Residual for Component Specifications

Jean-Baptiste Raclet

I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° 1843

RESIDUAL FOR COMPONENT SPECIFICATIONS

JEAN-BAPTISTE RACLET

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Residual for Component Specifications

Jean-Baptiste Raclet*

Systèmes communicants
Projet S4

Publication interne n ° 1843 — Avril 2007 — 19 pages

Abstract: Component-based design aims at building new software systems from pre-existing components. However in current component platforms, reuse of a component is completed from its signature. Thus nothing can be inferred about the interaction between the reused component and its environment and behavioral mismatch may occur.

To express component reuse at a behavioral level, we introduce modal automata and acceptance automata as intuitive formalisms for behavioral interface description. From the expressiveness point of view, these formalisms allow to state some forms of liveness properties.

We argue that reusing a component $\mathcal{C}_1$, the behavior of which is described by the specification $\mathcal{S}_1$, in order to realize a global system specified by $\mathcal{S}$ amounts to exhibiting a residual specification $\mathcal{S}/\mathcal{S}_1$ so that any model $\mathcal{C}_2$ of $\mathcal{S}/\mathcal{S}_1$ when composed with $\mathcal{C}_1$ constitutes a composite system satisfying $\mathcal{S}$.

We define a quotient operation for modal automata and acceptance automata of polynomial complexity (quadratic in the size of the specifications).

Key-words: Component-based design, Behavioral interface, Adaptor synthesis, Modal and acceptance automata, Residual of specifications

(Résumé : tsvp)

* Jean-Baptiste.Raclet@irisa.fr

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Résidu de spécifications de composants

Résumé : Le développement logiciel basé composants vise à construire des logiciels à partir de composants préexistants. Cependant dans les plateformes orientées composants actuelles, un composant est réutilisé à partir des informations comprises dans sa signature. Ainsi, aucune déduction ne peut être faite sur la façon dont le composant réutilisé et son environnement interagissent et leurs comportements peuvent, en particulier, être incompatibles.

Pour exprimer la réutilisabilité d'un composant à un niveau comportemental, nous présentons les automates modaux et les automates à ensembles d'acceptation comme des formalismes intuitifs pour d'écrire l'interface d'un composant. D'un point de vue expressivité, ces formalismes permettent de formuler des propriétés de vivacité.

Nous réduisons la réutilisation d'un composant $\mathcal{C}_1$ dont le comportement est décrit grâce à la spécification $\mathcal{S}_1$ afin de réaliser un système global spécifié par $\mathcal{S}$, au calcul la spécification résiduelle $\mathcal{S}/\mathcal{S}_1$ dont chaque modèle $\mathcal{C}_2$ composé avec $\mathcal{C}_1$ constitue un système satisfaisant $\mathcal{S}$. Nous définissons une opération de quotient pour les automates modaux et les automates à ensembles d'acceptation dont la complexité est polynomiale (quadratique en la taille des spécifications).

Mots clés : Développement logiciel basé composants, Interface comportementale, Synthèse d'adaptateurs, Automate modal, Automate à ensembles d'acceptation, Résidu de spécifications

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 express component reuse at a behavioral level rather than at a signature level. More precisely, we study the two following issues:

- *(Q1) adaptability:* can a component $\mathcal{C}_1$, the behavior of which is described in its interface by the specification $\mathcal{S}_1$, be used to build a system satisfying a global specification $\mathcal{S}$?
- *(Q2) adaptor synthesis:* when a component $\mathcal{C}_1$ is adaptable into a specification $\mathcal{S}$, synthesize an adaptor $\mathcal{C}_2$ such that the composition of $\mathcal{C}_1$ and $\mathcal{C}_2$ satisfies $\mathcal{S}$.

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 top-down design. In [MvB83], top-down design was introduced as 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 [MvB83, EFYBvB06], finite state machine [YVB+01] (with inclusion of traces as equivalence relation), CCS or CSP processes [Par87, LX90] (with bisimulation as equivalence relation) or input/output automata [KNM97]. These models of specification are limited in expressivity as they are restricted to safety properties. In this paper, we improve expressivity of these previous approaches by using modal automata and acceptance automata to deal with forms of liveness properties.

Modal automata are standard finite automata with modalities "may" or "must" on transitions. They were originally used in [Lar89] to study the refinement of actions; here, they are introduced as an intuitive automata-based specification formalism. Contrary to standard automata, modal automata can express reactivity property like "any stimulus a (if any) is followed by a reaction b".

We also introduced acceptance automata as a specification formalism (they were originally studied in [Hen85] to model nondeterministic machines). Acceptance automata are standard finite automata each state of which is associated with a set of so called "acceptance sets". This set records the various situations, due to non determinism, that may be associated with a given state. Any such situation is given by the set of actions the system is ready to engage with (an acceptance set). This specification formalism subsumes modal

automata. They can express progressive property like "any stimulus a is followed by *at least* a or b as reaction".

Model automata and acceptance automata both specify sets of standard automata. We define a quotient operation for modal automata and acceptance automata such that:

- Solution for *(Q1)* is reduced to satisfiability of the residual specification $\mathcal{S}/\mathcal{S}_1$;
- Every model $\mathcal{C}_2$ of the specification $\mathcal{S}/\mathcal{S}_1$ is a suitable adaptor for *(Q2)*.

As the reused component is a black-box, the composition of $\mathcal{C}_1$ and $\mathcal{C}_2$ must realize $\mathcal{S}$ whatever the implementation $\mathcal{C}_1$ of $\mathcal{S}_1$ could be. Thus, the characteristic property of our quotient operation is the following:

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

Quotient of mu-calculus formulas was investigated in [AVW03]. 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 modal automata or acceptance automata as specifications are polynomial.

This paper is organized as follows: section 2 introduces modal automata as a specification formalism. The quotient of modal automata is proposed in section 3. Then section 4 is devoted to the quotient of acceptance automata. A comparison with related works of [PdAHSV02] and a hint for some line of future work conclude the paper.

2 Modal automata as specifications

In this section, we introduce modal automata as a formalism to specify sets of standard finite automata.

