A Specification Theory for Reachability by Design

Guillaume Verdier · Benoît Caillaud · Jean-Baptiste Raclet

Abstract Modular design aims at building complex reactive systems by assembling components, possibly taken off-the-shelf. This approach can be supported by a specification theory in which requirements correspond to specifications while components are models of the specifications.

In this article, we consider components which have to fulfill a reachability objective. A difficulty is then induced by the fact that reachability properties are not compositional. The approach we advocate consists in controlling the design flow of components, that is, the evolution of their specifications through combinations and refinements, in order to ensure a reachability property by construction.

We define specifications in terms of marked acceptance specifications which are automata enriched with variability information encoded by acceptance sets and with reachability constraints. We then develop a specification theory with both logical and structural composition operators and with quotient, ensuring reachability properties by design.

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 approach can be supported by a specification theory in which requirements correspond to specifications while components are models of the specifications.

In this article, we develop a specification theory for components which have to fulfill a reachability objective and propose 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 [8], 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 article 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.

Modal specifications (MS) [20, 19, 1] are widely acknowledged as a suitable specification formalism to develop specification theories [27, 12, 4], also in the timed [9, 11, 18] and quantitative [2, 3] contexts. In this article, we introduce marked acceptance specifications (MAS). They generalize MS first by offering the ability to specify what combination of transitions are allowed in a model, secondly by adding marked states representing reachability objectives.

We define for MAS the algebra expected for a complete specification theory including:

– refinement: it allows to test the safe substitution of a MAS by a more elaborate version of an initial requirement;
– product: this structural composition operator is determined by a compatibility criterion ensuring that compatible MAS can be refined independently and then composed, their product will satisfy a reachability objective. This principle, called independent implementability, is of key importance [17] and enables the concurrent design of systems that are then assembled in a bottom-up manner;
– conjunction: this logical composition operator enables viewpoint-design of components and merge of specifications sharing some similarities;
– quotient: this operator is a crucial feature for incremental design of component-based system. It enables top-down design by decomposing high-level specifications and allowing to test if a preexisting component can be reused.

Defining quotient leads us to solve an intermediate problem: the automatic refinement of a set of MAS whose independent design and product cannot guarantee the reachability objectives. In other words, it corresponds to enforcing the compatibility criterion allowing the product operation. This proves to be very interesting for the system designer to correct discrepant specifications.

Outline of the article. We first recall some definitions about automata and introduce MAS in Section 2. Each of the next sections are dedicated to an element of the specification theory for MAS: Section 3 for the refinement relation; Section 4 for the product and its associated compatibility criterion; Section 5 for the conjunction; Section 6 for the quotient. Section 7 includes a discussion on the limit of modal specifications, what could have been done and not done in this setting. Section 8 concludes the paper.
2 Modeling with marked acceptance specifications

In order to leave no ambiguity and to fix notations, we first recall some standard definitions on automata.

2.1 Background on automata

Let $\Sigma$ be a fixed finite alphabet of actions, a (deterministic) automaton over $\Sigma$ is a tuple $M = (R, r^0, \lambda, G)$ where $R$ is a finite set of states, $r^0 \in R$ is the unique initial state, $\lambda$ is a partial map from $R \times \Sigma$ to $R$ called labeled transition map and $G \subseteq R$ is the set of marked states. The set of fireable actions from a state $r$, denoted $\text{ready}(r)$, is the set of actions $a$ such that $\lambda(r, a)$ is defined. A transition $\lambda(r, a) = r'$ may be denoted $(r, a, r')$ or, for short and without ambiguity as automata are deterministic, $(r, a)$.

Given a state $r$, we define $\text{pre}^*(r)$ and $\text{post}^*(r)$ as the set of states that are respectively coreachable and reachable from $r$: they are the least sets such that $r \in \text{pre}^*(r)$, $r \in \text{post}^*(r)$ and for any $r'$, $a$ and $r''$ such that $\lambda(r', a) = r''$, $r' \in \text{pre}^*(r)$ if $r'' \in \text{pre}^*(r)$ and $r'' \in \text{post}^*(r)$ if $r' \in \text{post}^*(r)$. We also define $\text{pre}^+(r)$ as the union of $\text{pre}^-(r'')$ for all $r''$ such that $\exists a, \lambda(r', a) = r$ and $\text{post}^+(r)$ as the union of $\text{post}^-(\lambda(r, a))$ for all $a \in \text{ready}(r)$. Let $\text{Loop}(r) = \text{pre}^-(r) \cap \text{post}^+(r)$.

Given an automaton $M$ and a state $r$ of $M$, $r$ is a deadlock if $r \not\in G$ and $\text{ready}(r) = \emptyset$; $r$ belongs to a livelock if $\text{Loop}(r) \neq \emptyset$, $G \cap \text{Loop}(r) = \emptyset$ and there is no transition $\lambda(r', a) = r''$ such that $r' \in \text{Loop}(r)$ and $r'' \not\in \text{Loop}(r)$. If no state of an automaton $M$ is a deadlock then $M$ is said deadlock-free. If no state of $M$ belongs to a livelock then $M$ is livelock-free. An automaton is terminating if it is deadlock-free and livelock-free.

Two automata $M_1$ and $M_2$ are bisimilar if there exists a simulation relation $\pi \subseteq R_1 \times R_2$ such that $(r_1^0, r_2^0) \in \pi$ and for all $(r_1, r_2) \in \pi$, we have $\text{ready}(r_1) = \text{ready}(r_2)$, $r_1 \in G_1$ if and only if $r_2 \in G_2$ and for any $a \in \text{ready}(r_1)$, $(\lambda_1(r_1, a), \lambda_2(r_2, a)) \in \pi$.

The product of two automata $M_1$ and $M_2$, denoted $M_1 \times M_2$, is the automaton $(R_1 \times R_2, (r_1^0, r_2^0), \lambda, G_1 \times G_2)$ where $\lambda((r_1, r_2), a) = (\lambda_1(r_1, a), \lambda_2(r_2, a))$ when both $\lambda_1(r_1, a)$ and $\lambda_2(r_2, a)$ are defined.

2.2 Marked acceptance specifications

Acceptance trees have been introduced in [16] and considered in [26] as a specification formalism, called acceptance specifications, which generalizes modal specifications [24]. They are also named Boolean MS in [6]. We now enrich acceptance specifications 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 terminating automata called models (or sometimes implementations).

Definition 1 (Marked Acceptance Specification) A marked acceptance specification (MAS) over $\Sigma$ is a tuple $S = (Q, q^0, \delta, \text{Acc}, F)$ where $Q$ is a finite set of states, $q^0 \in Q$ is the unique initial state, $\delta : Q \times \Sigma \rightarrow Q$ is the labeled transition
map, $\text{Acc} : Q \rightarrow 2^{2^\Sigma}$ associates to each state a set of ready sets which is called its acceptance set and $F \subseteq Q$ is a set of marked states.

Basically, a ready set corresponds to a set of actions a model of the specification is ready to engage in; an acceptance set can thus be seen as a list of possible choices made available by the system designer.

**Example 1** Let $\Sigma = \{a, b, c, d\}$. Figure 1 and Figure 2 depict two MAS over $\Sigma$ in which marked states are double-circles.

In this paper, MAS 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 Section 3.

The underlying automaton associated to $S$ is $\text{Un}(S) = (Q, q^0, \delta, F)$. The sets $\text{pre}^*(q)$, $\text{post}^*(q)$, $\text{pre}^+(q)$, $\text{post}^+(q)$, $\text{ready}(q)$ and $\text{Loop}(q)$ for any state $q$ of $S$ corresponds respectively to $\text{pre}^*(q)$, $\text{post}^*(q)$, $\text{pre}^+(q)$, $\text{post}^+(q)$, $\text{ready}(q)$ and $\text{Loop}(q)$ in $\text{Un}(S)$, as defined in Section 2.1.

**Definition 2 (Satisfaction)** A terminating automaton $M$ satisfies a MAS $S$, denoted $M \models S$, 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 $(r, q) \in \pi$:
- $\text{ready}(r) \in \text{Acc}(q)$;
- if $r \in G$ then $q \in F$, and;
- for any $a \in \text{ready}(r)$, we have: $(\lambda(r, a), \delta(q, a)) \in \pi$.

**Example 2** Two models $M_1$ and $M_2$ of the MAS in Figure 1 are depicted in Figure 3. The corresponding simulation relations justifying satisfaction are $\pi_1 = \{(0', 0), (1', 1), (2', 2)\}$ and $\pi_2 = \{(0''', 0), (1''', 1), (2''', 0), (3''', 1), (4''', 2)\}$.

Observe that the transitions labeled by $b$ and $c$ and $d$ are optional in state 0 and in state 1 from the MAS $S$ as these actions are not present in all sets in $\text{Acc}(0)$ and $\text{Acc}(1)$ and thus may not be present in any model of the specification. Moreover, state 3 in $S$ is marked to encode the constraint that it must be reached in any model. As a result, although the actions $b$, $c$ and $d$ are optional, at least
Fig. 3: Two models $M_1$ and $M_2$ of the MAS $S$ in Figure 1.

Fig. 4: A model $M$ of the MAS $S$ in Figure 2.

one of the three must be present in any model of $S$. This kind of constraint entails that MAS are more expressive than modal specifications.

The terminating automata $M$ in Figure 4 satisfies the MAS in Figure 2 because of the simulation relation $\pi = \{(0', 0), (1', 1), (2', 2), (3', 1)\}$.

Observe that, according to the second item of Definition 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 $1'$.

The set of models of $S$ is denoted $\mathcal{J}_S$. A MAS is said satisfiable if and only if $\mathcal{J}_S \neq \emptyset$. Two MAS $S_1$ and $S_2$ are said equivalent, written $S_1 \equiv S_2$, if and only if they admit the same implementations: $\mathcal{J}_{S_1} = \mathcal{J}_{S_2}$. Any unsatisfiable specification is mapped on a special specification denoted $S_\bot$, with $\mathcal{J}_{S_\bot} = \emptyset$.

The introduced semantic induces some simplifications in the structure of the MAS that we discuss now. This will then lead to the definition of an associated normal form.

- **Attractability.** A MAS is said attracted in $q$ when $\text{post}^*(q) \cap F \neq \emptyset$.
- **Acc-consistency.** A state $q$ is Acc-consistent when $\text{Acc}(q) \neq \emptyset$.
- **$F$, Acc-consistency.** A state $q$ is $F$, Acc-consistent when $\emptyset \in \text{Acc}(q)$ implies $q \in F$.
- **$\delta$, Acc-consistency.** A state $q$ is $\delta$, Acc-consistent when, for any action $a \in \Sigma$, $\delta(q, a)$ is defined if and only if there exists an $X \in \text{Acc}(q)$ such that $a \in X$.

**Remark 1** When $\text{Acc}(q) = \emptyset$, $q$ cannot belong to a simulation relation stating that $M \models S$ as we cannot find an $X \in \text{Acc}(q)$ such that $\text{ready}(r) = X$ for some $r$.

When $\emptyset \in \text{Acc}(q)$ and $(r, q) \in \pi$, we can have $\text{ready}(r) = \emptyset$ that is, there is no outgoing transition from $r$. As $M$ is terminating, this requires that $r$ is marked and thus, $q$ is also marked.

**Definition 3 (Normal form)** A MAS is in normal form if it is attracted, Acc-consistent, $F$, Acc-consistent and $\delta$, Acc-consistent in every state $q$. Moreover, $S_\bot$ is in normal form.

**Theorem 1** Every marked acceptance specification is equivalent to a marked acceptance specification in normal form.
Proof of this proposition is by construction of a MAS in normal form $\rho(S)$ and then proving that $S$ and $\rho(S)$ are equivalent. This construction is detailed in Algorithm 1 which returns a MAS corresponding to $\rho(S)$; it defines a pruning operation which removes all the states which are not attracting or not Acc-consistent and updates the acceptance sets and the transition map to enforce $F$, Acc-consistency and $\delta$, Acc-consistency.

