Residual for Component Specifications

Jean-Baptiste Raclet¹

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 $S_1$, be used to build a system satisfying a global specification $S$? If so, what are the components that, when composed with the reused component, constitute a composite system satisfying $S'$?

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

¹ Email: Jean-Baptiste.Raclet@irisa.fr
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 $S_1 \times X \simeq S$ with $S_1$ called the context, $S$ 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 $S/S_1$. 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 $\Sigma$ to denote the universe of events. $\Sigma^*$ denotes the set of all finite length event sequences, called traces, with $\epsilon$ the zero length trace. A language $L$ over $\Sigma$ is a subset of $\Sigma^*$. 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 $\Sigma$, 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 $\Sigma$ denoted $\Pi_{\Sigma}(w)$ is the trace obtained from $w$ by erasing the events not belonging to $\Sigma$. 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 $\bar{L}$ consists of all the prefixes of all the traces in $L$. A language $L$ is said prefix-closed when $L = \bar{L}$.

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
Raclet

now introduce modal specifications which generalize the use of modalities on events to non-necessarily regular languages:

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

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

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

1. $\text{must}(u) \subseteq \text{may}(u)$;
2. $\text{may}(u) = L_u$.

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 $S$ could be denoted $L(S)$, $\text{must}(S)$ and $\text{may}(S)$. The prefix-closed language $L(S)$ may be called the support of $S$. The set of modal specifications over $\Sigma$ 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 $S \in \mathcal{MS}(\Sigma)$, noted $C \models S$, if:

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

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

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

Modal specifications are ordered by the following relation:

**Definition 3.4** $S_1 \preceq S_2$ if and only if:

- $L(S_1) \subseteq L(S_2)$ and
- $\forall u \in L(S_1)$, $\begin{cases} \text{may}(S_1)(u) \subseteq \text{may}(S_2)(u) \\ \text{must}(S_1)(u) \supseteq \text{must}(S_2)(u) \end{cases}$

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


Proposition 3.6 \( S_1 \leq S_2 \) if and only if every model \( C \) of \( S_1 \) is also a model of \( S_2 \).

When \( L(S) \) is regular, \( S \) 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 mu-calculus as it includes conjunctions and greatest fix-points 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 \( pS \in p\mathcal{MS}(\Sigma) \) is a triple \( \langle L, \text{must}, \text{may} \rangle \) where \( L \) is a prefix-closed language over \( \Sigma \) and \( \text{must}, \text{may} : L \to \mathcal{P}(\Sigma) \) are partial functions that type events with no consistency constraint between \( \text{may}(u) \) and \( \text{must}(u) \).

A trace \( u \) of \( pS \) such that \( \text{must}(u) \notin \text{may}(u) \) 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 \( \text{must}(u) \notin \text{may}(u) \) then \( u \) can’t belong to a language \( C \) model of \( pS \) because this would imply, for \( a \in \text{must}(u) \) and \( a \notin \text{may}(u) \), on one hand that \( u.a \in C \) (as \( a \in \text{must}(u) \)) and, on the other hand, that \( u.a \notin C \) (as \( a \notin \text{may}(u) \)). 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 \( pS \) has no model or there exists a largest modal specification \( \rho(pS) \) smaller than \( pS \), and \( \rho(pS) \) has the same models as \( pS \).

The modal specification \( \rho(pS) \) is obtained from the pseudo-modal specification \( pS \) by application of the following steps:

(i) basis: we let \( R \) be a copy of \( pS \);

(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 \( \text{may}(R)(v) \) to enforce the consistency condition (C2):

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

(iii) \( \rho(pS) \) is built from \( R \): \( L(\rho(pS)) = \widehat{L(R)} \) and for all \( u \in L(\rho(pS)) \), \( \text{may}(\rho(pS))(u) = \text{may}(R)(u) \) and \( \text{must}(\rho(pS))(u) = \text{must}(R)(u) \).