2.1 Modal automata, modal trees and their models

Modal automata, originally introduced by Larsen [Lar89], can be seen as automata with a modality on each labeled transition: we distinguish actions that necessarily occur (must-modality) and actions that possibly occur (may-modality). Moreover, the absence of a transition labeled by an action a from a state q tells that this action is forbidden in this state. Because of the may modality, this model is well suited to represent partially specified systems. More formally:

Definition 1
A modal automaton $\mathcal{S}$ over an alphabet Σ is a tuple $(Q, q^0, \Delta, may, must)$ where Q is a non-empty and finite set of states and $q^0 \in Q$ is the initial state. The function $\Delta : Q \times \Sigma \to Q$ is a partial function describing a deterministic transition system. The functions $may, must : Q \to \mathcal{P}(\Sigma)$ are partial functions that type actions:

- $a \in may(q)$ *means that* a *is allowed in* q*;*
- $a \in must(q)$ *means that* a *is required in* q*;*
- $a \notin may(q)$ *(also denoted* $a \in \neg may(q)$*) means that* a *is forbidden in* q*.*

A model of a modal specification $\mathcal{S}$ is a deterministic automaton $\mathcal{C} = (R, r^0, \delta)$ with R a non-empty finite set of states, $r^0 \in R$ the initial state, $\delta : R \times \Sigma \to R$ a partial function of transition. We let $out(r)$ be the set of actions a such that $\delta(r, a)$ is defined. We extend this notation to traces (sequences of actions) $u \in \Sigma^*$: $out(u)$ indicates the possible actions in the state reached from the initial state after reading the word u.

The definition of the validation relation is the following:

Definition 2
The automaton $\mathcal{C}$ *is a model of the modal automaton* $\mathcal{S}$*, noted* $\mathcal{C} \models \mathcal{S}$*, if there exists a relation* $\sim \subseteq R \times Q$ *such that:*

- $r^0 \sim q^0$
- $\forall (r, q)$ *such that* $r \sim q$ *:* $must(q) \subseteq out(r) \subseteq may(q)$ *and furthermore,* $\forall a \in out(r), \delta(r, a) \sim \Delta(q, a)$*.*

Thus, every trace u of a model $\mathcal{C}$ of $\mathcal{S}$ is also a trace of $\mathcal{S}$ and if r and q are respectively the states reached after the word u in $\mathcal{C}$ and $\mathcal{S}$ from their initial state, then: $must(q) \subseteq out(r) \subseteq may(q)$ which means that in state r one can perform all required actions and none of the forbidden actions of a related state q of the specification. By determinacy, the simulation relation $\sim$ is unique when it exists. Note that this definition implies that $must(q) \subseteq may(q)$ which means that the actions required in every state q of the specification must also be allowed in this state. When this condition is does not hold, state q is said to be incoherent (see paragraph 2.3).

Definition 3
A modal tree $T \in MT(\Sigma)$ *is a triplet* $T = \langle L, must, may \rangle$ *where* $L \in TD(\Sigma)$ *is a tree domain that is a non empty prefix-closed language on* Σ *and* $may, must : L \to \mathcal{P}(\Sigma)$*.*

A modal tree can be seen as an (unfolded) modal automaton with $Q = L$, $q^0 = \varepsilon$ and $\Delta(u, a) = ua$. Modal trees are ordered by the following relation:

$$T_1 \leq T_2 \text{ iff } L(T_1) \subseteq L(T_2) \text{ and } \forall u \in L(T_1), \quad \begin{cases} may(T_1)(u) & \subseteq & may(T_2)(u) \\ must(T_1)(u) & \supseteq & must(T_2)(u) \end{cases}$$

Any tree domain L can be viewed as an (unfolded) automaton, and also as a modal tree where $may(u) = must(u) = out(u) = \{a \in \Sigma \mid u \cdot a \in L\}$ for all word $u \in L$. Hence $L \models T$ if and only if $L \leq T$. The specialization preorder is given by inclusion of the corresponding sets of models:

$$T_1 \sqsubseteq T_2 \text{ iff } \forall L \in TD(\Sigma) \quad L \models T_1 \Rightarrow L \models T_2$$

Thus, this preorder relation contains the order relation: $\leq \subseteq \sqsubseteq$. Finally we let $\equiv$ denote the induced equivalence relation on modal trees:

$$T_1 \equiv T_2 \text{ iff } \forall L \in TD(\Sigma) \quad L \models T_1 \Leftrightarrow L \models T_2$$

The logical fragment equivalent to modal automata has been identified in [FP06]. It is a fragment of the mu-calculus called the conjunctive nu-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.

2.2 The lattice of modal trees

The set of modal trees MT equipped with the partial order $\leq$ is a complete distributive lattice (hence a bounded lattice) where the meet $T_1 \wedge T_2$ is the modal tree over $L(T_1) \cap L(T_2)$ with:

$$\forall u \in L(T_1) \cap L(T_2), \quad \begin{cases} may(T_1 \wedge T_2)(u) & = & may(T_1)(u) & \cap & may(T_2)(u) \\ must(T_1 \wedge T_2)(u) & = & must(T_1)(u) & \cup & must(T_2)(u) \end{cases}$$

and the join $T_1 \vee T_2$ is the modal tree over $L(T_1) \cup L(T_2)$ with:

$$\forall u \in L(T_1) \cup L(T_2), \quad \begin{cases} may(T_1 \vee T_2)(u) & = & may(T_1)(u) & \cup & may(T_2)(u) \\ must(T_1 \vee T_2)(u) & = & must(T_1)(u) & \cap & must(T_2)(u) \end{cases}$$

if we assume that each map $must, may : \Sigma^* \to \wp(\Sigma)$ is extended to the whole of Σ^* by letting $may(u) = \emptyset$ and $must(u) = \Sigma$ when $u \notin L$. With this convention the least element is given by $L_\perp = \varepsilon$ and $must_\perp(\varepsilon) = \Sigma$ and $may_\perp(\varepsilon) = \emptyset$ and the greatest element is given by $L_\top = \Sigma^*$ and $must_\top(u) = \emptyset$ and $may_\top(u) = \Sigma$ for all $u \in \Sigma^*$.