**Proof** Following Definition 3, four cases may lead to detect that a MAS $S$ is not in normal form. For each of them, we associate a construction rule to obtain $\rho(S)$ that we analyze now:

- if $\neg$ Acc-consistent($q$) then $q$ cannot belong to a simulation relation stating that $M \models S$ as explained in Remark 1. We thus remove the state from $S$ (lines 6 to 10 of Algorithm 1);

- if $\neg$ attracted($q$) then no marked state is reachable from $q$ in $S$. We thus cannot include $q$ in a simulation relation to build a terminating model. As in the previous case, we remove $q$ from $S$ (lines 3 to 10 of Algorithm 1);

- if $\neg F$, Acc-consistent($q$), we remove $\emptyset$ from Acc($q$) (line 14 of Algorithm 1). This is a direct consequence of the fact that $\emptyset \in$ Acc($q$) is only relevant when $q$ is marked as advocated in Remark 1;

- last, the combination of items 1 and 3 of Definition 2 indicates that a ready set $X$ is relevant in $q$ if and only if the transition map is defined from $q$ for any $a \in X$. In lines 17 to 21 of Algorithm 1, we then update the acceptance sets and the transition map to make them consistent.

As a result, none of these four cases affects the set of models of the received MAS. These construction rules are iteratively applied until a fix-point is reached. The algorithm always finishes as at least the (finite) number of states or the (finite) size of the acceptance sets strictly decreases. $\square$

As a result of Theorem 1, from now on and without loss of generality, we always assume that MAS are in normal form.

At this point, the reader may wonder why attractability and the previous different forms of consistency are not fully part of the definition of MAS. The reason for this is because, in what follows, we propose composition operators on MAS and it is easier to define these constructions without trying to preserve these different requirements. Now if the combination of two MAS (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 acceptance specifications

A refinement relation aims at relating specifications at different stages of their design. Basically, it should correspond to refining the set of allowed implementations of a specification. Moreover, we shall see later that refinement should entail substitutability, meaning that the substitution of a specification $S_2$ by a refined version $S_1$ must not impact the possible and actual cooperation with other implementations that have been previously declared as legal for $S_2$. 

---

**Note:** The content provided is a natural language representation of the text from the image, focusing on the key points and concepts discussed. The full context and details can be found in the original document.
Algorithm 1 normal form (S: MAS): MAS

1: $S' \leftarrow S$
2: repeat
3:   unchanged $\leftarrow$ true
4:   for all $q' \in Q'$ do
5:     if $\neg$ attracted($q'$) $\lor$ $\neg$ Acc-consistent($q'$) then
6:       unchanged $\leftarrow$ false
7:     for all $q \in Q$ such that $\delta'(q, a) = q'$ do
8:       $\delta'(q, a) = \text{undefined}$
9:   $Q' \leftarrow Q' \setminus \{q'\}$
10: end if
11: if $\neg F$, Acc-consistent($q'$) then
12:   unchanged $\leftarrow$ false
13: Acc'($q'$) $\leftarrow$ Acc'($q'$) \{∅\}
14: end if
15: if $\neg \delta$, Acc-consistent($q'$) then
16:   unchanged $\leftarrow$ false
17: for all $a \in \text{ready}'(q') \setminus \bigcup \text{Acc}'(q')$ do
18:   $\delta'(q', a) = \text{undefined}$
19: end for
20: Acc'($q'$) $\leftarrow \{X \in \text{Acc}'(q') \mid \forall a \in X, \delta'(q', a) \text{ is defined} \}$
21: end if
22: if $Q' = \emptyset$ then
23:   return $S'$
24: end if
25: end for
26: until unchanged
27: return $S'$

Fig. 5: A MAS $S'$ refining the MAS $S$ in Figure 1

Definition 4 (Refinement) Given two MAS $S_1$ and $S_2$, $S_1$ is a refinement of $S_2$, denoted $S_1 \leq S_2$, if and only if there exists a simulation relation $\pi \subseteq Q_1 \times Q_2$ such that $(q_1', q_2') \in \pi$ and for all pairs $(q_1, q_2) \in \pi$:
- $\text{Acc}_1(q_1) \subseteq \text{Acc}_2(q_2)$;
- if $q_1 \in F_1$ then $q_2 \in F_2$;
- for any $a \in \text{ready}(q_1)$, we have: $(\delta_1(q_1, a), \delta_2(q_2, a)) \in \pi$.

Intuitively, refining a specification corresponds to removing ready sets from acceptance sets while preserving the marked nature of states. This relation is reflexive and transitive and it is thus a preorder.

Example 3 The MAS in Figure 5 is a refinement of the MAS in Figure 1 because of the simulation relation $\pi = \{(0', 0), (1', 1), (2', 0), (3', 1), (4', 2), (5', 2)\}$.

Theorem 2 Given two MAS $S_1$ and $S_2$, $S_1 \leq S_2$ if and only if $[S_1] \subseteq [S_2]$.
As a result, according to Definition 4, we have to discussed for unmarked (the implication from right to left in Theorem 2 is not true in general). This is tic. If nondeterminism is allowed, refinement becomes correct but not fully abstract model of the second one?

In a specification theory, this requires to lift the product operator from implementations to specifications; the product of two specifications then characterizes the behavior realized by the product of any implementations.

In our context, a difficulty is induced by the fact that reachability is not pre-

This leads us to first consider the following problem: given two MAS, can they be implemented concurrently i.e., such that it will be possible to reach a marked state from any state of the product of any model of the product with any model of the second one?

Proof ($\Rightarrow$) Suppose that $S_1 \subseteq S_2$ and $M \models S_1$ thanks respectively to the simulation relations $\pi$ and $\pi_1$. Define $\pi_2$ such that $(r,q_1) \in s_2$ if and only if there exists a state $q_1$ in $S_1$ such that $(r,q_1) \in \pi_1$ and $(q_1,q_2) \in \pi$. We prove that $M \models S_2$ thanks to $\pi_2$: 

- if $(r,q_1) \in \pi_1$ then $\text{ready}(r) \in \text{Acc}_1(q_1)$ by Definition 2; moreover, if $(q_1,q_2) \in \pi$ then $\text{Acc}_1(q_1) \subseteq \text{Acc}_2(q_2)$ by Definition 4. As a result, $\text{ready}(r) \in \text{Acc}_2(q_2)$;
- if $(r,q_1) \in \pi_1$ then $r \in G$ implies $q_2 \in F_2$ by Definition 2; moreover, if $(q_1,q_2) \in \pi$ then $q_1 \in F_1$ implies $q_2 \in F_2$ by Definition 4. As a result, $r \in G$ implies $q_2 \in F_2$;
- for any $a \in \text{ready}(r)$, if $(r,q_1) \in \pi_1$ then $(\lambda(r,a),\delta_1(q_1,a)) \in \pi_1$ by Definition 2; moreover, if $(q_1,q_2) \in \pi$ then $(\delta_1(q_1,a),\delta_2(q_2,a)) \in \pi$ by Definition 4. As a result, we have: $(\lambda(r,a),\delta_2(q_2,a)) \in \pi_2$.

($\Leftarrow$) Suppose that $[S_1] \subseteq [S_2]$. Define $\pi$ such that $(q_1^0,q_2^0) \in \pi$ and for all $q_1,q_2 \in \pi$, if $\delta_1(q_1,a)$ and $\delta_2(q_2,a)$ are defined then $(\delta_1(q_1,a),\delta_2(q_2,a)) \in \pi$. We prove that $S_1 \subseteq S_2$ thanks to $\pi$.

Observe first that if $\delta_1(q_1,a)$ is defined then $\delta_2(q_2,a)$ is also defined; this is a direct consequence to the fact that when $\delta_1(q_1,a)$ is defined, the transition can be included in some models which are also models of $S_2$ and thus $\delta_2(q_2,a)$ is defined.

- for all $X \in \text{Acc}_1(q_1)$, there exists some $M \models S_1$ such that $(r,q_1) \in \pi_1$ and $\text{ready}(r) = X$. As $[S_1] \subseteq [S_2]$, $M$ is also a model of $S_2$ and necessarily $\text{ready}(r) \in \text{Acc}_2(q_2)$. As a result, $\text{Acc}_1(q_1) \subseteq \text{Acc}_2(q_2)$;
- suppose that $q_1 \in F_1$ and consider $M \models S_1$ such that $(r,q_1) \in \pi_1$ and $r \in G$. As $[S_1] \subseteq [S_2]$, $M$ is also a model of $S_2$ and necessarily $q_2 \in F_2$. As a result, $q_1 \in F_1$ implies $q_2 \in F_2$;
- by definition of $\pi$, for any $a \in \text{ready}(q_1)$, we have $(\delta_1(q_1,a),\delta_2(q_2,a))$.

As a result, according to Definition 4 we have $S_1 \subseteq S_2$. \hfill $\Box$

Theorem 2 holds provided the marked acceptance specifications are determinis-
tic. If nondeterminism is allowed, refinement becomes correct but not fully abstract (the implication from right to left in Theorem 2 is not true in general). This is discussed for unmarked modal specifications in [22]; their counterexample can be adapted to our context by first embedding modal specifications into acceptance specifications following the approach developed in [20] and by then marking every state.

4 Product of marked acceptance specifications

The construction of complex systems can be achieved by product of components. In a specification theory, this requires to lift the product operator from implementations to specifications; the product of two specifications then characterizes the behavior realized by the product of any implementations.

In our context, a difficulty is induced by the fact that reachability is not pre-

This leads us to first consider the following problem: given two MAS, can they be implemented concurrently i.e., such that it will be possible to reach a marked state from any state of the product of any model of the first specification with any model of the second one?
4.1 Compatible reachability

We solve the problem by first considering the case of deadlock-free products in Section 4.1.1 and then the case of livelock-free products in Section 4.1.2. This leads us to define a compatibility criterion on MAS which is a prerequisite for the product of MAS.

4.1.1 Deadlock-free specifications

In this section, we propose a criterion allowing to check if two specifications $S_1$ and $S_2$ have respective models $M_1$ and $M_2$ such that $M_1 \times M_2$ has a deadlock. We first define compatible acceptance sets:

Definition 5 (Compatible acceptance sets) Two acceptance sets $A_1$ and $A_2$ are said to be compatible, denoted $\text{Compat}(A_1, A_2)$, if and only if for all $X_1 \in A_1$ and $X_2 \in A_2$, $X_1 \cap X_2 \neq \emptyset$.

We then identify deadlock-free pairs of states, that are pairs of states of two MAS $S_1$ and $S_2$ to be composed from which no deadlock may be generated in the product of any two respective implementations:

Definition 6 (Deadlock-free pair of states) Given two MAS $S_1$ and $S_2$ and two states $q_1$ of $S_1$ and $q_2$ of $S_2$, the pair $(q_1, q_2)$ is deadlock-free, denoted $\text{DeadFree}(q_1, q_2)$, if $\text{Acc}(q_1) = \text{Acc}(q_2) = \emptyset$ or $\text{Compat}(\text{Acc}(q_1), \text{Acc}(q_2))$.

Example 4 Consider the MAS $S_1$ and $S_2$ in Figure 6(b) and Figure 6(d), the pair formed by their initial state is not deadlock-free as $\text{Acc}(0) \neq \emptyset$, $\text{Acc}(0') \neq \emptyset$ and $\neg \text{Compat}(\text{Acc}(0), \text{Acc}(0'))$: $\{b\} \in \text{Acc}(0)$, $\{a\} \in \text{Acc}(0')$ and $\{b\} \cap \{a\} = \emptyset$.

