Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 215 (2008) 93–110
www.elsevier.com/locate/entcs

Residual for Component Specifications

Jean-Baptiste Raclet[1]

IRISA
Campus de Beaulieu
F-35042 Rennes Cedex, France

Abstract

We address the problem of component reuse by describing a quotient operation. Starting from the specifications of the behaviors of the component and of the desired overall system, this operation computes the residual specification characteristic of the systems that, when composed with the given component, satisfy the overall specification. This problem is solved when behaviors are given by modal or acceptance specifications and when composition allows mixed product and internalization of events. We show on an example how weak form of liveness constraint may be taken into account by this technique.

Keywords: Component-based design, Behavioral interface, Modal and acceptance specifications, Residual of specifications, Behavioral reuse.

1 Introduction

In current component platforms, a component is equipped with an interface which lists the signature of the services that the entity offers. This light description is sufficient to enable component reuse. However, it provides no guarantee that the reused component will interact suitably with its environment and critical behavioral mismatch such as deadlock may occur.

In this paper, we investigate the extension of component interfaces to behavioral descriptions in order to provide techniques to reason about component reuse at a behavioral level rather than at a signature level. More precisely, we study the following issues: can a component, the behavior of which is described in its interface by the specification $\mathcal{S}_1$, be used to build a system satisfying a

[1] Email: Jean-Baptiste.Raclet@irisa.fr

doi:10.1016/j.entcs.2008.06.023

global specification $\mathcal{S}$? If so, what are the components that, when composed with the reused component, constitute a composite system satisfying $\mathcal{S}$?

These problems can be seen as kinds of supervisor synthesis with the main difference that the reused component (corresponding to the plant in control theory) is a black-box. Indeed, a component must be reusable from the description of its behavior in its interface and not from its implementation which is unknown as it may have been developed by a third party.

Behavioral reuse of components can also be related to some works in equation solving. This problem introduced in [11] consists in solving the equation $\mathcal{S}_1 \times X \simeq \mathcal{S}$ with $\mathcal{S}_1$ called the context, $\mathcal{S}$ a global specification and $\simeq$ a trace equivalence relation. Solutions for this problem were proposed for various models of specification: finite automata [11,5], finite state machine [14] (with inclusion of traces as equivalence relation), CCS or CSP processes [12,10] (with bisimulation as equivalence relation) or input/output automata [8].

In this paper, we introduce modal specifications and acceptance specifications as intuitive formalisms for behavioral interface description. From the expressiveness point of view, they allow to state some forms of liveness properties. For each of this formalism, a quotient operation is defined to address the problem of behavioral reuse of a component as computing the residual specification $\mathcal{S}/\mathcal{S}_1$. We study two kinds of composition between components: synchronous composition and mixed product with internalization of events. This paper is organized as follows: after short preliminaries, section 3 introduces modal specification. The notion of residual specification for behavioral reuse is formalized in section 4. Then, the quotient of modal specification is proposed in section 5. Expressivity is improved thanks to acceptance specifications in section 6. Then, section 7 is devoted to the quotient operation corresponding to the use of mixed product with internalization of events as composition operation. An example and a hint for some line of future work conclude the paper.

2 Preliminaries

We use Σ to denote the universe of events. Σ^* denotes the set of all finite length event sequences, called traces, with ϵ the zero length trace. A language L over Σ is a subset of Σ^*. For $u \in L$, we let L_u be the set of events a such that the trace u followed by a (noted $u.a$) belong to L. The operations $\cap$ and $\cup$ over languages correspond to the set-theoretic intersection and union. The complement of the language L over Σ, noted $\neg L$, is the set $\Sigma^* \setminus L$. Given a trace $w \in \Sigma'$ and a subset of events $\Sigma \subseteq \Sigma'$, the projection of w on Σ denoted $\Pi_\Sigma(w)$ is the trace obtained from w by erasing the events not belonging to Σ.

A trace u is said to be a prefix of a trace w (noted $u \preceq w$), if $w = uv$. Given a language L its prefix closure $\tilde{L}$ consists of all the prefixes of all the traces in L. A language L is said prefix-closed when $L = \tilde{L}$.

3 Modal specifications

In this section we introduce modal specification as a formalism to specify sets of languages:

3.1 Modal specification and their models

Modal automata are standard finite automata with modalities "may" or "must" on transitions. They were originally used in [9] to study the refinement of actions. We now introduce modal specifications which generalize the use of modalities on events to non-necessarily regular languages:

Definition 3.1 A modal specification $\mathcal{S}$ over Σ is a triple $\langle L, must, may\rangle$ where L is a prefix-closed language over Σ and $must, may : L \to \mathcal{P}(\Sigma)$ are partial functions that type events: for $u \in L$,

- $a \in may(u)$ means that the event a is allowed after the trace u;
- $a \in must(u)$ means that the event a is required after the trace u;
- $a \notin may(u)$ (often denoted $a \in mustnot(u)$) means that a is forbidden after u.

For consistency, the following conditions are required for all $u \in L$:

(C1) $must(u) \subseteq may(u)$;

(C2) $may(u) = L_u$.

The consistency condition (C1) expresses that the events required after a trace u of the specification must also be allowed.

In the sequel the elements of the tuple corresponding to the modal specification $\mathcal{S}$ could be denoted $L(\mathcal{S})$, $must(\mathcal{S})$ and $may(\mathcal{S})$. The prefix-closed language $L(\mathcal{S})$ may be called the *support* of $\mathcal{S}$. The set of modal specifications over Σ is denoted $\mathcal{MS}(\Sigma)$.

A model of a modal specification is a language. The definition of the validation relation is the following:

Definition 3.2 A prefix-closed language $C \subseteq \Sigma^*$ is a model of the modal specification $\mathcal{S} \in \mathcal{MS}(\Sigma)$, noted $C \models \mathcal{S}$, if:

- $C \subseteq L(\mathcal{S})$;
- for all $u \in C$, $must(\mathcal{S})(u) \subseteq C_u \subseteq may(\mathcal{S})(u)$.

The interpretation of this definition is the following: every trace u of a model C is also a trace of $\mathcal{S}$ and the events available in C after a trace u are all the required events ($must(u) \subseteq C_u$) and none of the forbidden events ($C_u \subseteq may(u)$) after the trace u in the specification.

Example 3.3 The specification $\top = \langle \Sigma^*, must, may \rangle$ with, for all $u \in \Sigma^*$, $must(u) = \emptyset$ and $may(u) = \Sigma$, admit every language over Σ as model. $\bot = \langle \emptyset, \emptyset, \emptyset \rangle$ has no model.

Modal specifications are ordered by the following relation:

Definition 3.4 $\mathcal{S}_1 \leq \mathcal{S}_2$ if and only if:

- $L(\mathcal{S}_1) \subseteq L(\mathcal{S}_2)$ and
- $\forall u \in L(\mathcal{S}_1)$, $\begin{cases} may(\mathcal{S}_1)(u) \subseteq may(\mathcal{S}_2)(u) \\ must(\mathcal{S}_1)(u) \supseteq must(\mathcal{S}_2)(u) \end{cases}$

Remark 3.5 Any prefix-closed language C can be viewed as a modal specification where $may(u) = must(u) = C_u$ for all trace $u \in C$. Hence $C \models \mathcal{S}$ if and only if $C \leq \mathcal{S}$.