2.3 Reduction of inconsistency

As previously noticed, inconsistency may arise in some state. More precisely, a modal tree is said to be reduced if for all $u \in L$ one has $must(u) \subseteq may(u)$ and $u \cdot a \in L$ if and only if $u \in L$ and $a \in may(u)$. A node that does not satisfy one of the above properties is said to be *incoherent*. Thus a modal tree is reduced when it contains no incoherent node. The meet operation may introduce incoherent nodes but a join of a non empty family of reduced modal trees is always reduced. This allows us to establish the following:

Proposition 1
Either a modal tree T has no model or there exists a largest reduced modal tree $\rho(T)$ smaller than T, and $\rho(T)$ has the same models as T: $T \equiv \rho(T)$.

Proof First we notice that a language $L \in TD(\Sigma)$ viewed as a modal tree is reduced by definition: $must(u) = may(u)$ and the second condition is satisfied since this set $out(u)$ contains exactly those letters $a \in \Sigma$ such that $u \cdot a \in L$ and $out(u) = may(u)$. Thus if we let

$$\rho(T) = \vee \{T' \mid T' \text{ is reduced and } T' \leq T\}$$

the corresponding set contains all models of T and $L \models T \Rightarrow L \leq \rho(T)$ which shows that $T \sqsubseteq \rho(T)$. For the converse direction, either this set is empty, which is the case when and only when T has no models (and then $\rho(T)$ is the minimal element $\bot$) or the corresponding join is the largest reduced modal tree smaller than T. Moreover $\rho(T) \leq T$ and thus $\rho(T) \sqsubseteq T$, and T and $\rho(T)$ have the same models: $T \equiv \rho(T)$. $\diamondsuit$

The proof of the above proposition shows that any property of modal trees preserved by non-empty joins and satisfied on the sublattice of $TD(\Sigma)$ of models, provides a coherent notion of reduction: we don't lose any model by reduction. The following result shows that the chosen notion of reduction is indeed the largest one.

Proposition 2
A modal tree T is reduced if and only if

$$T = \bigvee \{L \mid L \models T\}$$

Proof The condition is clearly necessary since

$$\{L \in TD(\Sigma) \mid L \models T\} \subseteq \{T' \mid T' \text{ is reduced and } T' \leq T\}$$

And thus : $T = \bigvee \{L \in TD(\Sigma) \mid L \models T\} \leq \rho(T) \leq T$.

For the converse direction we can prove simultaneously by co-induction on the structure of a reduced modal tree $T = \langle L, must, may\rangle$ that

1. the set $[\![T]\!] = \{L \in TD(\Sigma) \mid L \models T\}$ of models of T is given by

$$[\![T]\!] = \left\{ \{\varepsilon\} \cup \bigcup_{a \in X} a \cdot L' \;\middle|\; must(\varepsilon) \subseteq X \subseteq may(\varepsilon) \text{ and } L' \in [\![(a^{-1}T)]\!] \right\}$$

   where $a^{-1}T = \langle a^{-1}L, a^{-1}must, a^{-1}may\rangle$ is the subtree of T rooted at the a-successor of the root of T:

$$\begin{array}{rcl} a^{-1}L & = & \{u \in \Sigma^* \mid a \cdot u \in L\} \\ \left(a^{-1}must\right)(u) & = & must(a \cdot u) \\ \left(a^{-1}may\right)(u) & = & may(a \cdot u) \end{array}$$

2. the set $[\![T]\!]$ is not empty;
3. $L = \bigcup [\![T]\!]$;
4. $may(u) = \bigcup out(L)(u)$ and $must(u) = \bigcap out(L)(u)$ with $L \in [\![T]\!]$ and such that $u \in L$.

From the last two items it follows that $T = \bigvee \{L \in TD(\Sigma) \mid L \models T\}$. $\diamondsuit$

If T is regular, i.e. is the unfolding of a finite modal automaton $\mathcal{S}$, then if it is non empty $\rho(T)$ is the unfolding of an automaton obtained by iteration of the following operations:

1. we remove all states q such that $must(q) \not\subseteq may(q)$ and consequently all transitions $q' \xrightarrow{a} q$ leading to that state;
2. we remove from the set $may(q)$ all letter a for which there is no a-labeled transition stemming from q (i.e. $a \notin out(q)$);
3. for every state q and letter $a \in out(q) \setminus (must(q) \cup may(q))$ we remove the a-labeled transition stemming from q;
4. we remove all states which are no longer accessible from the initial state.

Proposition 3
$T_1 \sqsubseteq T_2$ if and only if $\rho(T_1) \leq \rho(T_2)$

Proof ($\Rightarrow$) $T_1 \sqsubseteq T_2$ i.e. $\forall L : L \models T_1 \Rightarrow L \models T_2$ implies that:

$$\models_{T_1} \subseteq \models_{T_2} \text{ where } \models_{T_i} = \{L \mid L \models T_i\}$$

and thus $\rho(T_1) = \bigvee \models_{T_1} \leq \bigvee \models_{T_2} = \rho(T_2)$.

($\Leftarrow$) Let $L_1 \in TD(\Sigma)$ such that $L_1 \models T_1$. As $\rho(T_1) \equiv T_1$, $L_1 \models \rho(T_1)$.
Thus, as $\rho(T_1) \leq \rho(T_2)$, we have $L_1 \models \rho(T_2)$ and $L_1 \models T_2$. $\diamondsuit$

Thus, for reduced modal trees, the order relation and the inclusion of sets of models coincide.

Now we consider the problem of reuse of a component at a behavioral level and solve it when component specifications are modal automata.

3 Quotient of modal automata for behavioral reuse of components

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 $\mathcal{C}_2$ 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 the residual operation is the following:

Proposition 4
$\mathcal{C}_2 \models \mathcal{S}/\mathcal{S}_1$ iff $\forall \mathcal{C}_1.[\mathcal{C}_1 \models \mathcal{S}_1 \Rightarrow \mathcal{C}_1 \otimes \mathcal{C}_2 \models \mathcal{S}]$.

