal{C}_2$ we have $\mathcal{M}_1 \times \mathcal{M}_2 \models \mathcal{C}$ then $\mathcal{C}_1 \otimes \mathcal{C}_2 \leq \mathcal{C}$.*

One important principle in modular and concurrent design of systems is the fact that a property checked on a primary version of some system artifacts remains true on any refined version of them. This is what allows to guarantee that the system parts corresponding to compatible interfaces can be designed concurrently. This is respected for compatible reachability:

Proposition 4. *For all marked modal specifications $\mathcal{C}_1$, $\mathcal{C}'_1$ and $\mathcal{C}_2$, if $\mathcal{C}_1 \sim_{\mathcal{T}} \mathcal{C}_2$ and $\mathcal{C}'_1 \preceq \mathcal{C}_1$ then $\mathcal{C}'_1 \sim_{\mathcal{T}} \mathcal{C}_2$ and $\mathcal{C}'_1 \otimes \mathcal{C}_2 \preceq \mathcal{C}_1 \otimes \mathcal{C}_2$.*

Last, the product is a commutative and associative operator, meaning that interfaces can be assembled in any order without affecting the result.

Proposition 5. *The product of marked modal specifications is commutative and associative. Given three marked modal specifications $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$: $\mathcal{C}_1 \otimes \mathcal{C}_2 \equiv \mathcal{C}_2 \otimes \mathcal{C}_1$ and $\mathcal{C}_1 \otimes (\mathcal{C}_2 \otimes \mathcal{C}_3) \equiv (\mathcal{C}_1 \otimes \mathcal{C}_2) \otimes \mathcal{C}_3$.*

5.2 Optimistic composition of marked modal specifications

Consider again $\mathcal{C}_1$ and $\mathcal{C}_2$ from Fig. 3. They do not have a compatible reachability as $(3, 0)$ is an exception state pairs because $(4, 1)$ is a reachable dead-end from it. It is however pessimistic to declare $\mathcal{C}_1$ and $\mathcal{C}_2$ as not composable. Indeed, the system potentially formed by any model of $\mathcal{C}_1$ and $\mathcal{C}_2$ would not be closed as the occurence of $?e$ would still be under the control of the environment. Now by preventing the environment from producing $!e$ when $\mathcal{C}_1$ and $\mathcal{C}_2$ are in their initial state, the reachability of the exception state pairs $(4, 1)$ can be avoided. In this section, let us now be optimistic and declare composable any $\mathcal{C}_1$ and $\mathcal{C}_2$ if there exists *at least one* environment, closing the system and preventing the reachability of the *bad* states of $\mathcal{C}_1$ and $\mathcal{C}_2$ in which the reachability property cannot be guaranteed.

Definition 17 (Legal environment). *Given $\mathcal{M}$ and $\mathcal{E}$ two terminating automata, $\mathcal{E}$ is said to be a* legal *environment for $\mathcal{M}$, if and only if: the signature of $\mathcal{M}$ and $\mathcal{E}$ are composable; $\mathcal{M} \times \mathcal{E}$ is closed;* $\mathrm{Em}(\mathcal{M}) \sim_{\mathcal{T}} \mathrm{Em}(\mathcal{E})$*, that is $\mathcal{M} \times \mathcal{E}$ is terminating.*

Next, we define, for any automaton $\mathcal{M}$ (terminating or not) with $r^0 \in pre^*(G)$, the subautomaton $\mathcal{M}^\star = (pre^*(G), r^0, \Sigma, \lambda^\star, G)$ where $\lambda^\star(r,a) = r'$ if and only if $\lambda(r,a) = r'$ and $r, r' \in pre^*(G)$. It corresponds to the potential reachable part of $\mathcal{M}$ when interacting with a legal environment.