Definition 7 (Deadlock-free specifications) Two MAS $S_1$ and $S_2$ are deadlock-free if all the reachable pairs of states in $\text{Un}(S_1) \times \text{Un}(S_2)$ are deadlock-free.

Theorem 3 Two MAS $S_1$ and $S_2$ are deadlock-free if and only if for any $M_1 \models S_1$ and $M_2 \models S_2$, $M_1 \times M_2$ is deadlock-free.

Proof ($\Rightarrow$) Suppose that $(r_1, r_2)$ is a deadlock in $M_1 \times M_2$. Then $(r_1, r_2)$ is not marked and $\text{ready}((r_1, r_2)) = \emptyset$. Now $\text{ready}((r_1, r_2)) = \text{ready}(r_1) \cap \text{ready}(r_2)$ and moreover, $(r_1, q_1) \in \pi_1$ and $(r_2, q_2) \in \pi_2$ implies $\text{ready}(r_1) \in \text{Acc}(q_1)$ and
In consequence, we have \( \neg \text{DeadFree}(q_1, q_2) \). Moreover, \((r_1, r_2)\) is not marked so \((q_1, q_2)\) is not marked and \(\emptyset \notin \text{Acc}(q_1)\) and \(\emptyset \notin \text{Acc}(q_2)\). In consequence, we have \( \neg \text{DeadFree}(q_1, q_2) \) and \(S_1\) and \(S_2\) are not deadlock-free.

\((=)\) Suppose that \(S_1\) and \(S_2\) are not deadlock-free: there exists \(q_1\) and \(q_2\) such that \(\neg \text{DeadFree}(q_1, q_2)\). Then there exists \(X_1 \in \text{Acc}(q_1)\) and \(X_2 \in \text{Acc}(q_2)\) which verify \(X_1 \cap X_2 = \emptyset\). For any \(M_1 \models S_1\) and \(M_2 \models S_2\) with \((r_1, q_1) \in \pi_1\) and \((r_2, q_2) \in \pi_2\) such that \(\text{ready}(r_1) = X_1\) and \(\text{ready}(r_2) = X_2\), we have \(\text{ready}((r_1, r_2)) = X_1 \cap X_2 = \emptyset\) in \(M_1 \times M_2\). Moreover, \(\text{Acc}(q_1) \neq \{\emptyset\}\) (or \(\text{Acc}(q_2) \neq \{\emptyset\}\)), so there exists a model of \(S_1\) (resp. \(S_2\)) such that a state \(r\) implementing \(q_1\) (resp. \(q_2\)) is not marked and has at least one transition leading to another marked state, so \((r_1, r_2)\) is not marked. As a result, \((r_1, r_2)\) is a deadlock and \(M_1 \times M_2\) is not deadlock-free.

\(\square\)

4.1.2 Livelock-free specifications

In this section, we propose a criterion allowing to check if two specifications \(S_1\) and \(S_2\) have respective models \(M_1\) and \(M_2\) such that \(M_1 \times M_2\) has a livelock. This criterion is based on the identification of cycles shared between \(S_1\) and \(S_2\) together with a typing on transitions leaving these cycles. We then check that it is always possible to leave a cycle, no matter what implementation choices are made.

Before considering the common cycles, a first step consists in unfolding \(S_1\) and \(S_2\) so as possible synchronizations become unambiguous.

**Unfolding.** Given two specifications \(S_1\) and \(S_2\), we define the partners of a state \(q_1\) as \(Q_2(q_1) = \{q_2 \mid (q_1, q_2)\ is\ reachable\ in\ \text{Un}(S_1) \times \text{Un}(S_2)\}\); the set \(Q_1(q_2)\) is defined symmetrically. As a shorthand, if we know that a state \(q_1\) has exactly one partner, we will also use \(Q_2(q_1)\) to denote this partner.

We now show that, if some states of \(S_2\) have several partners, it is possible to transform it so that each of its states has at most one partner, while preserving the set of models of the specification.

**Definition 8 (Unfolding)** Given two MAS \(S_1\) and \(S_2\), the unfolding of \(S_2\) in relation to \(S_1\) is the specification \((Q_1 \cup \{q^?\}) \times Q_2, (q^0_1, q^0_2), \delta_u, \text{Acc}_u, (Q_1 \cup \{q^?\}) \times F_2\) where:

- \(q^?\) is a fresh state (\(q^?_1\) denotes a state in \(Q_1 \cup \{q^?\}\));
- \(\delta_u((q^?_1, q_2), a)\) is defined if and only if \(\delta_2(q_2, a)\) is defined and then:
  \[\delta_u((q^?_1, q_2), a) = \begin{cases} \delta_1(q_1, a), & \text{if } \delta_1(q_1, a) \text{ is defined} \\ (q^?, \delta_2(q_2, a)) & \text{otherwise} \end{cases}\]
- \(\text{Acc}_u((q^?_1, q_2)) = \text{Acc}_2(q_2)\).

**Example 5** Consider the MAS \(S_1\) and \(S_2\) in Figures 7(a) and 7(b) Some states of \(S_2\) have several partners: \(Q_1(0^{'}) = \{0, 2\}\) and \(Q_1(2^{'}) = \{3, 5\}\). The unfolding of \(S_2\) is shown in Figure 7(c) All its states have at most one partner: \(Q_2((0, 0^{'}) = \{0\}, Q_2((3, 2^{'}) = \{3\}, Q_2((?, 1^{'}) = \emptyset, \ldots\)
Acc(0) = \{\{a\}, \{a, d\}\}
Acc(1) = \{\{a\}, \{a, c\}\}
Acc(2) = \{\{b\}, \{b, c\}, \{b, d\}, \{b, c, d\}\}
Acc(3) = Acc(4) = Acc(5) = \{\}\n
Acc(0') = \{\{a, b, c, d\}\}
Acc(1') = \{\{a, c\}, \{a, d\}, \{a, c, d\}\}
Acc(2') = Acc(3') = \{\}\n
(c) Unfolding of S_2

Fig. 7: Example of unfolding
Lemma 1 Given two MAS $S_1$ and $S_2$ and $S_u$ the unfolding of $S_2$ in relation to $S_1$, $S_u \equiv S_2$.

Proof (⇒) Let $M$ be a model of $S_u$. Let $\pi_u$ be the simulation relation between the states of $M$ and the states of $S_u$ and let $\pi_2$ be the simulation relation such that $(r,q_2) \in \pi_2$ if and only if there exists a $q'_2$ such that $(r,(q'_2,q_2)) \in \pi_u$. $(r_0, q'_2) \in \pi_2$ and for any $(r,q_2) \in \pi_2$:
- $\text{ready}(r) \in \text{Acc}_2(q_2)$ as $\text{ready}(r) \in \text{Acc}_u((q'_2,q_2)) = \text{Acc}_2(q_2)$;
- if $r \in G$, $q_2 \in F_2$ as $(q'_2,q_2) \in (Q_1 \cup \{q'_1\}) \times F_2$;
- for any $a \in \text{ready}(r)$, $(\lambda(r,a),\delta_2(q_2,a)) \in \pi_2$ as $(\lambda(r,a),\delta_u((q'_2,q_2),a)) \in \pi_u$.
Thus $M$ is a model of $S_u$.

(⇐) Let $M$ be a model of $S_2$. Let $\pi_2$ be the simulation relation between the states of $M$ and the states of $S_2$ and let $\pi_u$ be the simulation relation such that $(r,(q'_2,q_2)) \in \pi_u$ if and only if $(r,q_2) \in \pi_2$ and $(q'_2,q_2)$ is reachable in $S_u$. $(r_0,(q'_2,q_2)) \in \pi_u$ and for any $(r,(q'_2,q_2)) \in \pi_2$:
- $\text{ready}(r) \in \text{Acc}_u((q'_2,q_2)) = \text{Acc}_u((q'_2,q_2))$;
- if $r \in G$, $(q'_2,q_2) \in (Q_1 \cup \{q'_1\}) \times F_2$ as $q_2 \in F_2$;
- for any $a \in \text{ready}(r)$, $(\lambda(r,a),\delta_u((q'_2,q_2),a)) \in \pi_u$ as $(\lambda(r,a),\delta_2(q_2,a)) \in \pi_2$.
Thus $M$ is a model of $S_u$. □

Lemma 2 Given two MAS $S_1$ and $S_2$ and $S_u$ the unfolding of $S_2$ in relation to $S_1$, for any $(q_1,(q'_1,q_2))$ reachable in $\text{Un}(S_1) \times \text{Un}(S_2)$, $q_1 = q'_1$.

Proof If a state is reachable in $\text{Un}(S_1) \times \text{Un}(S_2)$, there is a path from the initial state to it. By induction on this path:
- if it is empty, we are in the initial state $(q'_0,(q'_0,q'_0))$;
- otherwise, we are in a state $(q_1,(q_1,q_3))$ and there is a transition by an action $a$ to another state $(\delta_1(q_1,a),\delta_u((q_1,q_3),a))$.
As $\delta_1(q_1,a)$ is defined, $\delta_u((q_1,q_3),a) = (\delta_1(q_1,a),\delta_2(q_2,a))$, so the destination state is $(\delta_1(q_1,a),\delta_2(q_2,a))$.

Lemma 3 Given two MAS $S_1$ and $S_2$ and $S_u$ the unfolding of $S_2$ in relation to $S_1$, for any state $q_u$ of $S_u$, $|Q_1(q_u)| \leq 1$.

Proof Let suppose that $|Q_1(q_u)| > 1$. Then, there exists at least two different states $q_1$ and $q'_1$ such that $(q_1,q_u)$ and $(q'_1,q_u)$ are reachable in $\text{Un}(S_1) \times \text{Un}(S_2)$. By Definition 8 there exists some $q'_1 \in Q_1 \cup \{q_1\}$ and $q_2 \in Q_2$ such that $q_u = (q'_1,q_2)$.
By Lemma 2 $q_1 = q'_1$ and $q'_1 = q_1$, so $q_1 = q'_1$. But we know by hypothesis that they are different, so $|Q_1(q_u)| \leq 1$. □

Two MAS $S_1$ and $S_2$ have single partners if and only if for all $q_1 \in Q_1$, we have $|Q_2(q_1)| \leq 1$ and for all $q_2 \in Q_2$, we also have $|Q_1(q_2)| \leq 1$.

Theorem 4 Given two MAS $S_1$ and $S_2$, there exists some MAS $S'_1$ and $S'_2$, called unfoldings of $S_1$ and $S_2$, with single partners and which are equivalent to $S_1$ and $S_2$.

Proof Let $S'_1$ be the unfolding of $S_1$ in relation to $S_2$ and $S'_2$ the unfolding of $S_2$ in relation to $S'_1$.
By Lemma 1 we know that $S'_1$ has the same models as $S_1$ and $S'_2$ as $S_2$. 

By Lemma 3, we know that for any \( q'_1 \) in \( S'_1 \), \(|Q_2(q'_1)| \leq 1\) and that for any \( q'_2 \) in \( S'_2 \), \(|Q'_2(q'_2)| \leq 1\). Remains to prove that \(|Q'_2(q'_1)| \leq 1\).

Let \( q'_1 \) be a state of \( S'_1 \). If \(|Q_2(q'_1)| = 0\), then \(|Q'_2(q'_1)| = 0\) as \( S_2 \) and \( S'_2 \) have the same models. Otherwise, there exists a \( q_2 \) such that \( Q_2(q'_1) = \{ q_2 \} \). There exists then \( n \) states (with \( n > 0 \)) \( q'_2 \), of the form \( (q'_1, q_2) \). But each \( q'_2 \) is in relation with at most one state \( (q'_1, \ast) \) of \( S'_1 \), as \(|Q'_2(q'_2)| \leq 1\), and all these \( q'_i \) are different (as the \( q'_2 \) are different). So there is at most one \( q'_2 \) in relation with \( q'_1 \) and thus \(|Q'_2(q'_1)| \leq 1\). \( \square \)

Cycles. In order to detect livelocks, we need to study the cycles that may be present in the models of a specification. Intuitively, a cycle is characterized by its states and the transitions between them.