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 \( S_1 \land S_2 \) is the reduction
of the pseudo-modal specification $S_1 \& S_2$ over $L(S_1) \cap L(S_2)$ with:

$$\forall u \in L(S_1) \cap L(S_2), \begin{cases} 
\text{may}(S_1 \& S_2)(u) = \text{may}(S_1)(u) \cap \text{may}(S_2)(u) \\
\text{must}(S_1 \& S_2)(u) = \text{must}(S_1)(u) \cup \text{must}(S_2)(u)
\end{cases}$$

and the join $S_1 \vee S_2$ is the modal specification over $L(S_1) \cup L(S_2)$ with:

$$\begin{cases} 
\text{if } u \in L(S_1) \cap L(S_2), \text{may}(S_1 \vee S_2)(u) = \text{may}(S_1)(u) \cup \text{may}(S_2)(u), \\
\text{must}(S_1 \vee S_2)(u) = \text{must}(S_1)(u) \cap \text{must}(S_2)(u) \\
\text{if } u \in L(S_1) \setminus L(S_2), \text{may}(S_1 \vee S_2)(u) = \text{may}(S_1), \text{must}(S_1 \vee S_2)(u) = \text{must}(S_1)(u) \\
\text{if } u \in L(S_2) \setminus L(S_1), \text{may}(S_1 \vee S_2)(u) = \text{may}(S_2), \text{must}(S_1 \vee S_2)(u) = \text{must}(S_2)(u).
\end{cases}$$

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

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

### 4 Residual specification for behavioral reuse

In the sequel, a component is a pair $(C, S)$ such that $C \models S$ with $C$ called the implementation and $S$ the specification of the component. Reusing a component $(C_1, S_1)$ to realize a global system specified by $S$ amounts to exhibit a residual specification $S/S_1$ so that any component $(C_2, S/S_1)$ is such that the composition of $C_1$ with $C_2$ (notated $C_1 \otimes C_2$) satisfies $S$. In component-based design, components are regarded as black-box. As a result, the implementation $C_1$ of the component to be reused is unknown and its composition with the possible components $(C_2, S/S_1)$ must realize $S$ whatever the implementation $C_1$ of $S_1$ could be. Thus the characteristic property of a residual operation for behavioral reuse of a component is the following:

**Proposition 4.1** $C_2 \models S/S_1 \iff \forall C_1. [C_1 \models S_1 \Rightarrow C_1 \otimes C_2 \models S].$

In the next section, we establish this proposition when $S$ and $S_1$ are modal specifications and when $C_1$ and $C_2$ are composed using a synchronous product.

### 5 Quotient of modal specifications for behavioral reuse

The synchronous product of languages $C_1$ and $C_2$ over the same alphabet $\Sigma$ corresponds to the set theoretic intersection $C_1 \cap C_2$. We generalize the synchronous product to modal specifications:

**Definition 5.1** The synchronous product of $S_1$ and $S_2$ is the modal specification $S_1 \otimes S_2$ over $L(S_1) \cap L(S_2)$ with:

$$\forall u \in L(S_1) \cap L(S_2), \begin{cases} 
\text{may}(S_1 \otimes S_2)(u) = \text{may}(S_1)(u) \cap \text{may}(S_2)(u) \\
\text{must}(S_1 \otimes S_2)(u) = \text{must}(S_1)(u) \cap \text{must}(S_2)(u)
\end{cases}$$
Definition 5.3

(iii) If \( C_1, C_2 \subseteq \Sigma^* \) are viewed as modal specifications (cf. remark 3.5) then \( C_1 \otimes C_2 = C_1 \cap C_2 \).