We first establish proposition 4 for modal trees to make the technical parts easier to follow. Then we prove the result for regular modal trees (i.e. unfolding of finite modal automata). The product of implementations we consider here corresponds to the synchronous product of automata that is $\mathcal{C}_1 \otimes \mathcal{C}_2 = (R_1 \times R_2, (r_1^0, r_2^0), \delta)$ with $\delta((r_1, r_2), a) = (r_1', r_2')$ if and only if $\delta_1(r_1, a) = r_1'$ and $\delta_2(r_2, a) = r_2'$.

3.1 Quotient of modal trees

First we generalize the synchronous product to modal trees:

Definition 4
The synchronous product of T_1 and T_2 is the modal tree $T_1 \otimes T_2$ over $L(T_1) \cap L(T_2)$ with:

$$\forall u \in L(T_1) \cap L(T_2), \left\{ \begin{array}{lcll} must(T_1 \otimes T_2)(u) & = & must(T_1)(u) & \cap \quad must(T_2)(u) \\ may(T_1 \otimes T_2)(u) & = & may(T_1)(u) & \cap \quad may(T_2)(u) \end{array} \right.$$

Notice that:

i This operator is monotonic over the order relation $\leq$:

$$T_1 \leq T_2 \Rightarrow (T \otimes T_1 \leq T \otimes T_2 \text{ and } T_1 \otimes T \leq T_2 \otimes T)$$

ii $T_1 \otimes T_2$ is reduced if T_1 and T_2 are reduced.

iii If $L_1, L_2 \in TD(\Sigma)$ are viewed as modal trees then $L_1 \otimes L_2 = L_1 \cap L_2$.

At the level of tree domains, 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 $\neg M = \Sigma^* \setminus M$ is the set theoretic complement and where:

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

with $\preceq$ as the prefix order relation ($v \preceq u$ iff $\exists w \quad u = v \cdot w$), 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 tree T/T_1 that is the prefix-closed language $L(T/T_1)$. Now in order to define the typing functions $may(T/T_1)$ and $must(T/T_1)$, we proceed by case inspection. In the following definition, we give for each possible case, an intuitive interpretation of the resulting modality assuming that $L_1 \models T_1$ and we intend to have $L_2 \models T/T_1$ and $L_1 \otimes L_2 \models T$:

Definition 5
The quotient of the reduced modal trees T and T_1 is the modal tree T/T_1 over $\downarrow (L(T) \cup \neg L(T_1))$ with:

- *for all $u \in L(T/T_1) \cap L(T_1) = L(T) \cap L(T_1)$:*

- *if* $a \in must(T)(u) \cap must(T_1)(u)$*:*
  then a *is required in the global specification* T *that is* $u.a$ *must belong to* $L_1 \otimes L_2 = L_1 \cap L_2$*. As* a *is guaranteed in* T_1*,* $u.a \in L_1$ *for all* $L_1 \models T_1$ *with* $u \in L_1$*; thus* $u.a$ *must belong to* L_2 *to always have* $u.a \in L_1 \otimes L_2$*:* $a \in must(T/T_1)(u)$*.*
- *if* $a \in must(T)(u) \cap \neg must(T_1)(u)$*:*
  then a *is required in the global specification* T *but as* $a \notin must(T_1)(u)$*, there are some* $L_1 \models T_1$ *such that* $u.a \notin L_1$*; hence, for all* L_2*,* $L_1 \otimes L_2 \not\models T$*. As a result, the node* u *is incoherent in* T/T_1*. As* $u.a \in L(T/T_1)$*, we let* $a \in \neg may(T/T_1)(u)$ *to model this inconsistency.*
- *if* $a \in may(T)(u)$*:*
  then a *is allowed in the global specification* T *and* $u.a$ *may belong to* $L_1 \otimes L_2$*. Thus, whether or not* $u.a$ *belongs to* $L_1 \models T_1$*,* $u.a$ *can belong to* L_2 *without violating the specification* T*. Hence:* $a \in may(T/T_1)(u)$*.*
- *if* $a \in \neg may(T)(u) \cap \neg may(T_1)(u)$*:*
  then a *is forbidden in the global specification* T *and in* T_1 *thus, whether or not* $u.a \in L_2$*, we have* $u.a \notin L_1 \otimes L_2$ *which is conform to* T*. Hence:* $a \in may(T/T_1)(u)$*.*
- *if* $a \in \neg may(T)(u) \cap may(T_1)(u)$*:*
  then a *is forbidden in the global specification* T*. As there are some* $L_1 \models T_1$ *with* $u.a \in L_1$*, we forbid* a *in* T/T_1*:* $a \in \neg may(T/T_1)(u)$*. As a result, when* $L_2 \models T/T_1$*,* $u.a \notin L_1 \otimes L_2$ *which is conform to* T*.*

- *if* $u \in (L(T/T_1) \setminus L(T_1))$*: as* $u \notin L_1$ *and as* L_1 *is prefix-closed, sequences of actions* $w = u.v$ *may belong to* L_2*, they won't belong to* $L_1 \otimes L_2$*. As a result,* T/T_1 *is relaxed after the trace* u *by taking* $must(T/T_1)(u) = \emptyset$ *(nothing is required) and* $may(T/T_1)(u) = \Sigma$ *(every action is allowed).*

We establish that the adjoint operation of this quotient operation is the synchronous product of definition 4. We first prove this two lemmas:

Lemma 1
If $T_1 \leq T_2$ *then* $\rho(T_1) \leq \rho(T_2)$*.*

Proof By Prop. 3 and since $\leq \subseteq \sqsubseteq$ ♢

Lemma 2
If T *is reduced then* $T \leq T'$ *if and only if* $T \leq \rho(T')$*.*

Proof If $T \leq T'$ then, by lemma 1, $\rho(T) \leq \rho(T')$. As T is a reduced modal tree, $\rho(T) = T$ and $T \leq \rho(T')$.

Conversely $\rho(T') \leq T'$ together with $T \leq \rho(T')$ entail $T \leq T'$. ♢

Proposition 5
If T, T_1 and T_2 are reduced then: $T_1 \otimes T_2 \leq T$ iff $T_2 \leq T/T_1$.