**Definition 9 (Cycle)** Given a MAS \( S \), the partial map \( C : Q \rightarrow 2^Σ \) represents a cycle in \( S \) if and only if for any \( q \in \text{dom}(C) \):
- \( C(q) \neq \emptyset \);
- \( \exists X \in \text{Acc}(q), C(q) \subseteq X \);
- \( \text{dom}(C) \subseteq \text{post}(q) \);
- \( \forall a \in C(q), \lambda(q, a) \in \text{dom}(C) \).

**Definition 10 (Cycle implementation)** A model \( M \) of a specification \( S \) implements a cycle \( C \) if and only if there exists a set \( R \) of states of \( M \) such that:
- each \( q \in \text{dom}(C) \) is implemented by at least one state of \( R \);
- for each \( r \in R \) and for each \( q \) such that \((r, q) \in \pi\):
  - \( q \in \text{dom}(C) \);
  - \( C(q) \subseteq \text{ready}(r) \);
  - \( \forall a \in C(q), \lambda(r, a) \in R \);
  - \( \forall a \in \text{ready}(r) \setminus C(q), \lambda(r, a) \not\in R \).

**Example 6** For example, there are three possible cycles in the MAS \( S \) in Figure 2:
- \( \{1 \rightarrow \{b\}, 2 \rightarrow \{c\} \} \), \( \{1 \rightarrow \{b\}, 2 \rightarrow \{d\} \} \) and \( \{1 \rightarrow \{b\}, 2 \rightarrow \{c, d\} \} \). The model of \( S \) in Figure 4 implements this last cycle, with the set \( R = \{1', 2', 3'\} \).

A cycle is said to be implementable if there exists a model \( M \) of \( S \) implementing the cycle.

**Algorithm 2** defines the operation \( \text{Cycle}_{\text{imp}} - \text{rec} \) which computes the set of cycles in a specification passing by a given state. However, some of these cycles may not be implementable. Consider for example the MAS depicted in Figure 8, there is a cycle \( C = \{0 \rightarrow \{a\} \} \) but it is not implementable; indeed, any model of the specification must eventually realize the transition by \( b \) and then it can’t simultaneously realize \( a \) to make a cycle. Intuitively, a cycle is only implementable if including it still allows to reach a marked state. This means that either the cycle contains a marked state or it is possible to realize a transition that will leave the cycle, in addition to the transitions needed to realize it.

**Definition 11 (Implementable cycle)** Given a state \( q \) of a MAS \( S \), the set of implementable cycles of \( S \) passing by \( q \), \( \text{Cycle}_{\text{imp}}(S, q) \), is \( \{ C \in \text{Cycle}_{\text{imp}} - \text{rec}(S, q, \emptyset) \mid \text{dom}(C) \cap F \neq \emptyset \lor \exists q_C \in \text{dom}(C), \exists X \in \text{Acc}(q_C), C(q_C) \subseteq X \} \).

The set of implementable cycles of a MAS \( S \) is \( \text{Cycle}_{\text{imp}}(S) = \bigcup_{q \in Q} \text{Cycle}_{\text{imp}}(S, q) \).
Algorithm 2 Cycle\(_{rec} (S; MAS, q; State, cycle; Cycle): Set Cycle

1: if \( q \in \text{dom}(\text{cycle}) \) then return \{\text{cycle}\} end if
2: \( \text{res} \leftarrow \emptyset \)
3: for all \( A \in \text{Acc}(q) \) do
4: \( \text{cycle}_{acc} \leftarrow \{a \mid a \in A \land q \in \text{post}^*(\delta(q, a))\} \)
5: for all \( C \in 2^{\text{cycle}_{acc} \setminus \{\emptyset\}} \) do
6: \( \text{current} \leftarrow \{\text{cycle}\} \)
7: for all \( a \in C \) do
8: \( \text{current} \leftarrow \bigcup \text{cycle} \in \text{current} \text{Cycle}_{rec}(S, \delta(q, a), \text{cycle} \cup \{q \mapsto \text{current}\}) \)
9: end for
10: \( \text{res} \leftarrow \text{res} \cup \text{current} \)
11: end for
12: end for
13: return \( \text{res} \)

\[
\begin{align*}
&\text{Acc}(q_0) = \{\{a\}, \{b\}\} \\
&\text{Acc}(q_1) = \{\emptyset\}
\end{align*}
\]

Fig. 8: A MAS over \{a, b\} with no implementable cycle

The previous definition based on Algorithm 2 allows to characterize when a cycle can be implemented:

**Theorem 5** Given a MAS \( S \), a model \( M \) of \( S \) implements a cycle \( C \) if and only if \( C \in \text{Cycle}_{rec}(S) \).

**Proof** (⇒) Let \( C \) be a cycle in \( S \) and \( M \) a model of \( S \) implementing \( C \), with \( R \) the set of states of \( M \) implementing the states of \( C \). Let \( r \) be an element of \( R \) and \( q \) a state it implements. By definition, \( \text{Cycle}_{rec}(S) \) contains the result of \( \text{Cycle}_{rec}(S, q) \), which calls \( \text{Cycle}_{rec}(S, \delta(q, a), \text{cycle} \cup \{q \mapsto C\}) \). For an iteration of the loop at line 5, the variable \( C \) will take the value of \( C(q) \) and it will be inserted in the generated cycle. The algorithm will then be called recursively on the successors of \( q \) in the cycle, until \( q \) is reached again, thus obtaining \( C \).

(⇐) Let \( C \) be a cycle returned by \( \text{Cycle}_{rec}(S) \). It is possible to build an automaton \( M \) implementing the states and transitions of \( C \). The problem is to make sure that this automaton is terminating, ie. that it is possible to reach a marked state from any implementation of a state of \( \text{dom}(C) \). By [Definition 11], we know that \( \text{dom}(C) \cap F \neq \emptyset \) or \( \exists q \in \text{dom}(C), \exists X \in \text{Acc}(q), C(q) \subset X \). In the first case, there is a marked state in the loop, thus \( M \) is terminating. In the second case, we know that there is a state \( q \), implemented in \( M \) by a state \( r \), from which there is a transition by an action \( a \) which leaves the cycle, that is, \( \lambda(r, a) \) is not in the set of states implementing \( C \). There is thus no constraint on the transitions from \( \lambda(r, a) \) and it will be possible to reach a marked state from it (provided that \( S \) is well-formed). So \( M \) is terminating and in consequence is a model of \( S \).

**Livelock-freeness.** Given two specifications with single partners, we can now examine their cycles in order to check if there is a possible livelock in some of their models. To do so, we distinguish two kinds of transitions: those, denoted \( A \), which
are always realized when the cycle is implemented and those, denoted \( \mathcal{O} \), which may (or may not) be realized when the cycle is implemented. These two values are computed, for a given cycle, by Algorithm 3.

**Algorithm 3** critical \((S: \text{MAS}, \text{cycle: Cycle}) \times \text{Map State (Set (Set Action))}\)

1: \( \mathcal{A}: \text{Map State (Set (Set Action))} = \emptyset, \mathcal{O}: \text{Map State (Set (Set Action))} = \emptyset \)
2: for all \((q, A) \in \text{cycle}\) do
3: if \( A \not\in \text{Acc}(q) \) then
4: \( \mathcal{A}[q] \leftarrow \{X | A \times X \in \text{Acc}(q) \land A \subset X\} \)
5: else if \( \exists X \in \text{Acc}(q), A \subset X \) then
6: \( \mathcal{O}[q] \leftarrow \{X | A \times X \in \text{Acc}(q) \land A \subset X\} \)
7: end if
8: end for
9: return \( \mathcal{A}, \mathcal{O} \)

**Definition 12** Given two MAS \( S_1 \) and \( S_2 \) with single partners and a cycle \( C_1 \) in \( S_1 \) such that all its states have a partner, \( C_1 \) is livelock-free in relation to \( S_2 \), denoted \( \text{LiveFree}(C_1, S_2) \), if and only if, when the cycle \( C_2 = \{Q_2(q) \mapsto C_1(q) \mid q \in \text{dom}(C_1)\} \) is in \( \text{Cycle}_{\text{in}}(S_2) \):

1. \( \mathcal{A}_{C_1} \neq \emptyset, \mathcal{A}_{C_2} \neq \emptyset \) and there exists \( q'_1 \in \text{dom}(\mathcal{A}_{C_1}) \) such that \( Q_2(q'_1) \in \text{dom}(\mathcal{A}_{C_2}) \) and \( \text{Compat}(\mathcal{A}_{C_1}(q'_1), \mathcal{A}_{C_2}(Q_2(q'_1))) \) or
2. \( \mathcal{A}_{C_1} \neq \emptyset, \mathcal{A}_{C_2} = \emptyset \), \( \text{dom}(C_2) \cap F_2 = \emptyset \) and \( \forall q'_2 \in \text{dom}(\mathcal{O}_{C_2}), Q_1(q'_2) \in \text{dom}(\mathcal{A}_{C_1}) \) and \( \text{Compat}(\mathcal{A}_{C_1}(Q_1(q'_2)), \mathcal{O}_{C_2}(q'_2)) \) or
3. \( \mathcal{A}_{C_1} = \emptyset, \mathcal{A}_{C_2} \neq \emptyset \), \( \text{dom}(C_1) \cap F_1 = \emptyset \) and \( \forall q'_1 \in \text{dom}(\mathcal{O}_{C_1}), Q_2(q'_1) \in \text{dom}(\mathcal{A}_{C_2}) \) and \( \text{Compat}(\mathcal{O}_{C_1}(q'_1), \mathcal{A}_{C_2}(Q_2(q'_1))) \).

**Definition 13** (Livelock-free specifications) Two MAS \( S_1 \) and \( S_2 \) with single partners are **livelock-free** if all the implementable cycles of \( S_1 \) are livelock-free in relation to \( S_2 \).

Note that this definition only tests the implementable cycles of \( S_1 \). It is not necessary to do the symmetrical test (checking that the implementable cycles of \( S_2 \) verify \( \text{LiveFree} \)) because we only compare the cycle of \( S_1 \) with the same cycle in \( S_2 \) and the three tests of **Definition 12** are symmetric.

The previous definition offers a necessary and sufficient condition to identify MAS which can have two respective models whose product has a livelock:

**Theorem 6** Two MAS \( S_1 \) and \( S_2 \) with single partners are livelock-free if and only if for any \( M_1 \models S_1 \) and \( M_2 \models S_2 \), \( M_1 \times M_2 \) is livelock-free.

**Proof** \((\Rightarrow)\) Assume that there exists \( M_1 \models S_1, M_2 \models S_2 \) such that \( M_1 \times M_2 \) has a livelock, that is, there exists \( (r_1, r_2) \) such that \( \text{Loop}((r_1, r_2)) \neq \emptyset \), \( \text{Loop}((r_1, r_2)) \cap G = \emptyset \) and there is no transition \((r', a, r'')\) such that \( r' \in \text{Loop}((r_1, r_2)) \) and \( r'' \not\in \text{Loop}((r_1, r_2)) \).

- If there exists a cycle \( C_1 \in \text{Cycle}_{\text{in}}(S_1) \) which is implemented in \( M_1 \) by the states of \( \text{Loop}(r_1) \) and \( C_2 = \{Q_2(q) \mapsto C_1(q) \mid q \in \text{dom}(C_1)\} \) is implemented in \( M_2 \) by the states of \( \text{Loop}(r_2) \):
Theorem 7

Given two MAS unfoldings are livelock-free.