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

\[
\downarrow X = \{ u \in \Sigma^* \mid \forall v \ u \preceq v \Rightarrow v \in X \}
\]

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 \( S/S_1 \). Now in order to define the typing functions \( \text{may}(S/S_1) \) and \( \text{must}(S/S_1) \), we proceed by case inspection:

Definition 5.3 The quotient of the modal specifications \( S \) and \( S_1 \) is the pseudomodal specification \( S/S_1 \) over \( \downarrow (L(S) \cup \neg L(S_1)) \) with:

(i) for all \( u \in L(S/S_1) \cap L(S_1) \):

(a) if \( a \in \text{may}(S)(u) \cap \text{must}(S_1)(u) \) then \( a \in \text{may}(S/S_1)(u) \) and \( a \in \text{must}(S/S_1)(u) \);

(b) if \( a \in \text{must}(S)(u) \cap \neg \text{must}(S_1)(u) \) then \( a \in \text{must}(S/S_1)(u) \) but \( a \notin \text{may}(S/S_1)(u) \);

(c) if \( a \in (\text{may}(S)(u) \setminus \text{must}(S)(u)) \) then \( a \in \text{may}(S/S_1)(u) \);

(d) if \( a \in \text{mustnot}(S)(u) \cap \text{mustnot}(S_1)(u) \) then \( a \in \text{may}(S/S_1)(u) \);

(e) if \( a \in \text{mustnot}(S)(u) \cap \text{may}(S_1)(u) \) then \( a \in \text{mustnot}(S/S_1)(u) \);

(ii) if \( u \in (L(S/S_1) \setminus L(S_1)) \) then \( \text{must}(S/S_1)(u) = \emptyset \) and \( \text{may}(S/S_1)(u) = \Sigma \).

Now we give for each possible case, an intuitive interpretation of the resulting modality assuming that \( C_1 \models S_1 \) and we intend to have \( C_2 \models S/S_1 \) and \( C_1 \otimes C_2 \models S \):

(i) first, when \( u \in L(S/S_1) \cap L(S_1) \):

(a) \( a \) is required in the global specification \( S \) that is \( u.a \) must belong to \( C_1 \otimes C_2 = C_1 \cap C_2 \). As \( a \) is guaranteed in \( S_1 \), \( u.a \in C_1 \) for all \( C_1 \models S_1 \) with \( u \in C_1 \); thus \( u.a \) must belong to \( C_1 \) to always have \( u.a \in C_1 \otimes C_2 \). a \( \in \text{must}(S/S_1)(u) \) and \( a \in \text{may}(S/S_1)(u) \).

(b) \( a \) is required in the global specification \( S \) but as \( a \notin \text{must}(S_1)(u) \), there are some \( C_1 \models S_1 \) such that \( u.a \notin C_1 \); hence, for all \( C_2 \), \( C_1 \otimes C_2 \) can’t be a model of \( S \). As a result, the trace \( u \) must be incoherently in \( S/S_1 \) and we let \( a \in \text{must}(S/S_1)(u) \) but \( a \notin \text{may}(S/S_1)(u) \) to model this inconsistency.

(c) \( a \) is allowed in the global specification \( S \) and \( u.a \) may belong to \( C_1 \otimes C_2 \). Thus, whether or not \( u.a \) belongs to \( C_1 \models S_1 \), \( u.a \) can belong to \( C_2 \) without violating the specification \( S \). Hence: \( a \in \text{may}(S/S_1)(u) \).

(d) \( a \) is forbidden in the global specification \( S \) and in \( S_1 \) thus, whether or not \( u.a \in C_2 \), we have \( u.a \notin C_1 \otimes C_2 \) which is conform to \( S \). Hence: \( a \in \text{may}(S/S_1)(u) \).
(e) $a$ is forbidden in the global specification $S$. As there are some $C_1 \models S_1$ with $u.a \in C_1$, we forbid $a$ in $S/S_1$: $a \in \text{mustnot}(S/S_1)(u)$. As a result, when $C_2 \models S/S_1$, $u.a \notin C_1 \otimes C_2$ which is conform to $S$.

(ii) if $u \in (L(S/S_1) \setminus L(S_1))$: as $u \notin C_1$, $u$ may belong to $C_2$, it won’t belong to $C_1 \otimes C_2$. As a result, $S/S_1$ is relaxed after the trace $u$ by taking $\text{must}(S/S_1)(u) = \emptyset$ (nothing is required) and $\text{may}(S/S_1)(u) = \Sigma$ (every event is allowed).