Proposition 3.6 *$\mathcal{S}_1 \leq \mathcal{S}_2$ if and only if every model C of $\mathcal{S}_1$ is also a model of $\mathcal{S}_2$.*

When $L(\mathcal{S})$ is regular, $\mathcal{S}$ can be viewed as the unfolding of a modal automaton [9] all of whose states are final. The logical fragment equivalent to modal automata has been identified in [6]. It is a fragment of the mu-calculus called the conjunctive nu-calculus as it includes conjunctions and greatest fixpoints along with diamond and box modalities. It is strictly less expressive than mu-calculus as neither disjunction nor eventualities can be stated.

3.2 Pseudo-modal specifications

For technical reasons, we shall consider an extension of the class of modal specification called *pseudo-modal specification* and denoted $p\mathcal{MS}$. They are specifications where the consistency condition (C1) of the definition 3.1 is relaxed for some traces u:

Definition 3.7 A pseudo-modal specification $p\mathcal{S} \in p\mathcal{MS}(\Sigma)$ is a triple $\langle L, must, may \rangle$ where L is a prefix-closed language over Σ and $must, may : L \to \mathcal{P}(\Sigma)$ are partial functions that type events with *no* consistency constraint between $may(u)$ and $must(u)$.
A trace u of $p\mathcal{S}$ such that $must(u) \not\subseteq may(u)$ is said incoherently specified.

The relations $\models$ and $\leq$ for pseudo-modal specifications are the same as for modal specifications (cf. def. 3.2 and 3.4). Hence, if $must(u) \not\subseteq may(u)$ then u can't belong to a language C model of $p\mathcal{S}$ because this would imply, for $a \in must(u)$ and $a \notin may(u)$, on one hand that $u.a \in C$ (as $a \in must(u)$) and, on the other hand, that $u.a \notin C$ (as $a \notin may(u)$). This remark give the intuition of a reduction of a pseudo-modal specification into a modal specification with respect to its set of models; we let $\rho : p\mathcal{MS} \to \mathcal{MS}$ be the operation such that:

Proposition 3.8 *Either a pseudo-modal specification $p\mathcal{S}$ has no model or there exists a largest modal specification $\rho(p\mathcal{S})$ smaller than $p\mathcal{S}$, and $\rho(p\mathcal{S})$ has the same models as $p\mathcal{S}$.*

The modal specification $\rho(p\mathcal{S})$ is obtained from the pseudo-modal specification $p\mathcal{S}$ by application of the following steps:

(i) basis: we let R be a copy of $p\mathcal{S}$;

(ii) we let U be the set of traces u incoherently specified in R; we remove U from $L(R)$; for all trace v such that $v.a = u$ with $u \in U$, we remove a from $may(R)(v)$ to enforce the consistency condition $(C2)$;
When $a \in must(R)(v)$, v becomes incoherently specified in R. Thus, we repeat this step until there is no more incoherently specified trace in $L(R)$;

(iii) $\rho(p\mathcal{S})$ is built from R: $L(\rho(p\mathcal{S})) = \widetilde{L(R)}$ and for all $u \in L(\rho(p\mathcal{S}))$, $may(\rho(p\mathcal{S}))(u) = may(R)(u)$ and $must(\rho(p\mathcal{S}))(u) = must(R)(u)$.

3.3 Lattice of modal specifications

The set of modal specifications $\mathcal{MS}$ equipped with the partial order $\leq$ is a complete distributive lattice (hence a bounded lattice) where the meet $\mathcal{S}_1 \wedge \mathcal{S}_2$ is the reduction of the pseudo-modal specification $\mathcal{S}_1 \& \mathcal{S}_2$ over $L(\mathcal{S}_1) \cap L(\mathcal{S}_2)$ with:

$$\forall u \in L(\mathcal{S}_1) \cap L(\mathcal{S}_2), \quad \begin{cases} may(\mathcal{S}_1 \& \mathcal{S}_2)(u) &= may(\mathcal{S}_1)(u) \cap may(\mathcal{S}_2)(u) \\ must(\mathcal{S}_1 \& \mathcal{S}_2)(u) &= must(\mathcal{S}_1)(u) \cup must(\mathcal{S}_2)(u) \end{cases}$$

and the join $\mathcal{S}_1 \vee \mathcal{S}_2$ is the modal specification over $L(\mathcal{S}_1) \cup L(\mathcal{S}_2)$ with:

$$\begin{cases} \text{if } u \in L(\mathcal{S}_1) \cap L(\mathcal{S}_2),\ may(\mathcal{S}_1 \vee \mathcal{S}_2)(u) = may(\mathcal{S}_1)(u) \cup may(\mathcal{S}_2)(u), \\ \qquad\qquad\qquad\qquad must(\mathcal{S}_1 \vee \mathcal{S}_2)(u) = must(\mathcal{S}_1)(u) \cap must(\mathcal{S}_2)(u) \\ \text{if } u \in L(\mathcal{S}_1) \setminus L(\mathcal{S}_2),\ may(\mathcal{S}_1 \vee \mathcal{S}_2)(u) = may(\mathcal{S}_1), must(\mathcal{S}_1 \vee \mathcal{S}_2)(u) \\ \qquad\qquad\qquad\qquad = must(\mathcal{S}_1)(u) \\ \text{if } u \in L(\mathcal{S}_2) \setminus L(\mathcal{S}_1),\ may(\mathcal{S}_1 \vee \mathcal{S}_2)(u) = may(\mathcal{S}_2), must(\mathcal{S}_1 \vee \mathcal{S}_2)(u) \\ \qquad\qquad\qquad\qquad = must(\mathcal{S}_2)(u). \end{cases}$$

Proposition 3.9 *For all modal specification $\mathcal{S} \in \mathcal{MS}(\Sigma)$, we have:*

$$\mathcal{S} = \bigvee \{\mathcal{C} \mid \mathcal{C} \models \mathcal{S}\}$$

4 Residual specification for behavioral reuse

In the sequel, a component is a pair $(\mathcal{C}, \mathcal{S})$ such that $\mathcal{C} \models \mathcal{S}$ with $\mathcal{C}$ called the implementation and $\mathcal{S}$ the specification of the component. Reusing a component $(\mathcal{C}_1, \mathcal{S}_1)$ to realize a global system specified by $\mathcal{S}$ amounts to exhibit a residual specification $\mathcal{S}/\mathcal{S}_1$ so that any component $(\mathcal{C}_2, \mathcal{S}/\mathcal{S}_1)$ is such that the composition of $\mathcal{C}_1$ with $\mathcal{C}_2$ (noted $\mathcal{C}_1 \otimes \mathcal{C}_2$) satisfies $\mathcal{S}$. In component-based design, components are regarded as black-box. As a result, the implementation $\mathcal{C}_1$ of the component to be reused is unknown and its composition with the possible components $(\mathcal{C}_2, \mathcal{S}/\mathcal{S}_1)$ must realize $\mathcal{S}$ whatever the implementation $\mathcal{C}_1$ of $\mathcal{S}_1$ could be. Thus the characteristic property of a residual operation for behavioral reuse of a component is the following:

Proposition 4.1 $\mathcal{C}_2 \models \mathcal{S}/\mathcal{S}_1 \Leftrightarrow \forall \mathcal{C}_1.[\mathcal{C}_1 \models \mathcal{S}_1 \Rightarrow \mathcal{C}_1 \otimes \mathcal{C}_2 \models \mathcal{S}]$.

In the next section, we establish this proposition when $\mathcal{S}$ and $\mathcal{S}_1$ are modal specifications and when $\mathcal{C}_1$ and $\mathcal{C}_2$ are composed using a synchronous product.