Proof First we prove that $T_1 \otimes T_2 \leq T \Rightarrow T_2 \leq T/T_1$. For prefix closed languages $L(T), L(T_1), L(T_2)$, one has:

$$L(T_1) \cap L(T_2) \subseteq L(T) \Rightarrow L(T_2) \subseteq\downarrow (L(T) \cup \neg L(T_1))$$

We let $u \in L(T_2)$, we first have to prove that $may(T_2)(u) \subseteq may(T/T_1)(u)$.

- If $u \in L(T/T_1) \cap L(T_1)$ and $a \in may(T_2)(u)$, two cases can occur:
  - $a \in may(T_1)(u)$: as $u \in L(T_1) \cap L(T_2)$ and $T_1 \otimes T_2 \leq T$ then we have $may(T_1)(u) \cap may(T_2)(u) \subseteq may(T)(u)$ that is $a \in may(T)(u)$. Then by quotient definition, $a \in may(T/T_1)(u)$.
  - $a \in \neg may(T_1)(u)$: as $u \in L(T_1) \cap L(T_2)$ and $T_1 \otimes T_2 \leq T$ then we have $must(T_1)(u) \cap must(T_2)(u) \supseteq must(T)(u)$ and thus $a \notin must(T)(u)$. Then $a \in may(T)(u) \setminus must(T)(u)$ or $a \in \neg may(T)(u)$ and by quotient definition, we have $a \in may(T/T_1)(u)$.
- If $u \in L(T/T_1) \setminus L(T_1)$ and $a \in may_2(u)$, $may(T/T_1)(u) = \Sigma$ hence we have $a \in may(T/T_1)(u)$.

We let $u \in L(T_2)$, we then prove that $must(T_2)(u) \supseteq must(T/T_1)(u)$:

- If $u \in L(T/T_1) \cap L(T_1)$ and $a \in must(T/T_1)(u)$, by quotient definition $a \in must(T)(u)$ and $a \in must(T_1)(u)$. Furthermore, as $u \in L(T_1) \cap L(T_2)$ and $T_1 \otimes T_2 \leq T$, $must(T_1)(u) \cap must(T_2)(u) \supseteq must(T)(u)$ then $a \in must(T_2)(u)$.
- If $u \in L(T/T_1) \setminus L(T_1)$ then $must(T_2)(u) \supseteq must(T/T_1)(u)$ trivially holds since $must(T/T_1)(u) = \emptyset$.

Secondly we prove that $T_2 \leq T/T_1 \Rightarrow T_1 \otimes T_2 \leq T$. We have for prefix closed languages: $L(T_2) \subseteq L(T/T_1) \Rightarrow L(T_1) \cap L(T_2) \subseteq L(T)$.

We let $u \in L(T_1) \cap L(T_2)$, we have to prove that $may(T_1)(u) \cap may(T_2)(u) \subseteq may(T)(u)$ and $must(T_1)(u) \cap must(T_2)(u) \supseteq must(T)(u)$.
Note that, as $u \in L(T_2)$ then $u \in L(T/T_1)$ and $u \in L(T/T_1) \cap L(T_1)$.
As $T_2 \leq T/T_1$ and T_2 is reduced, we have according to lemma 2: $T_2 \leq \rho(T/T_1)$. Thus u is a trace of $\rho(T/T_1)$ and the state reached in T/T_1 from the initial state after u is coherent.
Let $a \in may(T_1)(u) \cap may(T_2)(u)$, as $T_2 \leq T/T_1$ and $a \in may(T_2)(u)$, we have $a \in may(T/T_1)(u)$. Then, by the definition of the quotient, $a \in may(T)(u)$.
Now if $a \in must(T)(u)$, as there is no inconsistency in T/T_1 after u, $a \in must(T_1)(u)$ and $a \in must(T/T_1)(u)$. Then, as $must(T_2)(u) \supseteq must(T/T_1)(u)$, $a \in must(T_1)(u) \cap must(T_2)(u)$. ♢

Next to prove proposition 4 for modal trees, we prove the following lemma:

Lemma 3
$\vee\{L \otimes L' \mid L \models T\} = \vee\{L \mid L \models T\} \otimes L'$

Proof As the join of tree domains is the union of languages, we have:
$$\begin{array}{rcl} \vee\{L \otimes L' \mid L \models T\} & = & \bigcup_i (L^i \cap L') \text{ with } L^i \models T \\ & = & (\bigcup_i L^i) \cap L' \\ & = & \vee\{L \mid L \models T\} \otimes L' \end{array}$$ ♢

Proposition 6
If T and T_1 are reduced then:

$$L_2 \models T/T_1 \text{ iff } \forall L_1.[L_1 \models T_1 \Rightarrow L_1 \otimes L_2 \models T]$$

Proof ($\Rightarrow$) According to prop. 5, if $L_2 \models T/T_1$ then $L_2 \otimes T_1 \leq T$.

Moreover as $L_1 \models T_1$ then $L_1 \otimes L_2 \leq T_1 \otimes L_2$. As a result, $L_1 \otimes L_2 \leq T$ that is $L_1 \otimes L_2 \models T$.

($\Leftarrow$) If for all L_1 such that $L_1 \models T_1$ we have $L_1 \otimes L_2 \models T$ then:

$$\vee\{L_1 \otimes L_2 \mid L_1 \models T_1\} \leq T$$

Thus, by lemma 3, $\vee\{L_1 \mid L_1 \models T_1\} \otimes L_2 \leq T$ i.e. $T_1 \otimes L_2 \leq T$. According to prop. 5, $L_2 \leq T/T_1$ hence $L_2 \models T/T_1$. ♢

To generalize this result for modal trees to modal automata, we prove that some kind of regularity is preserved when quotienting two regular modal trees:

3.2 Quotient of regular modal trees

Definition 6
A modal tree is said to be regular if $L(T)$ is regular and if for all sets $X \subseteq \Sigma$, the languages $L_X^{must}(T) = \{u \in L(T) \mid must(u) = X\}$ and $L_X^{may}(T) = \{u \in L(T) \mid may(u) = X\}$ are regular sets.