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

**Proposition 5.4** If $S$, $S_1$ and $S_2$ are modal specifications over $\Sigma$ then:

$$S_1 \otimes S_2 \leq S \iff S_2 \leq \rho(S/S_1)$$

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

**Lemma 5.5** $\forall \{C \otimes C' \mid C \models S\} = \forall \{C \mid C \models S\} \otimes C'$

**Proposition 5.6** If $S$ and $S_1$ are modal specifications over $\Sigma$ then:

$$C_2 \models S/S_1 \text{ iff } \forall C_1. [C_1 \models S_1 \Rightarrow C_1 \otimes C_2 \models S]$$

**Proof.** $(\Rightarrow)$ According to prop. 5.4, if $C_2 \models S/S_1$ (that is $C_2 \models \rho(S/S_1)$) then $C_2 \otimes S_1 \leq S$. Moreover as $C_1 \models S_1$ then $C_1 \otimes C_2 \leq S_1 \otimes C_2$. As a result, $C_1 \otimes C_2 \leq S$ that is $C_1 \otimes C_2 \models S$.

$(\Leftarrow)$ If for all $C_1$ such that $C_1 \models S_1$ we have $C_1 \otimes C_2 \models S$ then:

$$\forall \{C_1 \otimes C_2 \mid C_1 \models S_1\} \leq S$$

Thus, by lemma 5.5, $\forall \{C_1 \mid C_1 \models S_1\} \otimes C_2 \leq S$ i.e. $S_1 \otimes C_2 \leq S$ (by prop.3.9). According to prop. 5.4, $C_2 \leq S/S_1$ hence $C_2 \models S/S_1$. $\square$

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 $\text{must}("a")$ 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 $\text{may}("a")$ 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 $\text{must}(u) \subseteq X \subseteq \text{may}(u)$. This set of ”acceptance” set is thus given by:

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

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

Thus, for example if \( \text{may}(u) = \{b_1, b_2\} \) and \( \text{must}(u) = \emptyset \), we obtain \( \text{Acc}(u) = \{\emptyset, \{b_1\}, \{b_2\}, \{b_1, b_2\}\} \). 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 \( S \) is a pair \( S = \langle L, \text{Acc} \rangle \) where \( L \) is a prefix-closed language over \( \Sigma \) and \( \text{Acc} : L \rightarrow \mathcal{P}(\mathcal{P}(\Sigma)) \) is a map associating each trace \( u \in L \) to its acceptance set. For consistency, we require for all trace \( u \in L \):

1. \((C1)\) \( \text{Acc}(u) \neq \emptyset \)
2. \((C2)\) \( u.a \in L \) if and only there exists at least one set \( X \in \text{Acc}(u) \) such that \( a \in X \).

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

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

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

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

**Remark 6.4** \( \text{Acc}(u) = \emptyset \) is different from \( \emptyset \in \text{Acc}(u) \). 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:

\[ S_1 \leq S_2 \text{ iff } L(S_1) \subseteq L(S_2) \text{ and } \forall u \in L(S_1), \text{Acc}(S_1)(u) \subseteq \text{Acc}(S_2)(u) \]

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

6.2 Pseudo-acceptance specifications

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

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

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

(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 $\text{Acc}(R)(v)$ all sets containing the letter $a$ to enforce the consistency condition (C2);

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

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

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 $S_1 \land S_2$ is the reduction of the pseudo-acceptance specification $S_1 \& S_2$ whose support is $L(S_1) \land L(S_2)$ and with $\text{Acc}(S_1 \& S_2)(u) = \text{Acc}(S_1)(u) \land \text{Acc}(S_2)(u)$. The join $S_1 \lor S_2$ is defined over $L(S_1) \lor L(S_2)$ by:

$$
\begin{align*}
\text{if } u \in L(S_1) \land L(S_2), & \text{ } \text{Acc}(S_1 \land S_2)(u) = \text{Acc}(S_1) \lor \text{Acc}(S_2) \\
\text{if } u \in L(S_1) \setminus L(S_2), & \text{ } \text{Acc}(S_1 \land S_2)(u) = \text{Acc}(S_1) \\
\text{if } u \in L(S_2) \setminus L(S_1), & \text{ } \text{Acc}(S_1 \land S_2)(u) = \text{Acc}(S_2)
\end{align*}
$$