5 Quotient of modal specifications for behavioral reuse

The synchronous product of languages $\mathcal{C}_1$ and $\mathcal{C}_2$ over the same alphabet Σ corresponds to the set theoretic intersection $\mathcal{C}_1 \cap \mathcal{C}_2$. We generalize the synchronous product to modal specifications:

Definition 5.1 The synchronous product of $\mathcal{S}_1$ and $\mathcal{S}_2$ is the modal specification $\mathcal{S}_1 \otimes \mathcal{S}_2$ over $L(\mathcal{S}_1) \cap L(\mathcal{S}_2)$ with:

$$\forall u \in L(\mathcal{S}_1) \cap L(\mathcal{S}_2), \begin{cases} must(\mathcal{S}_1 \otimes \mathcal{S}_2)(u) = must(\mathcal{S}_1)(u) \cap must(\mathcal{S}_2)(u) \\ may(\mathcal{S}_1 \otimes \mathcal{S}_2)(u) \ = may(\mathcal{S}_1)(u) \ \cap may(\mathcal{S}_2)(u) \end{cases}$$

Remark 5.2 (i) This operator is monotonic over the order relation $\leq$:

$$\mathcal{S}_1 \leq \mathcal{S}_2 \Rightarrow (\mathcal{S} \otimes \mathcal{S}_1 \leq \mathcal{S} \otimes \mathcal{S}_2 \text{ and } \mathcal{S}_1 \otimes \mathcal{S} \leq \mathcal{S}_2 \otimes \mathcal{S})$$

(ii) If $\mathcal{C}_1, \mathcal{C}_2 \subseteq \Sigma^*$ are viewed as modal specifications (cf. remark 3.5) then $\mathcal{C}_1 \otimes \mathcal{C}_2 = \mathcal{C}_1 \cap \mathcal{C}_2$.

Now, to define the quotient operation, we start from the observation that, for any prefix-closed languages L, M, and N: $L \cap M \subseteq N \Leftrightarrow L \subseteq \Downarrow (N \cup \neg M)$ where:

$$\Downarrow X = \{u \in \Sigma^* \mid \forall v \quad v \preceq u \Rightarrow v \in X\}$$

denote the prefix interior of a set X; it is an interior operation giving the greatest prefix-closed subset of the given set, when such a subset exists and the empty set otherwise. This remark is used to define the support of the modal specification $\mathcal{S}/\mathcal{S}_1$. Now in order to define the typing functions $may(\mathcal{S}/\mathcal{S}_1)$ and $must(\mathcal{S}/\mathcal{S}_1)$, we proceed by case inspection:

Definition 5.3 The quotient of the modal specifications $\mathcal{S}$ and $\mathcal{S}_1$ is the pseudo-modal specification $\mathcal{S}/\mathcal{S}_1$ over $\Downarrow (L(\mathcal{S}) \cup \neg L(\mathcal{S}_1))$ with:

(i) for all $u \in L(\mathcal{S}/\mathcal{S}_1) \cap L(\mathcal{S}_1)$:
  (a) if $a \in must(\mathcal{S})(u) \cap must(\mathcal{S}_1)(u)$ then $a \in must(\mathcal{S}/\mathcal{S}_1)(u)$ and $a \in may(\mathcal{S}/\mathcal{S}_1)(u)$;
  (b) if $a \in must(\mathcal{S})(u) \cap \neg must(\mathcal{S}_1)(u)$ then $a \in must(\mathcal{S}/\mathcal{S}_1)(u)$ but $a \notin may(\mathcal{S}/\mathcal{S}_1)(u)$;
  (c) if $a \in (may(\mathcal{S})(u) \setminus must(\mathcal{S})(u))$ then $a \in may(\mathcal{S}/\mathcal{S}_1)(u)$;
  (d) if $a \in mustnot(\mathcal{S})(u) \cap mustnot(\mathcal{S}_1)(u)$ then $a \in may(\mathcal{S}/\mathcal{S}_1)(u)$;
  (e) if $a \in mustnot(\mathcal{S})(u) \cap may(\mathcal{S}_1)(u)$ then $a \in mustnot(\mathcal{S}/\mathcal{S}_1)(u)$;

(ii) if $u \in (L(\mathcal{S}/\mathcal{S}_1) \setminus L(\mathcal{S}_1))$ then $must(\mathcal{S}/\mathcal{S}_1)(u) = \emptyset$ and $may(\mathcal{S}/\mathcal{S}_1)(u) = \Sigma$.

Now we give for each possible case, an intuitive interpretation of the resulting modality assuming that $\mathcal{C}_1 \models \mathcal{S}_1$ and we intend to have $\mathcal{C}_2 \models \mathcal{S}/\mathcal{S}_1$ and $\mathcal{C}_1 \otimes \mathcal{C}_2 \models \mathcal{S}$:

(i) first, when $u \in L(\mathcal{S}/\mathcal{S}_1) \cap L(\mathcal{S}_1)$:
  (a) a is required in the global specification $\mathcal{S}$ that is $u.a$ must belong to $\mathcal{C}_1 \otimes \mathcal{C}_2 = \mathcal{C}_1 \cap \mathcal{C}_2$. As a is guaranteed in $\mathcal{S}_1$, $u.a \in \mathcal{C}_1$ for all $\mathcal{C}_1 \models \mathcal{S}_1$ with $u \in \mathcal{C}_1$; thus $u.a$ must belong to $\mathcal{C}_2$ to always have $u.a \in \mathcal{C}_1 \otimes \mathcal{C}_2$: $a \in must(\mathcal{S}/\mathcal{S}_1)(u)$ and $a \in may(\mathcal{S}/\mathcal{S}_1)(u)$.

(b) a is required in the global specification $\mathcal{S}$ but as $a \notin must(\mathcal{S}_1)(u)$, there are some $\mathcal{C}_1 \models \mathcal{S}_1$ such that $u.a \notin \mathcal{C}_1$; hence, for all $\mathcal{C}_2$, $\mathcal{C}_1 \otimes \mathcal{C}_2$ can't be a model of $\mathcal{S}$. As a result, the trace u must be incoherently in $\mathcal{S}/\mathcal{S}_1$ and we let $a \in must(\mathcal{S}/\mathcal{S}_1)(u)$ but $a \notin may(\mathcal{S}/\mathcal{S}_1)(u)$ to model this inconsistency.

(c) a is allowed in the global specification $\mathcal{S}$ and $u.a$ may belong to $\mathcal{C}_1 \otimes \mathcal{C}_2$. Thus, whether or not $u.a$ belongs to $\mathcal{C}_1 \models \mathcal{S}_1$, $u.a$ can belong to $\mathcal{C}_2$ without violating the specification $\mathcal{S}$. Hence: $a \in may(\mathcal{S}/\mathcal{S}_1)(u)$.

(d) a is forbidden in the global specification $\mathcal{S}$ and in $\mathcal{S}_1$ thus, whether or not $u.a \in \mathcal{C}_2$, we have $u.a \notin \mathcal{C}_1 \otimes \mathcal{C}_2$ which is conform to $\mathcal{S}$. Hence: $a \in may(\mathcal{S}/\mathcal{S}_1)(u)$.