Definition 14 (Compatible reachability)

We now propose a criterion allowing to check if two specifications $S_1$ and $S_2$ are compatible, i.e., they have different actions or different source states. If in both models, some of these transitions are in $A$, (they have to be present whenever the cycle is implemented), the test 1 of Definition 12 will detect that they are not compatible. If there are some transitions in $A$, but none in $C$, test 2 will detect that $M_2$ may implement a transition that will not be covered by the transitions in $A$. Test 3 handles the symmetrical case. Finally, if there are transitions neither in $A$, nor $C$, it is always possible to generate a livelock and all three tests fail.

Otherwise, multiple cycles are implemented simultaneously in the model by unfolding them or two slightly different cycles are implemented in $M_1$ and $M_2$, and then there will also be a livelock in the models which implement only one of the cycles, which brings us back to the first case.

$(\epsilon)$ Assume that $S_1$ and $S_2$ are not livelock-free. Then, there exists a cycle $C_1$ such that $\neg \text{LiveFree}(C_1, S_2)$. Then, the three conditions of Definition 13 are all false.

- If $A_{C_1} \neq \emptyset$ and $A_{C_2} \neq \emptyset$, then for any state $q_1' \in \text{dom}(A_{C_1})$ in $S_1$, we have $\neg \text{Compat}(A_{C_1}(q_1'), A_{C_2}(Q_2(q_1')))$. So there exists a model $M_1$ of $S_1$ implementing $C_1$ and a model $M_2$ of $S_2$ implementing $C_2$ such that there is no transition leaving the cycle in their product, hence there is a livelock in $M_1 \times M_2$.

- If $A_{C_1} \neq \emptyset$ and $A_{C_2} = \emptyset$, there exists a state $q_2' \in \text{dom}(O_{C_2})$ such that we have $\neg \text{Compat}(A_{C_1}(Q_1(q_2')), O_{C_2}(q_2'))$. So for any model $M_1$ of $S_1$ implementing $C_1$, its product with a model $M_2$ of $S_2$ implementing $C_2$ for which the only transition leaving the cycle is from an implementation of $q_2'$ will have a livelock.

- If $A_{C_1} = \emptyset$ and $A_{C_2} \neq \emptyset$, we are in the case symmetric to the previous one.

- If $A_{C_1} = \emptyset$ and $A_{C_2} = \emptyset$, either one of the specifications has no transitions leaving the cycle ($O_{C_1} = \emptyset$ too), so there are some models such that their product has a livelock, or both $O_{C_1}$ and $O_{C_2}$ are not empty, and then there exists an $M_1 \models S_1$ implementing $C_1$ such that the only transition(s) leaving the cycle is (are) from a state $r_1$ and an $M_2 \models S_2$ implementing $C_2$ such that the only transition(s) leaving the cycle is (are) from a state $r_2$ which is never paired with $r_1$ in $M_1 \times M_2$, hence there is a livelock in $M_1 \times M_2$. □

4.1.3 Specifications with compatible reachability

By combining the tests for deadlock-free and livelock-free specifications, we can now propose a criterion allowing to check if two specifications $S_1$ and $S_2$ have respective models $M_1$ and $M_2$ such that $M_1 \times M_2$ is not terminating:

Definition 14 (Compatible reachability) Two MAS $S_1$ and $S_2$ have a compatible reachability, denoted $S_1 \sim_\mathcal{T} S_2$, if and only if they are deadlock-free and their unfoldings are livelock-free.

Theorem 7 Given two MAS $S_1$ and $S_2$, $S_1 \sim_\mathcal{T} S_2$ if and only if for any $M_1 \models S_1$ and $M_2 \models S_2$, $M_1 \times M_2$ is terminating.
Proof By definition, $S_1 \sim_T S_2$ if and only if $S_1$ and $S_2$ are deadlock-free and livelock-free. By Theorems 3, 4 and 6, this is equivalent to: for any $M_1 \models S_1$ and $M_2 \models S_2$, $M_1 \times M_2$ is deadlock-free and livelock-free, that is, $M_1 \times M_2$ is terminating. □

4.2 Product definition

Given two MAS with compatible reachability, we can now compute their product which generalizes the product of implementations:

Definition 15 (Product) Given two MAS $S_1$ and $S_2$ with compatible reachability, their product $S_1 \otimes S_2$ is the normal form of the MAS $(Q_1 \times Q_2, (q_1^0, q_2^0), \delta, \text{Acc}, F_1 \times F_2)$ with $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$ when both $\delta_1(q_1, a)$ and $\delta_2(q_2, a)$ are defined and $\text{Acc}(q_1, q_2) = \{A_1 \cap A_2 | A_1 \in \text{Acc}(q_1) \land A_2 \in \text{Acc}(q_2)\}$.

Theorem 8 Given two MAS $S_1$ and $S_2$ with compatible reachability, for any $M_1 \models S_1$ and $M_2 \models S_2$, $M_1 \times M_2 \models S_1 \otimes S_2$.

Proof By Theorem 7 $M_1 \times M_2$ is terminating.

Let $\pi_i$ be the simulation relation of $M_i \models S_i$ for $i \in \{1, 2\}$ and $\pi$ the simulation relation such that $((r_1, r_2), (q_1, q_2)) \in \pi$ if and only if $(r_1, r_2)$ is reachable in $M_1 \times M_2$, $(r_1, q_1) \in \pi_1$ and $(r_2, q_2) \in \pi_2$. For any $((r_1, r_2), (q_1, q_2)) \in \pi$:

- $\text{ready}(r_1, r_2) = \text{ready}(r_1) \cap \text{ready}(r_2) \in \text{Acc}(q_1, q_2)$ by definition of the acceptance set of the product;
- $(r_1, r_2) \in G_1 \times G_2$ implies that $(q_1, q_2) \in F_1 \times F_2$ as $r_1 \in G_1$ implies that $q_1 \in F_1$ and $r_2 \in G_2$ that $q_2 \in F_2$;
- for any $a$ and $a'$ such that $\lambda((r_1, r_2), a) = a'$, $(r_1, r_2, a) \in \pi$ is trivial as $\lambda((r_1, r_2), a) = (\lambda_1(r_1, a), \lambda_2(r_2, a))$. □

Moreover, $S_1 \otimes S_2$ gives the most precise characterization of the behavior of the product of any models $M_1$ of $S_1$ and $M_2$ of $S_2$:

Proposition 1 Given two MAS $S_1$ and $S_2$, if $S_1 \sim_T S_2$ and if there exists a MAS $S$ such that for any $M_1 \models S_1$ and $M_2 \models S_2$ we have $M_1 \times M_2 \models S$ then $S_1 \otimes S_2 \subseteq S$.

Proof By contradiction assume that for any $M_1 \models S_1$ and $M_2 \models S_2$ we have $M_1 \times M_2 \models S$ but $S_1 \otimes S_2 \not\subseteq S$. Then, there exists an execution common to $\text{Un}(S_1 \otimes S_2)$ and $\text{Un}(S)$ leading to some state $(q_1, q_2)$ in $S_1 \otimes S_2$ and $q$ in $S$ such that $\text{Acc}(q_1, q_2) \not\subseteq \text{Acc}(q)$ that is, there exist $A_1 \in \text{Acc}(q_1)$ and $A_2 \in \text{Acc}(q_2)$ such that $A_1 \cap A_2 \not\subseteq \text{Acc}(q)$. Consider now $M_i$ such that $(r_i, q_i) \in \pi_i$ and $\text{ready}(r_i) = A_i$, for $i = 1, 2$, the product $M_1 \times M_2$ cannot be a model of $S$ as $\text{ready}(r_1, r_2) = A_1 \cap A_2 \not\subseteq \text{Acc}(q)$ which contradicts the assumption made at the beginning of the proof. □

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 specifications can be designed concurrently. This is respected for compatible reachability and product:
Proposition 2 For all MAS $S_1$, $S'_1$ and $S_2$, if $S_1 \sim_{\mathcal{T}} S_2$ and $S'_1 \leq S_1$ then $S'_1 \sim_{\mathcal{T}} S_2$ and $S'_1 \otimes S_2 \leq S_1 \otimes S_2$.

Proof Let $M_1$ and $M_2$ be models of $S'_1$ and $S_2$. As $S'_1 \leq S_1$, by Theorem 2 $M_1$ is also a model of $S_1$. Moreover, the product $M_1 \times M_2$ is terminating as $S_1 \sim_{\mathcal{T}} S_2$, by Theorem 7. As a result, by Theorem 7 $S'_1 \sim_{\mathcal{T}} S_2$.

Let $\pi_1$ be the simulation relation of $S'_1 \leq S_1$ and $\pi$ the simulation relation such that $((q_1', q_2), (q_1, q_2)) \in \pi$ if and only if $(q_1', q_2)$ is reachable in $S'_1 \otimes S_2$ and $(q_1, q_2) \in \pi$. For any $((q_1', q_2), (q_1, q_2)) \in \pi$:

- Let $A$ be an element of $\text{Acc}((q_1, q_2))$. By definition of the acceptance set of the product, there exist $A'_1 \in \text{Acc}_1(q_1')$ and $A_2 \in \text{Acc}_2(q_2)$ such that $A = A'_1 \cap A_2$. As $S'_1 \leq S_1$, $A'_1 \in \text{Acc}_1(q_1)$ too, so $A = A'_1 \cap A_2 \in \text{Acc}((q_1, q_2))$, hence $\text{Acc}((q_1', q_2)) \subseteq \text{Acc}((q_1, q_2))$.
- $(q_1, q_2) \in F_1 \times F_2$ implies $(q_1, q_2) \in F_1 \times F_2$ as $q'_1 \in F'_1$ implies $q_1 \in F_1$ by definition of the refinement.
- For any $a$ and $q'$ such that $\delta((q_1', q_2), a) = q'$, $(q', \delta((q_1, q_2), a)) \in \pi$ is trivial as $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$ and $S'_1 \leq S_1$. \hfill \square

5 Conjunction of marked acceptance 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 defining a conjunction operator. Moreover in [15], the authors point out that, during the design cycle, a designer may be tempted to merge two specifications which share some similarities in order to use a same implementation for the two specifications. More formally, this corresponds to looking for a shared refinement of the specifications, if it exists.

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

Definition 16 (Conjunction) Given two MAS $S_1$ and $S_2$, the conjunction of $S_1$ and $S_2$, denoted $S_1 \land S_2$, is the normal form $\rho(S_1 \land S_2)$ of $S_1 \land S_2 = (Q_1 \times Q_2, (q_1^0, q_2^0), \delta, \text{Acc}, F_1 \times F_2)$ with $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$ when both $\delta_1(q_1, a)$ and $\delta_2(q_2, a)$ are defined, and $\text{Acc}((q_1, q_2)) = \text{Acc}_1(q_1) \cap \text{Acc}_2(q_2)$.

Considering the manipulations done to obtain the acceptance sets of $S_1 \& S_2$, the $F$, $\text{Acc}$-consistency and the $\delta$, $\text{Acc}$-consistency may not be respected in this specification. We thus explicitly impose a normalization step in order to have an iterative process as explained at the end of Section 2.2.

Theorem 9 Given two MAS $S_1$ and $S_2$: $[S_1 \land S_2] = [S_1] \land [S_2]$.

Proof Assume that $M \models S_i$ thanks to $\pi_i$ for $i = 1, 2$ and define $\pi$ such that $(r, (q_1, q_2)) \in \pi$ if and only if $(r, q_i) \in \pi_i$. We show that $\pi$ allows to state that $M \models S_1 \land S_2$:

- $\text{ready}(r) \in \text{Acc}_i(q_i)$ as $(r, q_i) \in \pi_i$ and thus $\text{ready}(r) \in \text{Acc}((q_1, q_2))$ by definition of $\land_i$. \hfill \square
be the specification of an available black-box component. The specification $S$ operator. More precisely, let components. In a specification theory, this approach can be promoted via a quotient

Efficient modular design can be achieved by enabling the reuse of preexisting components. In a specification theory, this approach can be promoted via a quotient guaranteeing termination in Section 6.4.