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 $S_1$ and $S_2$ is the acceptance specification $S_1 \otimes S_2$ over $L(S_1) \land L(S_2)$ with for all $u \in L(S_1) \land L(S_2)$:

$$
\text{Acc}(S_1 \otimes S_2)(u) = \{ X_1 \cap X_2 | X_1 \in \text{Acc}(S_1)(u) \text{ and } X_2 \in \text{Acc}(S_2)(u) \}
$$

This operation has for adjoint the following quotient operation:

**Definition 6.9** The quotient of the acceptance specification $S$ and $S_1$ is the pseudo-acceptance specification $S / S_1$ over $\downarrow \downarrow (L(S) \lor \neg L(S_1))$ with, for all $u \in L(S / S_1) \land L(S_1)$: $\text{Acc}(S / S_1)(u) = \{ Y \in \mathcal{P}(\Sigma) | \forall X \in \text{Acc}(S_1)(u), X \land Y \in \text{Acc}(S)(u) \}$ and for all $u \in (L(S / S_1) \land L(S_1))$, $\text{Acc}(S / S_1)(u) = \mathcal{P}(\Sigma)$.

**Proposition 6.10** If $S$, $S_1$ and $S_2$ are acceptance specifications over $\Sigma$ then:

$$
S_1 \otimes S_2 \preceq S \iff S_2 \preceq \rho(S / S_1)
$$
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 $S$ and $S_1$ are acceptance specifications then:

$$C_2 \models S / S_1 \iff \forall C_1. \models C_1 \Rightarrow C_1 \otimes C_2 \models S$$

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

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

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

- $k : \mathcal{AS} \rightarrow \mathcal{MS}$
  
  $k(S') = (L', \text{must'}, \text{may'})$ with
  
  $$\begin{align*}
  \text{may}'(u') &= \bigcup_{X \in \text{Acc'}(u')} X \\
  \text{must}'(u') &= \bigcap_{X \in \text{Acc'}(u')} X
  \end{align*}$$

We have: $k \circ j = \text{Id}$ but $j \circ k \neq \text{Id}$. 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 $S$ and $S_1$ two modal specifications, we let $S'$ be the acceptance specifications obtained by quotienting $j(S)$ and $j(S_1)$ using definition 6.9. Then the modal specification $k(S')$ is identical to the one obtained by quotienting $S$ and $S_1$ using definition 5.3. \hfill \square

Synchronous product is a very restrictive form of composition. In proposition 6.11, the component $(C_2, S / S_1)$ restrict the behavior of all possible $(C_1, S_1)$ so that the composite system $C_1 \otimes C_2$ realizes $S$. In the next section, we investigate an approach where the component $(C_2, S / S_1)$ may also contribute directly to the realization of the specification $S$: given $S$ a global specification over $\Sigma$ and $(C_1, S_1)$ a component to be reused over $\Sigma_1$, the events belonging to $\Sigma \setminus \Sigma_1$ are realized by the component $(C_2, S / S_1)$. Thus, we now consider residual of specifications when component composition corresponds to mixed product [4]. We also consider internalization of event that is $C_1$ and $C_2$ 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 $C_1$ and $C_2$ two languages respectively over $\Sigma_1$ and $\Sigma_2$, the mixed product of $C_1$ and $C_2$ is the language:

$$C_1 \bowtie C_2 = \{ w \in (\Sigma_1 \cup \Sigma_2)^* \mid \Pi_{\Sigma_1}(w) \in C_1 \text{ and } \Pi_{\Sigma_2}(w) \in C_2 \}$$
• Given \( C \) a language over \( \Sigma' \) and \( \Sigma \subseteq \Sigma' \), the restriction of \( C \) to the alphabet \( \Sigma \) is the language:
  \[
  C_{|\Sigma} = \{ u \in \Sigma^* \mid u = \Pi_{\Sigma}(w) \text{ with } w \in C \}
  \]