(e) a is forbidden in the global specification $\mathcal{S}$. As there are some $\mathcal{C}_1 \models \mathcal{S}_1$ with $u.a \in \mathcal{C}_1$, we forbid a in $\mathcal{S}/\mathcal{S}_1$: $a \in mustnot(\mathcal{S}/\mathcal{S}_1)(u)$. As a result, when $\mathcal{C}_2 \models \mathcal{S}/\mathcal{S}_1$, $u.a \notin \mathcal{C}_1 \otimes \mathcal{C}_2$ which is conform to $\mathcal{S}$.

(ii) if $u \in (L(\mathcal{S}/\mathcal{S}_1) \setminus L(\mathcal{S}_1))$: as $u \notin \mathcal{C}_1$, u may belong to $\mathcal{C}_2$, it won't belong to $\mathcal{C}_1 \otimes \mathcal{C}_2$. As a result, $\mathcal{S}/\mathcal{S}_1$ is relaxed after the trace u by taking $must(\mathcal{S}/\mathcal{S}_1)(u) = \emptyset$ (nothing is required) and $may(\mathcal{S}/\mathcal{S}_1)(u) = \Sigma$ (every event is allowed).

The adjoint operation of this quotient operation is the synchronous product of definition 5.1:

Proposition 5.4 *If $\mathcal{S}$, $\mathcal{S}_1$ and $\mathcal{S}_2$ are modal specifications over Σ then:*

$$\mathcal{S}_1 \otimes \mathcal{S}_2 \leq \mathcal{S} \Leftrightarrow \mathcal{S}_2 \leq \rho(\mathcal{S}/\mathcal{S}_1)$$

To prove proposition 4.1 for modal specifications and synchronous product, we need the following lemma:

Lemma 5.5 $\vee\{\mathcal{C} \otimes \mathcal{C}' \mid \mathcal{C} \models \mathcal{S}\} = \vee\{\mathcal{C} \mid \mathcal{C} \models \mathcal{S}\} \otimes \mathcal{C}'$

Proposition 5.6 *If $\mathcal{S}$ and $\mathcal{S}_1$ are modal specifications over Σ then:*

$$\mathcal{C}_2 \models \mathcal{S}/\mathcal{S}_1 \text{ iff } \forall \mathcal{C}_1.[\mathcal{C}_1 \models \mathcal{S}_1 \Rightarrow \mathcal{C}_1 \otimes \mathcal{C}_2 \models \mathcal{S}]$$

Proof. ($\Rightarrow$) According to Prop. 5.4, if $\mathcal{C}_2 \models \mathcal{S}/\mathcal{S}_1$ (that is $\mathcal{C}_2 \models \rho(\mathcal{S}/\mathcal{S}_1)$) then $\mathcal{C}_2 \otimes \mathcal{S}_1 \leq \mathcal{S}$. Moreover as $\mathcal{C}_1 \models \mathcal{S}_1$ then $\mathcal{C}_1 \otimes \mathcal{C}_2 \leq \mathcal{S}_1 \otimes \mathcal{C}_2$. As a result, $\mathcal{C}_1 \otimes \mathcal{C}_2 \leq \mathcal{S}$ that is $\mathcal{C}_1 \otimes \mathcal{C}_2 \models \mathcal{S}$.

($\Leftarrow$) If for all $\mathcal{C}_1$ such that $\mathcal{C}_1 \models \mathcal{S}_1$ we have $\mathcal{C}_1 \otimes \mathcal{C}_2 \models \mathcal{S}$ then:

$$\vee\{\mathcal{C}_1 \otimes \mathcal{C}_2 \mid \mathcal{C}_1 \models \mathcal{S}_1\} \leq \mathcal{S}$$

Thus, by lemma 5.5, $\vee\{\mathcal{C}_1 \mid \mathcal{C}_1 \models \mathcal{S}_1\} \otimes \mathcal{C}_2 \leq \mathcal{S}$ i.e. $\mathcal{S}_1 \otimes \mathcal{C}_2 \leq \mathcal{S}$ (by Prop. 3.9). According to Prop. 5.4, $\mathcal{C}_2 \leq \mathcal{S}/\mathcal{S}_1$ hence $\mathcal{C}_2 \models \mathcal{S}/\mathcal{S}_1$. □

As previously pointed out, the disjunction is not included in the logical fragment equivalent to modal automata. Therefore particular liveness properties can't be stated in this framework. For instance, let us consider the following "progressive" property: "any stimulus a is followed by *at least* b_1 or b_2 as reaction". This can't be specified with a modal specification: b_1 and b_2 can't belong to $must(\text{"}a\text{"})$ because this would request that every stimulus a is followed by both b_1 and b_2; b_1 and b_2 can't also belong to $may(\text{"}a\text{"})$ because the language such that the stimulus a is followed by no reaction would be a model of the specification.

A trace u in a modal specification specifies any situation where the system is ready to engage in a set of events X, if and only when $must(u) \subseteq X \subseteq may(u)$. This set of "acceptance" set is thus given by:

$$Acc(u) = \{X \in \mathcal{P}(\Sigma) \mid must(u) \subseteq X \subseteq may(u)\}$$

By definition this set is closed under union, intersection and convexity (that is given $X, Y \in Acc(u)$ and a set Z such that $X \subseteq Z \subseteq Y$ then $X \cup Y$, $X \cap Y$ and $Z \in Acc(u)$) and may and must modalities may be recovered as $may(u) = \bigcup_{X \in Acc(u)} X$ and $must(u) = \bigcap_{X \in Acc(u)} X$.

Thus, for example if $may(u) = \{b_1, b_2\}$ and $must(u) = \emptyset$, we obtain $Acc(u) = \{\emptyset, \{b_1\}, \{b_2\}, \{b_1, b_2\}\}$. If we want to specify that at least b_1 or b_2 occur, the specified acceptance set should be $\{\{b_1\}, \{b_2\}, \{b_1, b_2\}\}$ which is no longer closed by intersection. According to this example, closure by intersection should be relaxed to deal with such "progressive" properties. Trees labeled by acceptance set closed by union and convexity have been studied in [7]. In the next section, we propose a quotient operation for acceptance specifications with no closure constraint over the acceptance set.

6 Improving expressivity with acceptance spec

We generalize the previous framework presented for modal specifications to acceptance specifications:

6.1 Acceptance specifications and their models

Definition 6.1 An acceptance specification $\mathcal{S}$ is a pair $\mathcal{S} = \langle L, Acc \rangle$ where L is a prefix-closed language over Σ and $Acc : L \to \mathcal{P}(\mathcal{P}(\Sigma))$ is a map associating each trace $u \in L$ to its acceptance set. For consistency, we require for all trace $u \in L$:

(C1) $Acc(u) \neq \emptyset$

(C2) $u.a \in L$ if and only there exists at least one set $X \in Acc(u)$ such that $a \in X$

The condition (C2) can be rephrase in $L_u = \bigcup_{X \in Acc(u)} X$. The set of acceptance specifications over Σ is denoted $\mathcal{AS}(\Sigma)$.

Definition 6.2 A prefix-closed language $C \subseteq \Sigma^*$ is a model of the acceptance specification $\mathcal{S} \in \mathcal{AS}(\Sigma)$, noted $C \models \mathcal{S}$, if:

- $C \subseteq L(\mathcal{S})$;
- for all $u \in C$, $C_u \in Acc(\mathcal{S})(u)$.

Example 6.3 The specification $\top = \langle \Sigma^*, Acc \rangle$ with, for all $u \in \Sigma^*$, $Acc(u) = \mathcal{P}(\Sigma)$, admit every language over Σ as model. $\bot = \langle \emptyset, \emptyset \rangle$ has no model.