We first define an operation called pre-quotient. Given two MAS $S_1$ and $S_2$, it returns a MAS $S_1/S_2$ such that the product of any of its models with any model of $S_2$ will be an automaton which satisfies $S_1$ but does not guarantee the termination condition. Another operation, defined in Section 6.2 will then be used to define a quotient guaranteeing termination in Section 6.4.

**Definition 17 (Pre-quotient)** Given two MAS $S_1$ and $S_2$, their pre-quotient $S_1//S_2$ is the MAS $(Q_1 \times Q_2, (q_1^0, q_2^0), \delta, \text{Acc}, F)$ with:

- $r \in G$ implies that $(q_1, q_2) \in F_1 \times F_2$ as $r \in G$ implies that $q_i \in F_i$;
- for any $a$ and $r'$ such that $\lambda(r, a) = r'$, $(r', \delta((q_1, q_2), a)) \in \pi$ is trivial as we know that $(r', \delta_i(q_i, a)) \in \pi_i$ and $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$.

(\subseteq) Assume that $M \models S_1 \land S_2$ thanks to $\pi$ and define $\pi_i$ for $i = 1, 2$ such that $(r, q_i) \in \pi_i$ if and only if $(r, (q_1, q_2)) \in \pi$. We show that $\pi_i$ allows to state that $M \models S_i$:
- $\text{ready}(r) \in \text{Acc}_1(q_1) \land \text{Acc}_2(q_2)$ by definition of $\land$ and thus $\text{ready}(r) \in \text{Acc}_i(q_i)$;
- $r \in G$ implies that $q_i \in F_i$ as $r \in G$ implies that $(q_1, q_2) \in F_1 \times F_2$;
- for any $a$ and $r'$ such that $\lambda(r, a) = r'$, $(r', \delta_i(q_i, a)) \in \pi_i$ is trivial as we know that $(r', \delta((q_1, q_2), a)) \in \pi$ and $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$. □

**Proposition 3** For any MAS $S_1$, $S_2$ and $S$, the MAS $S_1 \land S_2$ is the greatest lower bound of $S_1$ and $S_2$ for the refinement relation: $S \leq S_1$ and $S \leq S_2$ if and only if $S \leq S_1 \land S_2$.

**Proof** This proposition is a consequence of Theorem 9 if $S \leq S_i$ for $i \in \{1, 2\}$ then, by Theorem 2 $[S] \subseteq [S_i]$. As a result, we have: $[S] \subseteq [S_1] \cap [S_2]$. By Theorem 9 this is equivalent to: $[S] \subseteq [S_1] \land [S_2]$. Last, we deduce from Theorem 2 that $S \leq S_1 \land S_2$. □

6 Quotient of marked acceptance specifications

Efficient modular design can be achieved by enabling the reuse of preexisting components. In a specification theory, this approach can be promoted via a quotient operator. More precisely, let $S_1$ be the specification of a target system and $S_2$ be the specification of an available black-box component. The specification $S_1/S_2$ characterizes all the components that, when composed with any model of $S_2$, conform with $S_1$. In other words, $S_1/S_2$ tells what remains to be implemented to realize $S_1$ while reusing a component doing $S_2$.

The quotient of specifications plays also a central role in contract-based design. In essence, a contract describes what a system should guarantee under some assumptions about its context of use. It can be modeled as a pair of specifications $(A, G)$ for, respectively, the assumptions and the guarantees. Satisfiability of a contract then corresponds to the satisfaction of the specification $G/A$ (see [7] for more explanations on contract satisfaction).

In this section, we study quotient of MAS hence enabling the incremental design of reachability properties.

6.1 Pre-quotient

We first define an operation called pre-quotient. Given two MAS $S_1$ and $S_2$, it returns a MAS $S_1/S_2$ such that the product of any of its models with any model of $S_2$ will be an automaton which satisfies $S_1$ but does not guarantee the termination condition. Another operation, defined in Section 6.2 will then be used to define a quotient guaranteeing termination in Section 6.4.

**Definition 17 (Pre-quotient)** Given two MAS $S_1$ and $S_2$, their pre-quotient $S_1//S_2$ is the MAS $(Q_1 \times Q_2, (q_1^0, q_2^0), \delta, \text{Acc}, F)$ with:
Theorem 10 (Correctness) Given two MAS $S_1$ and $S_2$ and an automaton $M \models S_1 \parallel S_2$, for any $M_2 \models S_2$ such that $M \times M_2 \models S_1$.

Proof Let $\pi_1$ and $\pi_2$ be the simulation relations of $M \models S_1 \parallel S_2$ and $M_2 \models S_2$. Let $\pi \subseteq ((R \times R) \times Q_1)$ be the simulation relation such that $((r, r_2), q_1) \in \pi$ if there exists a $q_2$ such that $(r_2, q_2) \in \pi_2$ and $(r, (q_1, q_2)) \in \pi_1$. For any $((r, r_2), q_1) \in \pi$: 
- $\text{ready}(r, r_2) \in \text{Acc}(q_1)$: by definition of the product of automata, $\text{ready}(r, r_2) = \text{ready}(r) \cap \text{ready}(r_2)$ and by definition of the acceptance set of the pre-quotient, this intersection is in the acceptance set of $q_1$.
- $(r, r_2) \in G_1 \times G_2$ implies $q_1 \in F_1$: by definition of the pre-quotient, if $r \in G_1$, then $q_1 \in F_1$.
- for any $a$, if $\lambda((r, r_2), a) = (r', r'_2)$, then $\delta_1(q_1, a)$ is defined and $((r', r'_2), \delta_1(q_1, a)) \in \pi$: $\lambda(r, a) = r'$, so $\delta_2(q_2, a) = r_2$ and, by definition of the pre-quotient, $q_1 = q'_1$ and $q_2 = q'_2$: $(r, (q_1, q_2)) \in \pi_1$ so $(r', (q'_1, q'_2)) \in \pi_1$, $(r_2, q_2) \in \pi_2$, so $(r'_2, q'_2) \in \pi_2$, hence $((r', r'_2), q'_1) \in \pi$. \hfill $\square$

In general, it is also expected that the specification returned by a quotient should also be complete, that is, it should characterize all the possible automata whose product with any model of $S_2$ is a model of $S_1$. However, this can lead to a very large specification as the quotient $S_1/S_2$ should then include all the transitions which are not fireable in $S_2$ (and thus removed in the product of the models). In this paper, we propose to return a compact quotient specification without unnecessary transitions regarding $S_2$, i.e. without the transitions that will always be cut by the product with a model of $S_2$. Then, completeness of a quotient $S_1/S_2$ amounts to guarantee that any automaton whose product with any model of $S_2$ is a model of $S_1$ is a model of $S_1/S_2$ after the removal of these useless transitions.

Definition 18 (Unnecessary transition) Given a MAS $S$ and an automaton $M$, $M$ has no unnecessary transition regarding $S$, denoted $M \sim_{\text{q}} S$, 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 $(r, q) \in \pi$:
- $\text{ready}(r) \subseteq \bigcup \text{Acc}(q)$;
- for every $r'$ such that $\lambda(r, a) = r'$, $(r', \delta(q, a)) \in \pi$.

Definition 19 Given an automaton $M$ and a MAS $S$, $\rho_{\text{n}}(M, S)$ is the automaton $M' = (R \times Q, (r^0, q^0), \lambda', G \times Q)$ with:

$$
\lambda'(r, q, a) = \begin{cases}
\lambda(r, a), & \text{if } a \in \bigcup \text{Acc}(q) \\
\text{undefined otherwise}
\end{cases}
$$

Theorem 11 Given an automaton $M$ and a MAS $S$, we have: $\rho_{\text{n}}(M, S) \sim_{\text{q}} S$. Moreover, for all $M_S \models S$, the automata $M \times M_S$ and $\rho_{\text{n}}(M, S) \times M_S$ are bisimilar.

Proof $\rho_{\text{n}}(M, S) \sim_{\text{q}} S$: let $\pi$ be the simulation relation such that for any state $(r, q)$ of $\rho_{\text{n}}(M, S)$, $(r, q) \in \pi$: by definition of $\rho_{\text{n}}$, $\text{ready}(r, q) \subseteq \bigcup \text{Acc}(q)$.
Given two \( S_1 \) and \( S_2 \) and an automaton \( M \) such that \( M \sim_{\mathcal{U}} S_2 \) and for all \( M_2 \models S_2 \) we have \( M \times M_2 \models \models S_1 \), then \( M \models \models S_1 / S_2 \).

Proof Let \( \pi \) be a simulation relation such that \( \langle r^0, (q_1^0, q_2^0) \rangle \in \pi \) and for any \((r, (q_1, q_2)) \in \pi\), \( a \) and \( r' \) such that \( \lambda(r, a) = r' \), \( (r', \delta((q_1, q_2), a)) \in \pi \). This definition of \( \pi \) is only correct if for any \((r, (q_1, q_2)) \in \pi\) and \( a \) such that \( \lambda(r, a) \) is defined, \( \delta((q_1, q_2), a) \) is defined. As \( M \sim_{\mathcal{U}} S_2 \), \( a \in \bigcup \text{Acc}(q_2) \), so there exists an \( X \in \text{Acc}(q_2) \) such that \( a \in X \) and then \( \delta_2(q_2, a) \) is defined (as \( S_2 \) is well-formed). As \( \delta_2(q_2, a) \) is defined, there exists an automaton \( M_2 \models S_2 \) with a state \( r_2 \) implementing \( q_2 \) such that \((r, r_2)\) is reachable in \( M \times M_2 \) and \( \lambda_2(r_2, a) \) is defined. Then, \( \lambda(r, r_2, a) \) is defined and, as \( M \times M_2 \models S_1 \), it implies that \( \delta((q_1, q_2), a) \) is defined.

There are then three points to prove for any \((r, (q_1, q_2)) \in \pi\):
- \( \text{ready}(r) \in \text{Acc}(q_1, q_2) \): by definition of the pre-quotient, \( \text{ready}(r) \) must verify two properties:
  - \( \forall X_2 \in \text{Acc}_2(q_2), \text{ready}(r) \cap X_2 \subseteq \text{Acc}_1(q_1) \):
    - Let \( X_2 \) be an element of \( \text{Acc}_2(q_2) \). There exists an automaton \( M_2 \) with a state \( r_2 \) such that \( (r, r_2) \) is reachable in \( M \times M_2 \) and \( \text{ready}(r_2) = X_2 \).
    - Then, as \( M \times M_2 \models S_1 \) by a simulation relation \( \pi_\times \) and \( ((r, r_2), q_1) \in \pi_\times \), \( \text{ready}(r) \cap \text{ready}(r_2) = \text{ready}(r) \cap X_2 \subseteq \text{Acc}_1(q_1) \).
  - \( \text{ready}(r) \subseteq \bigcup \text{Acc}_1(q_1) \cap \bigcup \text{Acc}_2(q_2) \):
    - By definition of \( \sim_{\mathcal{U}} \), \( \text{ready}(r) \subseteq \bigcup \text{Acc}_2(q_2) \).
    - Assume that \( \text{ready}(r) \not\subseteq \bigcup \text{Acc}_1(q_1) \): there is an \( a \in \text{ready}(r) \) such that \( a \not\in \bigcup \text{Acc}_1(q_1) \). As \( M \) has no unnecessary transition regarding \( S_2 \), there is a model \( M_2 \) of \( S_2 \) with a state \( r_2 \) such that \( (r, r_2) \) is reachable in \( M \times M_2 \) and \( a \in \text{ready}(r_2) \). Then, the transition \((r, r_2, a)\) is defined in \( M \times M_2 \). As \( M \times M_2 \models S_1 \), the transition \((q_1, a)\) has to be defined, which is in contradiction with the hypothesis that \( a \notin \bigcup \text{Acc}_1(q_1) \). Thus, \( \text{ready}(r) \subseteq \bigcup \text{Acc}_1(q_1) \).
- \( r \in G \) implies \((q_1, q_2) \in F \) if, that is \( q_1 \in F_1 \) or \( q_2 \notin F_2 \):
  - This property is only false if \( r \in G \), \( q_1 \notin F_1 \) and \( q_2 \in F_2 \). In this case, there exists an automaton \( M_2 \models S_2 \) with a state \( r_2 \) such that \( (r, r_2) \) is reachable in \( M \times M_2 \) and \( r_2 \in G_2 \). Then, \( M \times M_2 \models S_1 \) by a simulation relation \( \pi_\times \), \((r, r_2), q_1) \in \pi_\times \) and \( (r, r_2) \) is marked. By definition of satisfaction, it implies that \( q_1 \in F_1 \), which is impossible as we already know that \( q_1 \notin F_1 \). So \( r \in G \) implies \((q_1, q_2) \in F \).
  - for any \( a \) and \( r' \) such that \( \lambda(r, a) = r' \), \( (r', \delta((q_1, q_2), a)) \in \pi \) is trivial by definition of \( \pi \). \( \Box \)

**Corollary 1 (Completeness)** Given two \( S_1 \) and \( S_2 \) and an automaton \( M \) such that for all \( M_2 \models S_2 \), we have \( M \times M_2 \models \models S_1 \), then \( \rho_u(M, S_2) \models \models S_1 / S_2 \).