• Given \( C \) a language over \( \Sigma \) and \( \Sigma \subseteq \Sigma' \), the expansion of \( C \) to the alphabet \( \Sigma' \) is the language:
  \[
  C_{|\Sigma'} = \{ w \in \Sigma'^* \mid \Pi_{\Sigma}(w) \in C \}
  \]

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 \( S_1 \in \mathcal{AS}(\Sigma_1) \) and \( S_2 \in \mathcal{AS}(\Sigma_2) \) is the acceptance specification \( S_1 \uplus S_2 \) over \( L(S_1) \uplus L(S_2) \) with for all \( u \in L(S_1) \uplus L(S_2) \):

\[
\text{Acc}(S_1 \uplus S_2)(w) = \{ (X_1 \cup (\Sigma_2 \setminus \Sigma_1)) \cap (X_2 \cup (\Sigma_1 \setminus \Sigma_2)) \mid X_1 \in \text{Acc}(S_1)(\Pi_{\Sigma_1}(w)) \text{ and } X_2 \in \text{Acc}(S_2)(\Pi_{\Sigma_2}(w)) \}
\]

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 \( S \in \mathcal{AS}(\Sigma) \), the \( \tau \)-expansion of \( S \) to \( \Sigma' \) is the acceptance specification \( \tau_{\Sigma'}(S) \) over \( (L(S))_{|\Sigma'} \) with:

\[
\text{Acc}(\tau_{\Sigma'}(S))(w) = \{ Y \cup (\Sigma' \setminus \Sigma) \mid Y \in \text{Acc}(S)(\Pi_{\Sigma}(w)) \}
\]

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 \( S_1 \in \mathcal{AS}(\Sigma_1) \) and \( S_2 \in \mathcal{AS}(\Sigma_2) \):

\[
S_1 \uplus S_2 = \tau_{\Sigma_1 \uplus \Sigma_2}(S_1) \otimes \tau_{\Sigma_1 \uplus \Sigma_2}(S_2)
\]

The restriction operation adjoint of the \( \tau \)-expansion is the following:

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

\[
\text{Acc}(\Pi_{\Sigma}(S'))(u) = \{ Y \mid Y \cup (\Sigma' \setminus \Sigma) \in \bigcap \{ \text{Acc}(S')(w) \mid w \in L(S') \text{ and } \Pi_{\Sigma}(w) = u \} \}
\]

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

\[
\tau_{\Sigma'}(S) \leq S' \iff S \leq \Pi_{\Sigma}(S')
\]

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 $S \in \mathcal{AS}(\Sigma')$, the restriction of $S$ to $\Sigma$ is the acceptance specification $S|_\Sigma$ over $(L(S))|_\Sigma$ with:

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

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

$$\text{Acc}(S'|_{\Sigma'})(w) = \{ X \mid X \cap \Sigma \in \text{Acc}(S')((\Pi_\Sigma(w)) \} \cup \{ X \mid X \neq \emptyset \text{ and } X \subseteq (\Sigma' \setminus \Sigma) \}$$

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

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

$$S|_\Sigma \leq S' \iff S \leq S'|_{\Sigma'}$$

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

From the previous propositions, we can deduce:

$$(S_1 \bowtie S_2)|_{\Sigma} \leq S \iff S_1 \bowtie S_2 \leq S|_{(\Sigma_1 \cup \Sigma_2)}$$ by prop. 7.9

$$\iff \tau_{\Sigma_1 \cup \Sigma_2}(S_1) \otimes \tau_{\Sigma_1 \cup \Sigma_2}(S_2) \leq S|_{(\Sigma_1 \cup \Sigma_2)}$$ by prop. 7.4

$$\iff \tau_{\Sigma_1 \cup \Sigma_2}(S_2) \leq S|_{(\Sigma_1 \cup \Sigma_2)} / \tau_{\Sigma_1 \cup \Sigma_2}(S_1)$$ by prop. 6.10

$$\iff S_2 \leq \Pi_{\Sigma_2}(S|_{(\Sigma_1 \cup \Sigma_2)} / \tau_{\Sigma_1 \cup \Sigma_2}(S_1))$$ by prop. 7.6