Proposition 7
If T and T_1 are regular modal trees then T/T_1 is also a regular modal tree.

Proof According to the definition of the quotient, $L_X^{must}(T/T_1)$ and $L_X^{may}(T/T_1)$ are regular as they are built from the regular languages $L_X^{must}(T_1)$, $L_X^{may}(T_1)$, $L_X^{must}(T)$ and $L_X^{may}(T)$ using intersection, union and complement operations.

For instance, for all $X \neq \emptyset$:
$$\begin{array}{rcl} L_X^{must}(T/T_1) & = & \{u \in L(T/T_1) \cap L(T_1) \mid must(T)(u) = X \text{ and} \\ & & must(T_1)(u) = X\} \\ & = & L_X^{must}(T) \cap L_X^{must}(T/T_1) \end{array}$$ ♢

Thus, from definition 5 we deduce the definition of the quotient of modal automata:

Definition 7
The modal automaton $\mathcal{S}/\mathcal{S}_1$ *where* $\mathcal{S} = (Q, q^0, \Delta, must, may)$ *and* $\mathcal{S}_1 = (Q_1, q_1^0, \Delta_1, must_1, may_1)$ *are reduced modal automata defined over* Σ*, is the modal automaton* $((Q \times Q_1) \cup \{\top, \bot\}$, (q^0, q_1^0), $\Delta_/$, $must_/$, $may_/)$ *with:*

- *for the state* $\top$, $must_/(\top) = \emptyset$, $may_/(\top) = \Sigma$ *and* $\Delta_/(\top, a) = \top, \forall a \in \Sigma$
- *for the state* $\bot$, $must_/(\bot) = \Sigma$ *and* $may_/(\bot) = \emptyset$
- *and for each pair* (q, q_1)*:*
  - *if* $a \in must(q) \cap must_1(q_1)$ *then* $a \in must_/((q, q_1))$ *and* $\Delta_/((q, q_1), a) = (q', q_1')$ *with* $q' = \Delta(q, a)$ *and* $q_1' = \Delta_1(q_1, a)$
  - *if* $a \in must(q) \cap \neg must_1(q_1)$ *then* $a \in \neg may_/((q, q_1))$ *and* $\Delta_/((q, q_1), a) = \bot$ *(hence* (q, q_1) *will be removed when reducting* $\mathcal{S}/\mathcal{S}_1$*)*
  - *if* $a \in may(q) \cap must_1(q_1)$ *or* $a \in may(q) \cap may_1(q_1)$ *then* $a \in may_/((q, q_1))$ *and* $\Delta_/((q, q_1), a) = (q', q_1')$ *with* $q' = \Delta(q, a)$ *and* $q_1' = \Delta_1(q_1, a)$
  - *if* $a \in may(q) \cap \neg may_1(q_1)$ *or* $a \in \neg may(q) \cap \neg may_1(q_1)$ *then* $a \in may_/((q, q_1))$ *and* $\Delta_/((q, q_1), a) = \top$
  - *if* $a \in \neg may(q) \cap must(q_1)$ *and* $a \in \neg may(q) \cap may_1(q_1)$ *then* $a \in \neg may_/((q, q_1))$

  The size of the modal automaton $\mathcal{S}/\mathcal{S}_1$ is in $\mathcal{O}(|\mathcal{S}| \times |\mathcal{S}_1|)$.

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 situation where every send message (action *msg*) should be acknowledged either positively (*ack*) or negatively (*nack*). This can't be specified with a modal automaton: *ack* and *nack* can't belong to $must(q)$ because this would request that every message is acknowledged both positively and negatively; *ack* and *nack* can't also belong to $may(q)$ because the automaton that acknowledges no message would be a model of the specification.

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

$$Acc(q) = \{X \in \wp(\Sigma) \text{ st. } must(q) \subseteq X \subseteq may(q)\}$$

By definition this set is closed under union, intersection and convexity (that is if $X, Y \in Acc(q)$ and $X \subseteq Z \subseteq Y$ then $X \cup Y$, $X \cap Y$ and $Z \in Acc(q)$) and may and must modalities may be recovered as $may(q) = \bigcup_{X \in Acc(q)} X$ and $must(q) = \bigcap_{X \in Acc(q)} X$. Notice that this correspondence between may and must modalities and acceptance sets is a biunivoque correspondence but for incoherent state ($must(q) \not\subseteq may(q)$) that are associated with an empty acceptance set ($Acc(q) = \emptyset$) and then "normalized" as $may(q) = \emptyset$ and $must(q) = \Sigma$.

Thus, for example if $may(q) = \{ack, nack\}$ and $must(q) = \emptyset$, we obtain $Acc(q) = \{\emptyset, \{ack\}, \{nack\}, \{ack, nack\}\}$. If we want to specify that at least *ack* or *nack* occur, the specified set of acceptance sets should be $\{\{ack\}, \{nack\}, \{ack,$

nack}} 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 sets closed by union and convexity have been studied in [Hen85]. In the next section, we propose a quotient operation for acceptance automata with no closure constraint over the set of acceptance sets.

4 Improving expressivity with acceptance automata

Definition 8
An acceptance automaton $\mathcal{S}$ *over an alphabet* Σ *is a tuple* $(Q, q^0,$
$\Delta, Acc)$ *where* Q *is a non-empty and finite set of states and* $q^0 \in Q$ *is the initial state. The function* $\Delta : Q \times \Sigma \to Q$ *is a partial function describing a deterministic transition system and* $Acc : Q \to \wp(\wp(\Sigma))$ *is a map associating each state* q *to its set of acceptance sets.*

The definition of the validation relation is the following:

Definition 9
An automaton $\mathcal{C}$ *is a model of an acceptance automaton* $\mathcal{S}$ *if there exists a relation* $\sim \subseteq R \times Q$ *such that:*

- $r^0 \sim q^0$
- $\forall (r, q)$ *such that* $r \sim q$ *:* $out(r) \in Acc(q)$ *and* $\forall a \in out(r), \delta(r, a) \sim \Delta(q, a)$.

Thus, when $r \sim q$ the set of actions that can be performed in r corresponds to one acceptance set of q.