Proof By Theorem 11 we know that \( \rho_u(M, S_2) \sim_{\mathcal{U}} S_2 \) and for any \( M_2 \models S_2 \), \( \rho_u(M, S_2) \times M_2 \) is bisimilar to \( M \times M_2 \), which implies that \( \rho_u(M, S_2) \times M_2 \models \models S_1 \). Then, Theorem 12 implies that \( \rho_u(M, S_2) \models \models S_1 / S_2 \). \( \Box \)
Acc(0) = \{\{a\}, \{a, b\}\}
Acc(1) = \{\{a\}, \{a, b\}\}
Acc(2) = \{\emptyset\}

Acc(0') = \{\{a, b\}\}
Acc(1') = \{\{a\}, \{a, b\}\}
Acc(2') = \{\emptyset\}

Remark 2 This pre-quotient operation returns a specification $S_1 / S_2$ which does not always have a compatible reachability with the divisor $S_2$. For example, consider the specifications $S_1$ and $S_2$ of Figures 9(a) and 9(b); their prequotient is shown in Figure 9(c). If we take the models $M_1$ of $S_1 / S_2$ (Figure 9(e)) and $M_2$ of $S_2$ (Figure 9(d)), their product is not terminating as it has a livelock; hence, the result of the prequotient does not have a compatible reachability with the divisor $S_2$. One may think that the prequotient is erroneous and should not allow to realize only the transition $a$ from the state $(0, 0')$ (ie. that $\text{Acc}((0, 0'))$ should only be $\{\{a, b\}\}$). Indeed, it would forbid the incorrect model, but it would also disallow some valid models such as $M_2'$ of Figure 9(f) which does not always realize the transition $b$, but does it once and can thus synchronize with any model of $S_2$, as they always realize this transition.

The construction proposed in the next section will allow to refine $S_1 / S_2$ in order to enforce the criterion in Definition 14 for compatible reachability.

6.2 Incompatible reachability correction

We now consider the following problem: consider two MAS $S_1$ and $S_2$ that do not have a compatible reachability, can we refine $S_1$ such that the obtained specification $S_1'$ has a compatible reachability with $S_2$? Solving this problem allows to automatically assist the system designer when a step of the design flow leads to incompatible specifications. This problem also arises after the pre-quotient operations as pointed out in Remark 2 and the proposed solution will be reused in Section 6.3.
6.2.1 Deadlock correction

First, given $S_1$ and $S_2$ two non-deadlock-free MAS, we propose to refine the set of models of $S_1$ such that the obtained MAS $S'_1$ is deadlock-free with $S_2$. For this, we iteratively eliminate all pairs of states $(q_1, q_2)$ such that $\text{DeadFree}(q_1, q_2)$ is false, as described in Algorithm 4.

**Algorithm 4** dead_correction ($S_1$: MAS, $S_2$: MAS): MAS

1: $S'_1 \leftarrow S_1$
2: $\text{dead_pairs} \leftarrow \{(q_1, q_2) \mid q_1 \in Q_1 \land q_2 \in Q_2 \land \neg \text{DeadFree}(q_1, q_2)\}$
3: for all $(q_1, q_2) \in \text{dead_pairs}$ do
4:    $\text{Acc}_1(q_1) \leftarrow X_1 \mid X_1 \in \text{Acc}_1(q_1) \land \forall X_2 \in \text{Acc}_2(q_2). X_1 \cap X_2 ≠ \emptyset$
5:    $S'_1 \leftarrow \rho(S'_1)$
6: end for
7: return $S'_1$

Note that Algorithm 4 may return $S'_1$ which then means that for any model $M_1$ of $S_1$, there exists a model $M_2$ of $S_2$ such that $M_1 \times M_2$ has a deadlock.

**Theorem 13 (Deadlock correction)** Given two MAS $S_1$ and $S_2$, $M_1 \models S_1$ is such that for any $M_2 \models S_2$, $M_1 \times M_2$ is deadlock-free if and only if $M_1 \models \text{dead_correction}(S_1, S_2)$.

**Proof** ($\Rightarrow$) Assume that for any $M_1 \models S_1$ and $M_2 \models S_2$, $M_1 \times M_2$ is deadlock-free. By Theorem 3, $S_1$ and $S_2$ are deadlock-free, which implies that there is no pair of states $(q_1, q_2)$ such that $\neg \text{DeadFree}(q_1, q_2)$. Thus, the set dead_pairs in Algorithm 4 is empty and dead_correction($S_1, S_2$) = $S_1$, so $M_1 \models \text{dead_correction}(S_1, S_2)$.

($\Leftarrow$) Assume that there exists an $M_2 \models S_2$ such that $M_1 \times M_2$ has a deadlock pair of states $(r_1, r_2)$. By Theorem 3, this implies that $S_1$ and $S_2$ are not deadlock-free and thus that there exists a pair of states $(q_1, q_2)$ (implemented by $(r_1, r_2)$) reachable in $\text{Un}(S_1) \times \text{Un}(S_2)$ such that $\neg \text{DeadFree}(q_1, q_2)$. Then, in dead_correction($S_1, S_2$), either the acceptance set of $q_1$ has been reduced so that $\text{Compat}(\text{Acc}_1(q_1), \text{Acc}_2(q_2))$ is true and $\text{DeadFree}(q_1, q_2)$ or $q_1$ is not reachable anymore and then $(q_1, q_2)$ is not reachable in $\text{Un}(S_1) \times \text{Un}(\text{dead_correction}(S_1, S_2))$. Consequently, either $\emptyset \neq \text{Acc}_1(q_1)$ or $(r_1, r_2) \notin \pi$, and thus $M_1$ is not a model of dead_correction($S_1, S_2$). □

6.2.2 Livelock correction

Secondly, given $S_1$ and $S_2$ two deadlock-free MAS, we propose to refine the set of models of $S_1$ such that the obtained specification $S'_1$ is livelock-free with $S_2$. In order to avoid potential livelocks between two MAS, we will use two methods: removing some transitions so that states from which it is not possible to guarantee termination will not be reached and forcing some transitions to be eventually realized in order to guarantee that it will be possible to leave cycles without marked states. For this last method, we introduce marked acceptance specifications with priorities that are MAS in which we identify some transitions called priorities; in the satisfaction relation, we then constraint to eventually realize these transitions.
Definition 20 (MAS with priorities) A marked acceptance specification with priorities (MASp) is a tuple $(Q, q_0, \delta, \text{Acc}, P, F)$ where $(Q, q_0, \delta, \text{Acc}, F)$ is a MAS and $P : 2^{Q \times \Sigma}$ is a set of priorities.

Definition 21 (Satisfaction) An automaton $M$ implements a MASp $S$ if $M$ implements the underlying MAS and for all $P \in P$, either $\forall (q, a) \in P$, $\forall r, (r, q) \not\in \pi$ or $\exists (q, a) \in P, \exists r, (r, q) \in \pi \land a \in \text{ready}(r)$.

Intuitively, $P$ represents a conjunction of disjunctions: at least one transition from each element of $P$ must be implemented by the models of the specification.

Let $S_1$ and $S_2$ be two MAS and $q_1$ a state of $S_1$ such that $q_1$ belongs to a livelock. Then, there exists a cycle $C_1$ in $S_1$ and its partner $C_2$ in $S_2$ such that the conditions given in Definition 12 are false. Given this cycle, Algorithm 5 ensures that the possible livelock will not happen, either by adding some priorities or removing some transitions. We then iterate over the possible cycles, fixing the cycles which may cause a livelock, as described in Algorithm 6.

Example 7 [Figure 10] shows some examples of the application of the different rules defined in Algorithm 6.

For the two MAS of Figure 9, the correction is just to add a priority for the transition $((0, 0), b)$ of the pre-quotient: it disallows the invalid models such as $M_1^2$ of Figure 9(e) but is permissive enough to allow valid models like $M_1^2$ of Figure 9(d).

Algorithm 5 live_correction_cycle $(S_1$: MASp, $C_1$: Cycle, $S_2$: MAS, $C_2$: Cycle): MASp

1: if $A_{C_2} \neq \emptyset$ then
2: $Q_A \leftarrow \{(q_1 | Q_2(q_1) \in \text{dom}(A_{C_2}) \land \forall A \in A_{C_2}(Q_2(q_1)), A \cap \text{ready}(q_1) \neq \emptyset\}$
3: if $Q_A \neq \emptyset$ then
4: $P \leftarrow \{\bigcup_{1 \leq i \leq |Q_A|} \{(q_i, a) | a \in X_i \} | X_i \in \{A \cap \text{ready}(q_i) | A \in A_{C_2}(Q_2(q_i))\}\}$
5: return $(Q_1, q_0^1, \delta_1, \text{Acc}_1, P_1 \cup P, F_1)$
6: end if
7: else if $\text{dom}(C_2) \cap F_2 = \emptyset$ then
8: $\text{Acc}' \leftarrow \text{Acc}_1$
9: for all $q_1 \in \{Q_1(q_2) | q_2 \in \text{dom}(C_2)\}$ do
10: $\text{Acc}'(q_1) \leftarrow \{X | X \in \text{Acc}_1(q_1) \land \forall O \in \text{O}_{C_2}(Q_2(q_1)), X \cap O \neq \emptyset\}$
11: end for
12: return $\rho((Q_1, q_0^1, \delta_1, \text{Acc}', P_1, F_1))$
13: end if
14: $\text{Acc}' \leftarrow \text{Acc}_1$
15: for all $q_1 \in Q_1$ do
16: $\text{Acc}'(q_1) \leftarrow \{X | X \in \text{Acc}_1(q_1) \land \forall a \in X, \delta(q_1, a) \not\in \text{dom}(C_1)\}$
17: end for
18: return $\rho((Q_1, q_0^1, \delta_1, \text{Acc}', P_1, F_1))$

Theorem 14 (Livelock correction) Given two MAS $S_1$ and $S_2$, $M_1 \models S_1$ is such that for any $M_2 \models S_2$, $M_1 \times M_2$ is livelock-free if and only if $M_1 \models \text{live_correction}(S_1, S_2)$. 
Acc(0) = \{\{a\}, \{a, d\}\}
Acc(1) = \{\{a\}, \{a, c\}\}
Acc(2) = \{\{b\}, \{b, c\}, \{b, d\}, \{b, c, d\}\}
Acc(3) = Acc(4) = Acc(5) = \{\emptyset\}
P = \{(0, d), (2, c), (2, d)\}