Remark 6.4 $Acc(u) = \emptyset$ is different from $\emptyset \in Acc(u)$. The first situation is a violation of a consistency condition whereas the second reports that some models of the specification can perform no event after the trace u.

Definition 6.5 The order relation on acceptance specifications is given by inclusion of both corresponding languages and acceptance sets:

$$\mathcal{S}_1 \leq \mathcal{S}_2 \text{ iff } L(\mathcal{S}_1) \subseteq L(\mathcal{S}_2) \text{ and } \forall u \in L(\mathcal{S}_1), \quad Acc(\mathcal{S}_1)(u) \subseteq Acc(\mathcal{S}_2)(u)$$

Remark 6.6 Any language C can be viewed as an acceptance specification with $Acc(u) = C_u$ that is its acceptance set is a singleton for all trace $u \in C$. Hence $C \models \mathcal{S}$ if and only if $C \leq \mathcal{S}$.

6.2 Pseudo-acceptance specifications

Definition 6.7 A pseudo-acceptance specification $p\mathcal{S} \in p\mathcal{AS}(\Sigma)$ is a pair $\langle L, Acc \rangle$ where L is a language over Σ and $Acc : L \to \mathcal{P}(\mathcal{P}(\Sigma))$ is a map associating each trace u to its set of acceptance with *no* consistency constraint over Acc.
A trace u of $p\mathcal{S}$ is said incoherently specified when $Acc(u) = \emptyset$.

Every pseudo-acceptance specification $p\mathcal{S}$ can be reduced to an acceptance specification $\rho(p\mathcal{S})$ that admits the same models, by iteration of the following steps:

(i) basis: we let R be a copy of $p\mathcal{S}$;

(ii) we let U be the set of traces u incoherently specified in R; we remove U from $L(R)$; for all trace v such that $v.a = u$ with $u \in U$, we remove from $Acc(R)(v)$ all sets containing the letter a to enforce the consistency condition $(C2)$;

When $a \in \bigcap_{X \in Acc(R)(v)} X$, v becomes incoherently specified in R. Thus, we repeat this step until there is no more incoherently specified trace in $L(R)$;

(iii) $\rho(p\mathcal{S})$ is built from R: $L(\rho(p\mathcal{S})) = \widetilde{L(R)}$ and for all $u \in L(\rho(p\mathcal{S}))$, $Acc(\rho(p\mathcal{S}))(u) = Acc(R)(u)$.

6.3 Lattice of acceptance specifications

The set of acceptance specifications $\mathcal{AS}$ equipped with the partial order $\leq$ is a complete distributive lattice (hence a bounded lattice) where the meet $\mathcal{S}_1 \wedge \mathcal{S}_2$ is the reduction of the pseudo-acceptance specification $\mathcal{S}_1 \& \mathcal{S}_2$ whose support is $L(\mathcal{S}_1) \cap L(\mathcal{S}_2)$ and with $Acc(\mathcal{S}_1 \& \mathcal{S}_2)(u) = Acc(\mathcal{S}_1)(u) \cap Acc(\mathcal{S}_2)(u)$. The join $\mathcal{S}_1 \vee \mathcal{S}_2$ is defined over $L(\mathcal{S}_1) \cup L(\mathcal{S}_2)$ by:

$$\begin{cases} \text{if } u \in L(\mathcal{S}_1) \cap L(\mathcal{S}_2),\ Acc(\mathcal{S}_1 \vee \mathcal{S}_2)(u) = Acc(\mathcal{S}_1) \cup Acc(\mathcal{S}_2) \\ \text{if } u \in L(\mathcal{S}_1) \setminus L(\mathcal{S}_2),\ Acc(\mathcal{S}_1 \vee \mathcal{S}_2)(u) = Acc(\mathcal{S}_1) \\ \text{if } u \in L(\mathcal{S}_2) \setminus L(\mathcal{S}_1),\ Acc(\mathcal{S}_1 \vee \mathcal{S}_2)(u) = Acc(\mathcal{S}_2) \end{cases}$$

6.4 Quotient of acceptance specifications

We define quotient of acceptance specifications such that proposition 4.1 is verified with synchronous product as component composition:

Definition 6.8 The synchronous product of the acceptance specifications $\mathcal{S}_1$ and $\mathcal{S}_2$ is the acceptance specification $\mathcal{S}_1 \otimes \mathcal{S}_2$ over $L(\mathcal{S}_1) \cap L(\mathcal{S}_2)$ with for all $u \in L(\mathcal{S}_1) \cap L(\mathcal{S}_2)$:

$$Acc(\mathcal{S}_1 \otimes \mathcal{S}_2)(u) = \{X_1 \cap X_2 \mid X_1 \in Acc(\mathcal{S}_1)(u) \text{ and } X_2 \in Acc(\mathcal{S}_2)(u)\}$$

This operation has for adjoint the following quotient operation:

Definition 6.9 The quotient of the acceptance specification $\mathcal{S}$ and $\mathcal{S}_1$ is the pseudo-acceptance specification $\mathcal{S}/\mathcal{S}_1$ over $\Downarrow(L(\mathcal{S}) \cup \neg L(\mathcal{S}_1))$ with, for all $u \in L(\mathcal{S}/\mathcal{S}_1) \cap L(\mathcal{S}_1)$: $Acc(\mathcal{S}/\mathcal{S}_1)(u) = \{Y \in \mathcal{P}(\Sigma) \mid \forall X \in Acc(\mathcal{S}_1)(u), X \cap Y \in Acc(\mathcal{S})(u)\}$
and for all $u \in (L(\mathcal{S}/\mathcal{S}_1) \setminus L(\mathcal{S}_1))$, $Acc(\mathcal{S}/\mathcal{S}_1)(u) = \mathcal{P}(\Sigma)$.

Proposition 6.10 *If $\mathcal{S}$, $\mathcal{S}_1$ and $\mathcal{S}_2$ are acceptance specifications over Σ then:*

$$\mathcal{S}_1 \otimes \mathcal{S}_2 \leq \mathcal{S} \Leftrightarrow \mathcal{S}_2 \leq \rho(\mathcal{S}/\mathcal{S}_1)$$

Similarly to the proof for modal specifications, we use the previous result to establish the characteristic property of a residual operation for behavioral reuse of a component:

Proposition 6.11 *If $\mathcal{S}$ and $\mathcal{S}_1$ are acceptance specifications then:*

$$\mathcal{C}_2 \models \mathcal{S}/\mathcal{S}_1 \textit{ iff } \forall \mathcal{C}_1.[\mathcal{C}_1 \models \mathcal{S}_1 \Rightarrow \mathcal{C}_1 \otimes \mathcal{C}_2 \models \mathcal{S}]$$

As previously briefly noticed, acceptance specifications strictly subsumes modal specifications. Indeed, consider the two following transformations:

Definition 6.12 Let $\mathcal{S} = (L, must, may) \in \mathcal{MS}$ and $\mathcal{S}' = (L', Acc') \in \mathcal{AS}$:

- $j : \mathcal{MS} \to \mathcal{AS}$
  $j(\mathcal{S}) = (L, Acc)$ with $Acc(u) = \{X \in \mathcal{P}(\Sigma) \mid must(u) \subseteq X \subseteq may(u)\}$
- $k : \mathcal{AS} \to \mathcal{MS}$

$$k(\mathcal{S}') = (L', must', may') \text{ with } \begin{cases} may'(u') & = \bigcup_{X \in Acc'(u')} X \\ must'(u') & = \bigcap_{X \in Acc'(u')} X \end{cases}$$