As for modal automata, we first define the quotient operation for acceptance trees. We generalize the framework for modal trees to acceptance trees:

4.1 The framework of acceptance trees

Definition 10
An acceptance tree $T \in AT(\Sigma)$ *is a pair* $T = \langle L, Acc \rangle$ *where* $L \in TD(\Sigma)$ *is a tree domain and* $Acc : L \to \wp(\wp(\Sigma))$ *is a map associating each node* $u \in L$ *to its set of acceptance sets.*

The order relation on acceptance trees is given by both corresponding languages and set of acceptance sets:

$$T_1 \leq T_2 \text{ iff } L(T_1) \subseteq L(T_2) \text{ and } \forall u \in L(T_1), \quad Acc(T_1)(u) \subseteq Acc(T_2)(u)$$

It is a complete lattice where meets are given by intersection of both languages and set of acceptance sets, and joins are given by union of languages and set of acceptance sets if we assume that each map $Acc : \Sigma^* \to \wp(\wp(\Sigma))$ is extended to the whole of Σ^* by letting $Acc(u) = \emptyset$ when $u \notin L$, or with our definition: $Acc(u) = \bigcup_{u \in L_i} Acc_i(u)$. The least element

is given by $L_\perp = \varepsilon$ and $Acc(\varepsilon) = \emptyset$. The greatest element is given by $L_\top = \Sigma^*$ and $Acc_\top(u) = \wp(\Sigma)$ for all $u \in \Sigma^*$.

Any tree domain $L \in TD(\Sigma)$ can be viewed as an acceptance tree with $Acc(u) = out(u)$ that is its set of acceptance sets is a singleton. Now L is a model of an acceptance tree T if:

$$L \models T \text{ iff } \; L \leq T \text{ iff } \; \forall u \in L, \; out(u) \in Acc(T)(u)$$

An acceptance tree is said to be reduced if for all $u \in L$ one has $Acc(u) \neq \emptyset$ and $u \cdot a \in L$ if and only if $u \in L$ and there exists at least one set $X \in Acc(u)$ such that $a \in X$. Thus, the reduction operation for acceptance trees that preserves its set of models consists in iterating the following operations:

1. we remove all states q such that $Acc(q) = \emptyset$ and consequently all transitions $q' \xrightarrow{a} q$ leading to that state;
2. we remove from the set of acceptance sets of a state q all sets containing a letter a for which there is no a-labeled transition stemming from q (i.e. $a \notin out(q)$);
3. for every state q and letter $a \in out(q) \setminus \left(\bigcup_{X \in Acc(q)} X\right)$ we remove the a-labeled transition stemming from q;
4. we remove all states which are no longer accessible from the initial state.

An acceptance tree is said to be regular if L is regular and if for all sets $X \subseteq \Sigma$, the languages $L_X = \{u \in L \mid Acc(u) = X\}$ are regular.

4.2 Quotient of acceptance trees and acceptance automata

First we define the synchronous product of acceptance trees:

Definition 11
The synchronous product of the acceptance trees T_1 and T_2 is the acceptance tree $T_1 \otimes T_2$ over $L(T_1) \cap L(T_2)$ with for all $u \in L(T_1) \cap L(T_2)$:

$$Acc(T_1 \otimes T_2)(u) = \{X_1 \cap X_2 \mid X_1 \in Acc(T_1)(u), X_2 \in Acc(T_2)(u)\}$$

This operation is the adjoint of the following quotient operation:

Definition 12
The quotient of the reduced acceptance trees T and T_1 is the acceptance tree T/T_1 over $\downarrow (L(T) \cup \neg L(T_1))$ with, for all $u \in L(T/T_1) \cap L(T_1)$:

$$Acc(T/T_1)(u) = \{Y \in \wp(\Sigma) \mid \forall X \in Acc(T_1)(u), X \cap Y \in Acc(T)(u)\}$$

and for all $u \in (L(T/T_1) \setminus L(T_1))$, $Acc(T/T_1)(u) = \wp(\Sigma)$.

The proof of prop.6 for acceptance trees is similar to the one for modal trees. Thus we only give the proof of the following key proposition:

Proposition 8
If T, T_1 and T_2 are reduced acceptance trees then $T_1 \otimes T_2 \leq T$ if and only if $T_2 \leq T/T_1$.

Proof First, we prove that $T_1 \otimes T_2 \leq T$ implies $T_2 \leq T/T_1$. According to the definition of the order relation, we have to prove languages and set of acceptance sets inclusion for T_2 and T/T_1. Language inclusion is deduced from the previous remark about prefix-closed languages. Now, we let $u \in L(T_2)$ and $X_2 \in Acc(T_2)(u)$, we prove that the set $X_2 \in Acc(T/T_1)(u)$:

- if $u \in L(T/T_1) \cap L(T_1)$ then $u \in L(T_1) \cap L(T_2)$ and, for all $X_1 \in Acc(T_1)(u)$, $X_1 \cap X_2 \in Acc(T_1 \otimes T_2)(u)$ and, as $T_1 \otimes T_2 \leq T$, $X_1 \cap X_2 \in Acc(T)(u)$. Thus, by the definition of the quotient, $X_2 \in Acc(T/T_1)(u)$;
- if $u \in L(T/T_1) \setminus L(T_1)$ then $Acc(T/T_1)(u) = \wp(\Sigma)$ and $X_2 \in Acc(T/T_1)(u)$.