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 $S \in \mathcal{AS}(\Sigma)$, $S_1 \in \mathcal{AS}(\Sigma_1)$ and $\Sigma$ such that $\Sigma \subseteq \Sigma_1 \cup \Sigma_2$:

$$\forall \mathcal{C}_1, \mathcal{C}_1 \models S_1 \models (C_1 \bowtie C_2)|_{\Sigma} \models S_2 \iff \mathcal{C}_2 \models S_2|_{(\Sigma_1 \cup \Sigma_2)} / \tau_{\Sigma_1 \cup \Sigma_2}(S_1)$$

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 $S$ over the alphabet $\Sigma = \{msg, ok, fail\}$:
To realize $S$, we aim at reusing a component satisfying the following specification $S_1$ describing the behavior of a communication channel:

$S_1$ is defined over the alphabet $\Sigma_1 = \{\text{msg}, \text{send}, \text{ack}, \text{nack}, \text{fail}, \text{ok}\}$. Note that loss of message is allowed in $\text{Acc}(S_1)(2)$ which is not the case in $\text{Acc}(S_1)(4)$.

- We let $\Sigma_2 = \{\text{ack}, \text{nack}, \text{send}\}$. As $\Sigma \setminus \Sigma_1 = \emptyset$, $(C_2, S/S_1)$ will restrict the behavior of $(C_1, S_1)$ to enforce $S$. We have: $\tau_{\Sigma_1 \cup \Sigma_2}(S_1) = S_1$ as $\Sigma_2 \subseteq \Sigma_1$.
- We compute $S_{\downarrow(\Sigma_1 \cup \Sigma_2)}$:

- $\text{Acc}(S_{\downarrow(\Sigma_1 \cup \Sigma_2)})(0') = \{\{\text{msg}\}\} \cup X, X\}$ with $X \subseteq \Sigma_2$ and $X \neq \emptyset$.
- $\text{Acc}(S_{\downarrow(\Sigma_1 \cup \Sigma_2)})(1') = \{\{\text{ok}\}\} \cup X, X\}$ with $X \subseteq \Sigma_2$ and $X \neq \emptyset$.
- Then we compute the quotient $S_{\downarrow(\Sigma_1 \cup \Sigma_2)} / \tau_{\Sigma_1 \cup \Sigma_2}(S_1)$ (in the following figure, only transitions labeled by required events are depicted):
Last, we apply the operation $\Pi_{\Sigma_2}$ on the acceptance specification equivalent to the previous pseudo-acceptance specification. The result is the following:

\[
\begin{align*}
&\text{Acc}(00') = \{\{\text{msg}\} \cup X \mid X \subseteq \{\text{ok}, \text{fail}, \text{ack}, \text{nack}, \text{send}\}\}; \\
&\text{Acc}(11') = \{\{\text{send}\} \cup X \mid X \subseteq \{\text{ok}, \text{fail}, \text{ack}, \text{nack}, \text{msg}\}\} \\
&\text{Acc}(21') = \{\{\text{ack}, \text{nack}\} \cup X \mid X \subseteq \{\text{ok}, \text{fail}, \text{msg}, \text{send}\}\} \\
&\text{Acc}(31') = \{\{\text{send}\} \cup X \mid X \subseteq \{\text{msg}, \text{ok}, \text{fail}, \text{ack}, \text{nack}\}\} \\
&\text{Acc}(41') = \{\{\text{ack}\} \cup X, \{\text{nack}\} \cup X, \{\text{ack}, \text{nack}\} \cup X \mid X \subseteq \{\text{ok}, \text{fail}, \text{msg}, \text{send}\}\} \\
&\text{Acc}(51') = \{\{\text{ok}\} \cup X \mid X \subseteq \{\text{msg}, \text{fail}, \text{ack}, \text{nack}, \text{send}\}\} \\
&\text{Acc}(61') = \emptyset
\end{align*}
\]

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

\section{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 $S_1$.

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 i.e. 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.

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

**References**