We have: $k \circ j = Id$ but $j \circ k \neq Id$. Quotient operations for modal specifications and acceptance specifications can be related:

Proposition 6.13 *The quotient operation for modal specifications is a particularization of the quotient operation for acceptance specifications.*

Proof. Given $\mathcal{S}$ and $\mathcal{S}_1$ two modal specifications, we let $\mathcal{S}'$ be the acceptance specifications obtained by quotienting $j(\mathcal{S})$ and $j(\mathcal{S}_1)$ using definition 6.9. Then the modal specification $k(\mathcal{S}')$ is identical to the one obtained by quotienting $\mathcal{S}$ and $\mathcal{S}_1$ using definition 5.3. □

Synchronous product is a very restrictive form of composition. In proposition 6.11, the component $(\mathcal{C}_2, \mathcal{S}/\mathcal{S}_1)$ restrict the behavior of all possible $(\mathcal{C}_1, \mathcal{S}_1)$ so that the composite system $\mathcal{C}_1 \otimes \mathcal{C}_2$ realizes $\mathcal{S}$. In the next section, we investigate an approach where the component $(\mathcal{C}_2, \mathcal{S}/\mathcal{S}_1)$ may also contribute directly to the realization of the specification $\mathcal{S}$: given $\mathcal{S}$ a global specification over Σ and $(\mathcal{C}_1, \mathcal{S}_1)$ a component to be reused over Σ_1, the events belonging to $\Sigma \setminus \Sigma_1$ are realized by the component $(\mathcal{C}_2, \mathcal{S}/\mathcal{S}_1)$. Thus, we now consider residual of specifications when component composition corresponds to mixed product [4]. We also consider internalization of event that is $\mathcal{C}_1$ and $\mathcal{C}_2$ may evolved without being observed externally.

7 Using mixed product with internalization of events as component composition

We first recall the definition of mixed product, restriction and expansion of languages:

Definition 7.1 • Given $\mathcal{C}_1$ and $\mathcal{C}_2$ two languages respectively over Σ_1 and Σ_2, the mixed product of $\mathcal{C}_1$ and $\mathcal{C}_2$ is the language:

$$\mathcal{C}_1 \sqcap \mathcal{C}_2 = \{w \in (\Sigma_1 \cup \Sigma_2)^* \mid \Pi_{\Sigma_1}(w) \in \mathcal{C}_1 \text{ and } \Pi_{\Sigma_2}(w) \in \mathcal{C}_2\}$$

- Given $\mathcal{C}$ a language over Σ' and $\Sigma \subseteq \Sigma'$, the restriction of $\mathcal{C}$ to the alphabet Σ is the language:

$$\mathcal{C}_{\downarrow\Sigma} = \{u \in \Sigma^* \mid u = \Pi_\Sigma(w) \text{ with } w \in \mathcal{C}\}$$

- Given $\mathcal{C}$ a language over Σ and $\Sigma \subseteq \Sigma'$, the expansion of $\mathcal{C}$ to the alphabet Σ' is the language:

$$\mathcal{C}_{\uparrow\Sigma'} = \{w \in \Sigma'^* \mid \Pi_\Sigma(w) \in \mathcal{C}\}$$

Now, we generalize these operations for acceptance specifications:

7.1 Mixed product of acceptance specifications

Definition 7.2 The mixed product of the acceptance specifications $\mathcal{S}_1 \in \mathcal{AS}(\Sigma_1)$ and $\mathcal{S}_2 \in \mathcal{AS}(\Sigma_2)$ is the acceptance specification $\mathcal{S}_1 \sqcap \mathcal{S}_2$ over $L(\mathcal{S}_1) \sqcap L(\mathcal{S}_2)$ with for all $u \in L(\mathcal{S}_1) \sqcap L(\mathcal{S}_2)$:

$$\begin{aligned} Acc(\mathcal{S}_1 \sqcap \mathcal{S}_2)(w) = \{&(X_1 \cup (\Sigma_2 \setminus \Sigma_1)) \cap (X_2 \cup (\Sigma_1 \setminus \Sigma_2)) \mid \\ &X_1 \in Acc(\mathcal{S}_1)(\Pi_{\Sigma_1}(w)) \text{ and } X_2 \in Acc(\mathcal{S}_2)(\Pi_{\Sigma_2}(w))\} \end{aligned}$$

When $\Sigma_1 = \Sigma_2$ the definition of the synchronous product is retrieved.

The mixed product of acceptance specifications can be related in a general way to the synchronous product:

Definition 7.3 Given $\Sigma \subseteq \Sigma'$ and $\mathcal{S} \in \mathcal{AS}(\Sigma)$, the τ-expansion of $\mathcal{S}$ to Σ' is the acceptance specification $\tau_{\Sigma'}(\mathcal{S})$ over $(L(\mathcal{S}))_{\uparrow\Sigma'}$ with:

$$Acc(\tau_{\Sigma'}(\mathcal{S}))(w) = \{Y \cup (\Sigma' \setminus \Sigma) \mid Y \in Acc(\mathcal{S})(\Pi_\Sigma(w))\}$$

This operation consists in saturating the element of each acceptance set with all events of $(\Sigma' \setminus \Sigma)$. Thus, mixed product of acceptance specifications is reduced to synchronous product (cf. definition 6.8):

Proposition 7.4 *Given $\mathcal{S}_1 \in \mathcal{AS}(\Sigma_1)$ and $\mathcal{S}_2 \in \mathcal{AS}(\Sigma_2)$:*

$$\mathcal{S}_1 \sqcap \mathcal{S}_2 = \tau_{\Sigma_1 \cup \Sigma_2}(\mathcal{S}_1) \otimes \tau_{\Sigma_1 \cup \Sigma_2}(\mathcal{S}_2)$$

The restriction operation adjoint of the τ-expansion is the following:

Definition 7.5 Given $\Sigma \subseteq \Sigma'$ and $\mathcal{S}' \in \mathcal{AS}(\Sigma')$, the Π-restriction of $\mathcal{S}'$ to Σ is the acceptance specification $\Pi_\Sigma(\mathcal{S}')$ over $(L(\mathcal{S}))_{\downarrow\Sigma}$ with:

$$\begin{aligned} Acc(\Pi_\Sigma(\mathcal{S}'))(u) = \{Y \mid Y \cup (\Sigma' \setminus \Sigma) \in \bigcap \{&Acc(\mathcal{S}')(w) \mid w \in L(\mathcal{S}') \\ &\text{and } \Pi_\Sigma(w) = u\}\} \end{aligned}$$

Proposition 7.6 *Given* $\mathcal{S} \in \mathcal{AS}(\Sigma)$ *and* $\mathcal{S}' \in \mathcal{AS}(\Sigma')$ *with* $\Sigma \subseteq \Sigma'$,

$$\tau_{\Sigma'}(\mathcal{S}) \leq \mathcal{S}' \Leftrightarrow \mathcal{S} \leq \Pi_{\Sigma}(\mathcal{S}')$$

To deal with internalization of event, we now define the restriction of an acceptance specification to a sub-alphabet:

7.2 Restriction of acceptance specification

Definition 7.7 Given $\Sigma \subseteq \Sigma'$ and $\mathcal{S} \in \mathcal{AS}(\Sigma')$, the restriction of $\mathcal{S}$ to Σ is the acceptance specification $\mathcal{S}_{\downarrow\Sigma}$ over $(L(\mathcal{S}))_{\downarrow\Sigma}$ with:

$$Acc(\mathcal{S}_{\downarrow\Sigma})(u) = \bigcup \{ \{ X \cap \Sigma \mid X \in Acc(\mathcal{S})(w) \text{ and } X \cap \Sigma \neq \emptyset \text{ when } X \neq \emptyset \} \mid w \in L(\mathcal{S}) \text{ and } \Pi_{\Sigma}(w) = u \}$$

Definition 7.8 Given $\Sigma \subseteq \Sigma'$ and $\mathcal{S}' \in \mathcal{AS}(\Sigma)$, the expansion of $\mathcal{S}'$ to Σ' is the acceptance specification $\mathcal{S}'_{\uparrow\Sigma'}$ over $(L(\mathcal{S}'))_{\uparrow\Sigma'}$ with:

$$\begin{aligned} Acc(\mathcal{S}'_{\uparrow\Sigma'})(w) \quad = \quad & \{ X \mid X \cap \Sigma \in Acc(\mathcal{S}')(\Pi_{\Sigma}(w)) \} \cup \\ & \{ X \mid X \neq \emptyset \text{ and } X \subseteq (\Sigma' \setminus \Sigma) \} \end{aligned}$$

This operation consists in allowing after each trace u of $\mathcal{S}'$ finite sequences of events in $\Sigma' \setminus \Sigma$.

Proposition 7.9 *Given* $\mathcal{S} \in \mathcal{AS}(\Sigma')$ *and* $\mathcal{S}' \in \mathcal{AS}(\Sigma)$ *with* $\Sigma \subseteq \Sigma'$,

$$\mathcal{S}_{\downarrow\Sigma} \leq \mathcal{S}' \Leftrightarrow \mathcal{S} \leq \mathcal{S}'_{\uparrow\Sigma'}$$

7.3 Adjoint of the mixed product with internalization of events of acceptance specification

From the previous propositions, we can deduce:

$$\begin{aligned} (\mathcal{S}_1 \sqcap\!\sqcap \mathcal{S}_2)_{\downarrow\Sigma} \leq \mathcal{S} \quad & \Leftrightarrow \quad \mathcal{S}_1 \sqcap\!\sqcap \mathcal{S}_2 \leq \mathcal{S}_{\uparrow(\Sigma_1 \cup \Sigma_2)} && \text{by Prop. 7.9} \\ & \Leftrightarrow \quad \tau_{\Sigma_1 \cup \Sigma_2}(\mathcal{S}_1) \otimes \tau_{\Sigma_1 \cup \Sigma_2}(\mathcal{S}_2) \leq \mathcal{S}_{\uparrow(\Sigma_1 \cup \Sigma_2)} && \text{by Prop. 7.4} \\ & \Leftrightarrow \quad \tau_{\Sigma_1 \cup \Sigma_2}(\mathcal{S}_2) \leq \mathcal{S}_{\uparrow(\Sigma_1 \cup \Sigma_2)} / \tau_{\Sigma_1 \cup \Sigma_2}(\mathcal{S}_1) && \text{by Prop. 6.10} \\ & \Leftrightarrow \quad \mathcal{S}_2 \leq \Pi_{\Sigma_2}(\mathcal{S}_{\uparrow(\Sigma_1 \cup \Sigma_2)} / \tau_{\Sigma_1 \cup \Sigma_2}(\mathcal{S}_1)) && \text{by Prop. 7.6} \end{aligned}$$

Similarly to the proof for modal specifications in the synchronous case, we use the previous equivalence to establish the characteristic property of a residual operation for behavioral reuse of a component when the product of components is the mixed product with internalization of events:

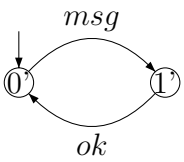
Proposition 7.10 *Given* $\mathcal{S} \in \mathcal{AS}(\Sigma)$, $\mathcal{S}_1 \in \mathcal{AS}(\Sigma_1)$ *and* Σ_2 *such that* $\Sigma \subseteq \Sigma_1 \cup \Sigma_2$*:*

$$\forall \mathcal{C}_1.[\mathcal{C}_1 \models \mathcal{S}_1 \Rightarrow (\mathcal{C}_1 \sqcap \mathcal{C}_2)_{\downarrow\Sigma} \models \mathcal{S}] \;\Leftrightarrow\; \mathcal{C}_2 \models \amalg_{\Sigma_2}(\mathcal{S}_{\uparrow(\Sigma_1\cup\Sigma_2)} / \tau_{\Sigma_1\cup\Sigma_2}(\mathcal{S}_1))$$

8 An example

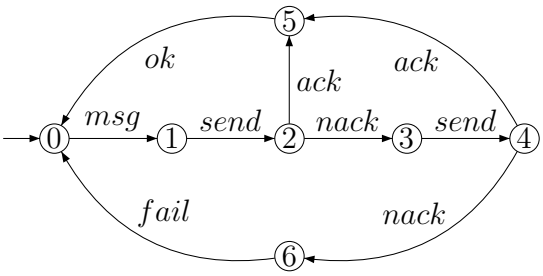
This example is inspired from [3]. In this paper, component interfaces are designed via interface automata. We refine the intended behavior thanks to *acceptance automata*; the quotient of acceptance specifications defined in 6.9 can be adapted when the support of the specification is a regular prefix-closed language [13].

- The goal is to build a system satisfying the following specification $\mathcal{S}$ over the alphabet $\Sigma = \{msg, ok, fail\}$:



	$Acc(S)$
0'	$\{\{msg\}\}$
1'	$\{\{ok\}\}$

- To realize $\mathcal{S}$, we aim at reusing a component satisfying the following specification $\mathcal{S}_1$ describing the behavior of a communication channel:

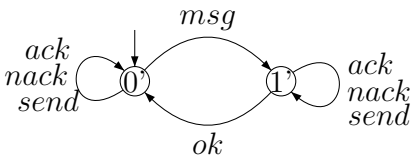


	$Acc(S_1)$
0	$\{\{msg\}\}$
1	$\{\{send\}\}$
2	$\{\{ack\}, \{nack\}, \{ack, nack\}\}$
3	$\{\{send\}\}$
4	$\{\{ack, nack\}\}$
5	$\{\{ok\}\}$
6	$\{\{fail\}\}$

  $\mathcal{S}_1$ is defined over the alphabet $\Sigma_1 = \{msg, send, ack, nack, fail, ok\}$. Note that loss of message is allowed in $Acc(\mathcal{S}_1)(2)$ which is not the case in $Acc(\mathcal{S}_1)(4)$.

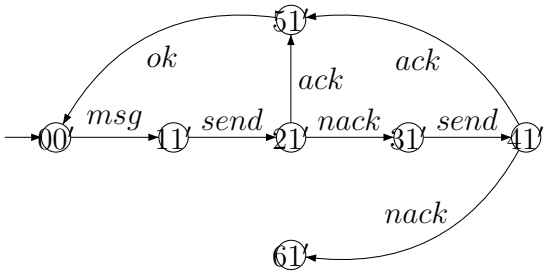
- We let $\Sigma_2 = \{ack, nack, send\}$. As $\Sigma \setminus \Sigma_1 = \emptyset$, $(\mathcal{C}_2, \mathcal{S}/\mathcal{S}_1)$ will restrict the behavior of $(\mathcal{C}_1, \mathcal{S}_1)$ to enforce $\mathcal{S}$. We have: $\tau_{\Sigma_1\cup\Sigma_2}(\mathcal{S}_1) = \mathcal{S}_1$ as $\Sigma_2 \subseteq \Sigma_1$.