(a) live_correction(S_1, S_2): example of compatible reachability correction for the first case (lines 1 to 6)

Acc(?', 0') = \Lambda acc(0, 0') = Acc(2, 0') = \{\{a, b, c, d\}\}
Acc(?', 1') = \{\{a, c\}, \{a, d\}, \{a, c, d\}\}
Acc(1, 1') = \{\{a, c\}, \{a, c, d\}\}
Acc(?', 2') = \Lambda acc(3, 2') = Acc(5, 2') = Acc(4, 3') = Acc(?', 3') = \{\emptyset\}

(b) live_correction(S_2, S_1): example of compatible reachability correction for the second case (lines 7 to 12)

Fig. 10: Examples of livelock correction for the specifications of Figure 7
Algorithm 6 live_correction \((S_1; MAS, S_2; MAS): MASp\)

1: \(S'_1 \leftarrow (Q_1, q^0_1, \delta_1, \text{Acc}_1, \emptyset, F_1)\)
2: for all \(C_1 \in \text{Cycle}_{\text{im}}(S_1)\) such that \(\forall q_1 \in \text{dom}(C_1), (Q_2(q_1)) = 1\) do
3:    if \(\neg \text{LiveFree}(C_1, S_2)\) then
4:      \(C_2 \leftarrow \{Q_2(q) \mapsto C_1(q) \mid q \in \text{dom}(C_1)\}\)
5:      \(S'_1 \leftarrow \text{live_correction}_\text{cycle}(S'_1, C_1, S_2, C_2)\)
6:    end if
7: end for
8: return \(S'_1\)

Proof \((\Rightarrow)\) Assume that for any \(M_1 \models S_1\) and \(M_2 \models S_2, M_1 \times M_2\) is live-lock-free. By \Theorem{6} \(S_1\) and \(S_2\) are live-lock-free which means, by \Definition{13} that for any implementable cycle \(C_1\) in \(S_1\) such that its states have a partner in \(S_2\), we have \LiveFree\(C_1, S_2\). In this case, the test at line \(\mathcal{R}\) of Algorithm 6 is always false and so \live_correction\(S_1, S_2\) returns \(S_1\), of which \(M_1\) is a model by hypothesis.

\((\Leftarrow)\) Assume that there exists an \(M_2 \models S_2\) such that \(M_1 \times M_2\) has a livelock.
- If there exists a cycle \(C_1 \in \text{Cycle}_{\text{im}}(S_1)\) which is implemented in \(M_1\) by the states of the loop in which there is a livelock when combined with \(M_2\). Thus, \live_correction\(C_1\) will be called with \(C_1\). There are three cases:
  - If \(\mathcal{A}_{C_2}\) is not empty, some transitions are present in all the models of \(S_2\) implementing \(C_2\), so the models of \(S_1\) should realize (at least) one of these transitions once. If it is possible, some priorities are added, see lines [3] to [5] of Algorithm 5. This addition will only remove the models of \(S_1\) that never realize any transition in \(\mathcal{A}_{C_2}\), and thus that will have a livelock with some models of \(M_2\) (which only realize the transition of \(\mathcal{A}_{C_2}\)).
  - If \(\mathcal{A}_{C_2}\) is empty but there is no marked state in \(C_2\), all the models of \(S_2\) implementing \(C_2\) will eventually realize a transition of \(\mathcal{O}_{C_2}\) in order to reach a marked state (as there is none in the cycle). The only way to avoid a livelock with any model of \(S_2\) is to realize all the transitions that these models may use to reach a marked state, which is done in lines [7] to [12].
  - Otherwise, there will always be a possible livelock with some models of \(S_2\), so the only possibility is to disallow all the models which implement this cycle, which is done in lines [13] to [18].

So \(M_1\) is not a model of the \(\text{MASp}\) returned by \live_correction\(\text{cycle}\) for \(C_1\) and thus it is not a model of \live_correction\(S_1, S_2\).
- Otherwise, multiple cycles are implemented simultaneously and there will also be livelocks in the models which implement only one of the cycles. As argued in the previous item, applying \live_correction\(\text{cycle}\) for these cycles will generate a specification forbidding the corresponding models, and then \(M_2\) will not be a model of the resulting specification as it only combines the behavior of these models.

By applying successively these operations (\dead_correction\ and \live_correction\), we define the following operation \(\rho_T:\)

\[
\rho_T(S_1, S_2) = \text{live_correction}(\text{dead_correction}(S_1, S_2), S_2)
\]

Given two \MAS\ \(S_1\) and \(S_2\), it refines the set of models of \(S_1\) as precisely as possible so that their product with any model of \(S_2\) is terminating.
A Specification Theory for Reachability by Design

Theorem 15 (Incompatible reachability correction) Given two MAS $S_1$ and $S_2$, for any $M \models p_T(S_1, S_2)$ and $M_2 \models S_2$, $M \times M_2$ is terminating, and an $M_1 \models S_1$ is such that for any $M_2 \models S_2$, $M_1 \times M_2$ is terminating if and only if $M_1 \models p_T(S_1, S_2)$.

Proof For any $M \models p_T(S_1, S_2)$ and $M_2 \models S_2$, $M \times M_2$ is terminating if and only if $M \times M_2$ is deadlock-free and livelock-free. By theorems 3 and 6, this is true if and only if $p_T(S_1, S_2)$ and $S_2$ are deadlock-free and livelock-free, which is true by definition of $p_T$ and Theorems 13 and 14. $\square$

6.3 Bounded Reachability

A MASp explicitly requires to eventually realize some transitions fixed in the priority set $P$. When bounding the delay before the implementation of the transitions, a MASp becomes a standard MAS.

More precisely, given a MASp $S_p$ and an integer $k$ to model a frequency, we can generate a MAS $S_b$ with a $k$-bounded reachability: for each transition $(q, a)$ in the priority set, the models of $S_b$ have to realize the transition by $a$ when $q$ is implemented for the $k$-th time; for the other implementations of $q$, we allow to realize $a$ or any ready set allowing to reach $q$.

For example, considering the MASp $S_p$ given by Figure 11(a) with the transition $(0, b)$ in its priority set, we can generate a 3-bounded MAS $S_b$ as given by Figure 11(b). The cycle involving the state 0 is unfolded to duplicate the state 0 three times (states $0', 0''$ and $0'''$ in $S_b$). The acceptance set of $0'$ and $0''$ allows either to realize $b$ which was prioritized or to reach the state $0'''$ whose acceptance set only contains the ready sets from $\text{Acc}(0)$ including the expected action $b$.

6.4 Quotient definition

We can now combine the pre-quotient and cleaning operations to define the quotient of two MAS.

Definition 22 The quotient of two MAS $S_1$ and $S_2$, denoted $S_1/S_2$, is $\rho_T(S_1/S_2, S_2)$. 
Theorem 16 (Soundness) Given two MAS $S_1$ and $S_2$ and an automaton $M \models S_1/S_2$, for any $M_2 \models S_2$, we have $M \times M_2 \models S_1$.

Proof By Theorem 15, we know that for any $M_2 \models S_2$, $M \times M_2$ is terminating. Thus, Theorem 10 implies that $M \times M_2 \models S_1$. $\square$

Theorem 17 (Completeness) Given two MAS $S_1$ and $S_2$ and an automaton $M$ such that for all $M_2 \models S_2$, we have $M \times M_2 \models S_1$, then $\rho_{u}(M, S_2) \models S_1/S_2$.

Proof We know by Corollary 1 that $\rho_{u}(M, S_2) \models S_1/S_2$. We then deduce by Theorem 15 that $\rho_{u}(M, S_2) \models S_1/S_2$. $\square$

Note that the approach described in 6.3 allows to have an homogeneous quotient and thus an iterative process: one may compute $(S_1/S_2)/S_3$ by first computing $S_1/S_2$, bound the obtained specification and then compute its quotient with $S_3$.

As a consequence, incremental design of component-based systems is enabled. Given $S_1$ and $S_2$, the system designer can either distribute the implementation tasks $S_1/S_2$ and $S_2$ or, alternatively, decide to reuse an off-the-shelf component implementing $S_2$. The product of the models of $S_2$ and $S_1/S_2$ will realize $S_1$ and will, in particular, satisfy by construction the reachability objectives it includes.

7 Discussion on the relevance of an acceptance-based approach versus a modal one

Nota: some knowledge about modal specifications is assumed in this section.

While our first attempt for bottom-up design of reachability properties was based on a marked version of modal specifications [11], we have switched in this article for an acceptance-based approach. The reason for this is twofold. First, MAS are more expressive than marked modal specifications (MMS) as modal specifications (MS) correspond to a strict subset of acceptance specifications (AS). Indeed, as already advocated in [20], let $\text{must}(q)$ and $\text{may}(q)$ be respectively the set of actions that are required and allowed in a state $q$ of an MS, we can embed MS into AS by associating to $q$ the following acceptance set: $\text{Acc}(q) = \{ X \mid \text{must}(q) \subseteq X \subseteq \text{may}(q) \}$. Then the set of models of the MS is exactly the set of models of the related AS. The structure of these derived acceptance sets is very particular: they are closed under union, intersection and convexity (that is, for $X, Y \in \text{Acc}(q)$ and $Z$ such that $X \subseteq Z \subseteq Y$, we have: $X \cup Y$, $X \cap Y$ and $Z$ in $\text{Acc}(q)$) while in the acceptance case, any ready set can be put in without restriction.

Based on this embedding, the constructions proposed in this article for product and conjunction of MAS can immediately be applied to MMS as the manipulations made in these operators preserve the particular structure of their acceptance sets. This is however not the case for quotient; this does not come from a flaw in our proposed construction but this is due to the limited expressivity offered by the modal case and this is the second reason to promote an acceptance based-approach. Consider indeed the two MMS $S_1$ and $S_2$ in Figure 12 in which optional transitions are represented with dashed arrows while required transitions are plain arrows. A correct and complete modal quotient in this example would tell in the
Fig. 12: Two MMS showing that a modal quotient cannot exist

initial state of $S_1/S_2$ that at least one action between $a$ and $b$ is required. This cannot be encoded by modalities but it can be correctly stated by the acceptance set $\{\{a\}, \{b\}, \{a, b\}, \{b, c\}, \{a, b, c\}\}$.

8 Related work and conclusion

In this article, we have introduced marked acceptance specifications, a specification formalism for underspecified systems under reachability constraints. We have developed a specification theory for them with refinement, product, conjunction and quotient allowing to guarantee by construction reachability properties. The proposed techniques have been implemented in a tool, MAccS [28].

Modal specifications enriched with marked states have been first introduced in [13] for the supervisory control of services. Product of MMS has been investigated in [11]. As quotient is not considered in these papers, the need for the more expressive framework of MAS was not found out as pointed in Section 7.

MAS can also be related to 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 [29] and operating guidelines [25, 23]. While both formalisms have a product operator, they are missing the conjunction and quotient operators.

The reachability considered in this paper can be stated in CTL by $\text{AG(EF(\text{final}))}$ and cannot be captured in LTL. Thus, satisfiability of a MAS cannot be based on the LTL model checking for modal specifications studied in [5].

A restriction of the theory developed in this article concerns the underlying determinism of MAS. Alternative semantics for nondeterministic modal specifications have been recently proposed in [24, 10]. A specification theory under reachability constraints in this context is left for future investigations.

References


1 Although this particular example could be tackled with disjunctive modal specifications [21], more sophisticated examples can be elaborated to similarly show that acceptance sets are necessary as soon as we deal with the quotient issue.