Definition 18 (Optimistic compatible reachability). *Two marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$ have an optimistic compatible reachability, noted $\mathcal{C}_1 \sim_{\mathcal{O}} \mathcal{C}_2$ if the pair of initial states (q_1^0, q_2^0) is not an exception state pairs.*

This criterion is sound and complete as stated by the following Theorem:

Theorem 4 (Independent implementability). *Given two marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$, $\mathcal{C}_1 \sim_{\mathcal{O}} \mathcal{C}_2$ if and only if for any $\mathcal{M}_1 \models \mathcal{C}_1$ and $\mathcal{M}_2 \models \mathcal{C}_2$ there exists a legal environment $\mathcal{E}$ for $\mathcal{M}_1 \times \mathcal{M}_2$.*

Definition 19 (Optimistic product). *Given two marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$ over composable signatures μ_1 and μ_2 and with optimistic compatible reachability, the optimistic product $\mathcal{C}_1 \parallel \mathcal{C}_2$ is the normal form of the marked modal specification $(Q, q^0, \Sigma_1 \cup \Sigma_2, \delta, must, may, F)$ over $\mu_1 \times \mu_2$ with:*

- *$Q = (Q_1 \times Q_2) \setminus \mathrm{E}x(\mathcal{C}_1, \mathcal{C}_2)$ and $q^0 = (q_1^0, q_2^0)$;*
- *for any $q_1 \in Q_1$, $q_2 \in Q_2$ and $a \in \Sigma_1 \cup \Sigma_2$:*
  - *if $a \in \Sigma_1 \setminus \Sigma_2$ and $(\delta_1(q_1,a), q_2) \notin \mathrm{E}x(\mathcal{C}_1, \mathcal{C}_2)$: $\delta\big((q_1,q_2),a\big) = (\delta_1(q_1,a), q_2)$;*
  - *if $a \in \Sigma_2 \setminus \Sigma_1$ and $(q_1, \delta_2(q_2,a)) \notin \mathrm{E}x(\mathcal{C}_1, \mathcal{C}_2)$: $\delta\big((q_1,q_2),a\big) = (q_1, \delta_2(q_2,a))$;*
  - *if $a \in \Sigma_1 \cap \Sigma_2$ and $(\delta_1(q_1,a), \delta_2(q_2,a)) \notin \mathrm{E}x(\mathcal{C}_1, \mathcal{C}_2)$: $\delta\big((q_1,q_2),a\big) = (\delta_1(q_1,a), \delta_2(q_2,a))$.*
- *$a \in must((q_1,q_2))$ if $a \in \big(must_1(q_1) \cup (\Sigma_2 \setminus \Sigma_1)\big) \cap \big(must_2(q_2) \cup (\Sigma_1 \setminus \Sigma_2)\big)$ and $\delta\big((q_1,q_2),a\big)$ is defined;*
- *$(q_1,q_2) \in F$ if and only if $q_1 \in F_1$ and $q_2 \in F_2$.*

Example 4. The optimistic product of $\mathcal{C}_1$ and $\mathcal{C}_2$ from Fig. 3 is depicted in Fig. 3(c). The action $?e$ is not allowed in the initial state as a legal environement would never produce $!e$ to prevent the reachability of the exception states $(3,0)$.

Proposition 6. *Given two marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$, if $\mathcal{C}_1 \sim_{\mathcal{O}} \mathcal{C}_2$ then for any $\mathcal{M}_1 \models \mathcal{C}_1$ and $\mathcal{M}_2 \models \mathcal{C}_2$, $(\mathcal{M}_1 \times \mathcal{M}_2)^\star \models \mathcal{C}_1 \parallel \mathcal{C}_2$.*

The next proposition states that $\mathcal{C}_1 \parallel \mathcal{C}_2$ is the minimal marked modal specification w.r.t. refinement enjoying the independent implementability property:

Proposition 7. *Given two marked modal specifications $\mathcal{C}_1$ and $\mathcal{C}_2$, if $\mathcal{C}_1 \sim_{\mathcal{O}} \mathcal{C}_2$ and if there exists a marked modal specification $\mathcal{C}$ such that for any $\mathcal{M}_1 \models \mathcal{C}_1$ and $\mathcal{M}_2 \models \mathcal{C}_2$ there exists a legal environment $\mathcal{E}$ for $\mathcal{M}_1 \times \mathcal{M}_2$ and $(\mathcal{M}_1 \times \mathcal{M}_2)^\star \models \mathcal{C}$, then $\mathcal{C}_1 \parallel \mathcal{C}_2 \leq \mathcal{C}$.*

Optimistic compatible reachability is preserved by refinement hence allowing concurrent design of sub-systems. Moreover, the optimistic product is monotonic with respect to the refinement relation and is also associative which guarantees independence in the design flow.

Proposition 8. *For all marked modal specifications $\mathcal{C}_1$, $\mathcal{C}'_1$ and $\mathcal{C}_2$, if $\mathcal{C}_1 \sim_{\mathcal{O}} \mathcal{C}_2$ and $\mathcal{C}'_1 \preceq \mathcal{C}_1$ then $\mathcal{C}'_1 \sim_{\mathcal{O}} \mathcal{C}_2$ and $\mathcal{C}'_1 \parallel \mathcal{C}_2 \preceq \mathcal{C}_1 \parallel \mathcal{C}_2$.*

Proposition 9. *The optimistic product of marked modal specifications is commutative and associative. Given three marked modal specifications $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$: $\mathcal{C}_1 \parallel \mathcal{C}_2 \equiv \mathcal{C}_2 \parallel \mathcal{C}_1$ and $\mathcal{C}_1 \parallel (\mathcal{C}_2 \parallel \mathcal{C}_3) \equiv (\mathcal{C}_1 \parallel \mathcal{C}_2) \parallel \mathcal{C}_3$.*

6 Related works and conclusion

Marked modal specifications can be used to express, in a modular manner, that a system should be capable of reaching one or several marked states representing either the completion of a composition of services or the quiescence of a network of interacting agents. They improve the expressive power of deterministic modal specifications that corresponds to the conjunctive ν-calculus [10] which does not allow to capture reachability properties.

The same goal can be achieved with automata-theoretic specifications in which states are annotated with propositional formulas expressing implementation variants and, possibly, an obligation of progress. This is the case of *annotated automata* [22] and *operating guidelines* [19,17]. While both formalisms have a product (or parallel) composition operator, they are missing the optimistic view of composition and also the conjunction operator that turns out to be instrumental as soon as components are described according to several distinct but interacting viewpoints [21].

The disjunctive variants of modal specifications [13,9] allows to constraint progress and thus to inductively express reachability. However no implementation relations including marked states nor optimistic composition have been proposed for these variants of modal specifications.

Marked modal specifications look similar to the modal specifications with marked states introduced in [6]. However, these two formalisms are very different because the satisfaction relation in [6] admits implementations having final states corresponding to a state of the specification that is not final. This is appropriate in the context of supervisory control synthesis. However, this semantics does not seem well-suited to a specification algebra with a refinement preorder, which explains why a different satisfaction relation is used for marked modal specifications.

References

1. de Alfaro, L., Henzinger, T.A.: Interface automata. In: Proc. of the 9th ACM SIGSOFT Inter. Symp. on Foundations of Software Engineering (FSE'01). pp. 109–120. ACM Press (2001)
2. Alur, R., Henzinger, T.A., Kupferman, O., Vardi, M.Y.: Alternating refinement relations. In: Proc. of the 9th Inter. Conf. on Concurrency Theory (CONCUR'98). LNCS, vol. 1466, pp. 163–178. Springer (1998)
3. Antonik, A., Huth, M., Larsen, K.G., Nyman, U., Wasowski, A.: 20 years of modal and mixed specifications. Bulletin of the EATCS 1(94) (2008)

4. Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, Ph.: Systems and Software Verification. Model-Checking Techniques and Tools. Springer (2001)
5. Caillaud, B., Raclet, J.B.: Ensuring reachability by design. Tech. rep., INRIA Research Report 7928 (2012), http://hal.inria.fr/hal-00696151
6. Darondeau, P., Dubreil, J., Marchand, H.: Supervisory control for modal specifications of services. In: Workshop on Discrete Event Systems (WODES'10). pp. 428–435. Berlin, Germany (August 2010)
7. Doyen, L., Henzinger, T.A., Jobstmann, B., Petrov, T.: Interface theories with component reuse. In: Proc. of the 8th Inter. Conf. on Embedded Software (EMSOFT'08). pp. 79–88. ACM Press (2008)
8. Fecher, H., de Frutos-Escrig, D., Lüttgen, G., Schmidt, H.: On the expressiveness of refinement settings. In: Proc. of the 3rd Inter. Conf. on Fundamentals of Software Engineering (FSEN'09). LNCS, vol. 5961, pp. 276–291. Springer (2009)
9. Fecher, H., Schmidt, H.: Comparing disjunctive modal transition systems with an one-selecting variant. J. Log. Algebr. Program. 77(1-2), 20–39 (2008)
10. Feuillade, G., Pinchinat, S.: Modal specifications for the control theory of discrete-event systems. Discrete Event Dynamic Systems 17(2), 181–205 (2007)
11. Henzinger, T.A., Sifakis, J.: The discipline of embedded systems design. IEEE Computer 40(10), 32–40 (2007)
12. Larsen, K.G., Nyman, U., Wasowski, A.: Modal I/O automata for interface and product line theories. In: Proc. of the 16th Euro. Symp. on Programming (ESOP'07). LNCS, vol. 4421, pp. 64–79. Springer (2007)
13. Larsen, K.G., Xinxin, L.: Equation solving using modal transition systems. In: Proc. of the 5th IEEE Symp. on Logic in Computer Science, LICS'90. pp. 108–117. IEEE Computer Society Press (1990)
14. Larsen, K.G.: Modal specifications. In: Automatic Verification Methods for Finite State Systems. LNCS, vol. 407, pp. 232–246. Springer (1989)
15. Larsen, K.G., Nyman, U., Wasowski, A.: On modal refinement and consistency. In: Proc. of the 18th Inter. Conf. on Concurrency Theory (CONCUR'07). pp. 105–119. Springer (2007)
16. Larsen, K.G., Thomsen, B.: A modal process logic. In: Proc. of the 3rd Annual Symp. on Logic in Computer Science (LICS'88). pp. 203–210. IEEE (1988)
17. Lohmann, N., Wolf, K.: Compact representations and efficient algorithms for operating guidelines. Fundam. Inform. 108(1-2), 43–62 (2011)
18. Lynch, N., Tuttle, M.R.: An introduction to Input/Output automata. CWI-quarterly 2(3), 219–246 (1989)
19. Massuthe, P., Schmidt, K.: Operating guidelines - an automata-theoretic foundation for the service-oriented architecture. In: QSIC. pp. 452–457. IEEE Computer Society (2005)
20. Raclet, J.B., Badouel, E., Benveniste, A., Caillaud, B., Legay, A., Passerone, R.: Modal interfaces: unifying interface automata and modal specifications. In: Proc. of the 9th Int. Conf. on Embedded Software (EMSOFT'09). pp. 87–96. ACM (2009)
21. Raclet, J.B., Benveniste, A., Caillaud, B., Legay, A., Passerone, R.: A modal interface theory for component-based design. Fundam. Inform. 107, 1–32 (2011)
22. Wombacher, A., Mahleko, B., Neuhold, E.J.: IPSI-PF - a business process matchmaking engine based on annotated finite state automata. Inf. Syst. E-Business Management 3(2), 127–150 (2005)