- We compute $S_{\uparrow(\Sigma_1 \cup \Sigma_2)}$:



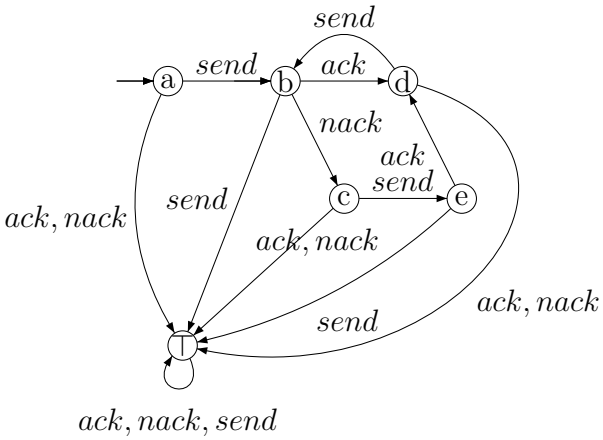
  - $Acc(S_{\uparrow(\Sigma_1 \cup \Sigma_2)})(0') = \{\{msg\}, \{msg\} \cup X, X\}$ with $X \subseteq \Sigma_2$ and $X \neq \emptyset$.
  - $Acc(S_{\uparrow(\Sigma_1 \cup \Sigma_2)})(1') = \{\{ok\}, \{ok\} \cup X, X\}$ with $X \subseteq \Sigma_2$ and $X \neq \emptyset$.

- Then we compute the quotient $\mathcal{S}_{\uparrow(\Sigma_1 \cup \Sigma_2)}/\tau_{\Sigma_1 \cup \Sigma_2}(\mathcal{S}_1)$ (in the following figure, only transitions labeled by required events are depicted):



  - $Acc(00') = \{\{msg\} \cup X \mid X \subseteq \{ok, fail, ack, nack, send\}\}$;
  - $Acc(11') = \{\{send\} \cup X \mid X \subseteq \{ok, fail, ack, nack, msg\}\}$
  - $Acc(21') = \{\{ack, nack\} \cup X \mid X \subseteq \{ok, fail, msg, send\}\}$
  - $Acc(31') = \{\{send\} \cup X \mid X \subseteq \{msg, ok, fail, ack, nack\}\}$
  - $Acc(41') = \{\{ack\} \cup X, \{nack\} \cup X, \{ack, nack\} \cup X \mid X \subseteq \{ok, fail, msg, send\}\}$
  - $Acc(51') = \{\{ok\} \cup X \mid X \subseteq \{msg, fail, ack, nack, send\}\}$
  - $Acc(61') = \emptyset$

- Last, we apply the operation $\amalg_{\Sigma_2}$ on the acceptance specification equivalent to the previous pseudo-acceptance specification. The result is the following:



- $Acc(a) = \{\{send\}, \{send, ack\}, \{send, nack\}, \{send, ack, nack\}\}$
- $Acc(b) = \{\{ack, nack\}, \{ack, nack, send\}\}$
- $Acc(c) = \{\{send\}, \{send, ack\}, \{send, nack\}, \{send, ack, nack\}\}$
- $Acc(d) = \{\{send\}, \{send, ack\}, \{send, nack\}, \{send, ack, nack\}\}$
- $Acc(e) = \{\{ack\}, \{ack, send\}\}$
- $Acc(\top) = \mathcal{P}(\Sigma_2)$.

We remark that a sent message that has been acknowledged negatively *must* then be acknowledged positively (in state e, *ack* is required and *nack* is forbidden).

9 Conclusion

In this paper, we have studied the problem of behavioral reuse of a component as the computation of a residual specification. We have introduced modal specification and acceptance specification as formalisms to specify component behavior. They allow to address restricted forms of liveness. Quotient of mu-calculus formulas was investigated in [1]. Mu-calculus is quite expressive but the complexity of the proposed quotient operation is double exponential in the size of the tree automata equivalent to the quotiented formulas. In contrast, our solutions using the automata-based version of modal and acceptance specifications are polynomial [13]. Furthermore, to our knowledge, our approach is the first to consider components as black box in equation solving: the equation is solved for a given set of possible implementations characterized by a specification $\mathcal{S}_1$.

Future works concern the application of these techniques to the component adaptation problem. In particular, these techniques seems suited when detection of mismatch between components is performed thanks to the description of the properties the system should verify [2]. Moreover modal

and acceptance specifications are sets equipped with a lattice structure and a monoid structure with a residual operation, adjoint of a commutative product operation ie. are residuated lattices. We are interested in a more precise characterization of the underlying algebraic structure of the sets of modal and acceptance specification in order to develop the basis of an algebraic theory of components adaptation and reuse.

Acknowledgement

The author would like to thank Eric Badouel and Philippe Darondeau for many suggestions and discussions.

References

[1] Arnold, A., A. Vincent and I. Walukiewicz, *Games for synthesis of controllers with partial observation*, Theoretical Computer Science **303** (2003), pp. 7–34.

[2] Canal, C., J. M. Murillo and P. Poizat, *Software adaptation*, L'Objet **12** (2006), pp. 9–31.

[3] de Alfaro, L. and T. A. Henzinger, *Interface automata*, in: *ESEC/FSE-9: Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering* (2001), pp. 109–120.

[4] Duboc, C., *Mixed product and asynchronous automata.*, Theor. Comput. Sci. **48** (1986), pp. 183–199.

[5] El-Fakih, K., N. Yevtushenko, S. Buffalov and G. von Bochmann, *Progressive solutions to a parallel automata equation.*, Theoretical Computer Science **362** (2006), pp. 17–32.

[6] Feuillade, G. and S. Pinchinat, *Modal specifications for the control theory of discrete event systems*, J. of Discrete Event Dynamic Systems **17** (2007).

[7] Hennessy, M., *Acceptance trees*, J. ACM **32** (1985), pp. 896–928.

[8] Kumar, R., S. Nelvagal and S. I. Marcus, *A discrete event systems approach for protocol conversion*, Discrete Event Dynamic Systems **7** (1997), pp. 295–315.

[9] Larsen, K. G., *Modal specifications*, in: *Automatic Verification Methods for Finite State Systems*, 1989, pp. 232–246.

[10] Larsen, K. G. and L. Xinxin, *Equation solving using modal transition systems*, in: *Proceedings of the Fifth Annual IEEE Symp. on Logic in Computer Science, LICS 1990* (1990), pp. 108–117.

[11] Merlin, P. and G. von Bochmann, *On the construction of submodule specifications and communication protocols*, ACM Trans. on Programming Languages and Systems **5** (1983), pp. 1–25.

[12] Parrow, J., *Submodule construction as equation solving in ccs*, Theoretical Computer Science **68** (1987), pp. 175–202.

[13] Raclet, J.-B., *Residual for component specifications*, Research Report 6196, INRIA (2007).

[14] Yevtushenko, N., T. Villa, R. K. Brayton, A. Petrenko and A. L. Sangiovanni-Vincentelli, *Solution of parallel language equations for logic synthesis*, in: *Proceedings of the International Conference on Computer-Aided Design*, 2001, pp. 103–110.