Secondly we prove that $T_2 \leq T/T_1 \Rightarrow T_1 \otimes T_2 \leq T$.
We let $u \in L(T_1) \cap L(T_2)$ and $X' \in Acc(T_1 \otimes T_2)(u)$, we have to prove that $X' \in Acc(T)(u)$. As $X' \in Acc(T_1 \otimes T_2)(u)$ and $u \in L(T_1) \cap L(T_2)$, there exist a set $X'_1 \in Acc(T_1)(u)$ and $X'_2 \in Acc(T_2)(u)$ such that $X' = X'_1 \cap X'_2$. For all $u \in L(T_2)$, $Acc(T_2)(u) \subseteq Acc(T/T_1)(u)$ hence $X'_2 \in Acc(T/T_1)(u)$. As $u \in L(T/T_1) \cap L(T_1)$, for all $X_1 \in Acc(T_1)(u)$, $X'_2 \cap X_1 \in Acc(T)(u)$. In particular, for $X_1 = X'_1$, we have $X' = X'_2 \cap X'_1 \in Acc(T)(u)$. $\diamondsuit$

The definition of the quotient of acceptance automata is derived from the definition 12 as regular acceptance trees are stable by quotient (the proof is omitted):

Definition 13
*The quotient of the reduced acceptance automata $\mathcal{S} = (Q, q^0, \Delta,$
$Acc)$ and $\mathcal{S}_1 = (Q_1, q_1^0, \Delta_1, Acc_1)$ over the alphabet Σ is the acceptance automata $\mathcal{S}/\mathcal{S}_1 = ((Q \times Q_1) \cup \{\top\}, (q^0, q_1^0), \Delta_/, Acc_/)$ with:*

- $\Delta_/(\top, a) = \top$ *for all* $a \in \Sigma$, *and* $Acc_/(\top) = \wp(\Sigma)$
- $\forall (q, q_1) \in Q \times Q_1$,
  $Acc_/((q, q_1)) = \{Y \in \wp(\Sigma)$ *s.t.* $\forall X \in Acc_1(q_1), X \cap Y \in Acc(q)\}$
- $\forall a \in Acc_/((q, q_1))$,
  $\Delta_/((q, q_1), a) = (q', q'_1)$ *if* $\Delta(q, a) = q'$ *and* $\Delta_1(q_1, a) = q'_1$,
  else $\Delta_/((q, q_1), a) = \top$.

As previously briefly noticed, acceptance automata (AA) strictly subsumes modal automata (MA). Indeed, consider the two following transformations:

Definition 14
Let $\mathcal{S} = (Q, q^0, \Delta, must, may) \in MA$ *and* $\mathcal{S}' = (Q', q^{0\prime}, \Delta',$ $Acc') \in AA$*:*

- $j : MA \to AA$
  $j(\mathcal{S}) = (Q, q^0, \Delta, Acc)$ *with* $Acc(q) = \{X \in \wp(\Sigma) \mid must(q) \subseteq X \subseteq may(q)\}$
- $\Pi : AA \to MA$
  $\Pi(\mathcal{S}') = (Q', q^{0\prime}, \Delta', must', may')$ *with* $\left\{ \begin{array}{lcl} may'(q') & = & \bigcup_{X \in Acc'(q')} X \\ must'(q') & = & \bigcap_{X \in Acc'(q')} X \end{array} \right.$

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

Proposition 9
The quotient operation for modal automata is a particularization of the quotient operation for acceptance automata.

Proof Given $\mathcal{S}$ and $\mathcal{S}_1$ two modal automata, we let $\mathcal{S}'$ be the acceptance automaton obtained by quotienting $j(\mathcal{S})$ and $j(\mathcal{S}_1)$ using definition 13. Then the modal automata $\Pi(\mathcal{S}')$ is identical to the one obtained by quotienting $\mathcal{S}$ and $\mathcal{S}_1$ using definition 7. $\diamondsuit$

5 Conclusion

In this paper, we reduced the problem of behavioral reuse of a component to the computation of a residual specification. We introduced modal automata and acceptance automata as formalisms to specify component behavior. They allow to address restricted forms of liveness.

In [PdAHSV02], reuse of a component is formalized and characterized using a game-theoretic framework: an adaptor is synthesized from a winning strategy of a two-player game between the reused component and the global specification, on one side, and the adaptor, on the other side. When the specification includes a liveness condition, an ω-regular winning condition is taken. However expressivity is limited for specifying component behavior as components are standard finite automata.

Our research now concentrates on enriching our specification formalisms by the addition of properties on states. We will also investigate the existence of the residual operation for a variant of the product operation that correspond to some kind of parallel product rather than the usual sequential composition (because reactive components interact more as coroutines than sequentially composed elements). Moreover modal and acceptance automata are sets equipped with a lattice structure and a monoid structure with a residual operation, adjoint of a commutative product operation i.e. are commutative residuated lattices. We are interested in a more precise characterization of the underlying algebraic structure of the sets of modal and acceptance automata in order to develop the basis of an algebraic theory of components adaptation and reuse.

Acknowledgments

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

References

[AVW03] André Arnold, Aymeric Vincent, and Igor Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303(1):7–34, 2003.

[EFYBvB06] Khaled El-Fakih, Nina Yevtushenko, Sergey Buffalov, and Gregor von Bochmann. Progressive solutions to a parallel automata equation. *Theoretical Computer Science*, 362(1–3):17–32, 2006.

[FP06] Guillaume Feuillade and Sophie Pinchinat. Modal specifications for the control theory of discrete event systems. *J. of Discrete Event Dynamic Systems*, 2006. online first.

[Hen85] Matthew Hennessy. Acceptance trees. *J. ACM*, 32(4):896–928, 1985.

[KNM97] Ratnesh Kumar, Sudhir Nelvagal, and Steven I. Marcus. A discrete event systems approach for protocol conversion. *Discrete Event Dynamic Systems*, 7(3):295–315, 1997.

[Lar89] Kim Guldstrand Larsen. Modal specifications. In *Automatic Verification Methods for Finite State Systems*, pages 232–246, 1989.

[LX90] Kim Guldstrand Larsen and Liu Xinxin. Equation solving using modal transition systems. In *Proceedings of the Fifth Annual IEEE Symp. on Logic in Computer Science, LICS 1990*, pages 108–117. IEEE Computer Society Press, 1990.

[MvB83] Philip Merlin and Gregor von Bochmann. On the construction of submodule specifications and communication protocols. *ACM Trans. on Programming Languages and Systems*, 5(1):1–25, 1983.

[Par87] Joachim Parrow. Submodule construction as equation solving in ccs. *Theoretical Computer Science*, 68:175–202, 1987.

[PdAHSV02] Roberto Passerone, Luca de Alfaro, Thomas A. Henzinger, and Alberto Sangiovanni-Vincentelli. Convertibility verification and converter synthesis: Two faces of the same coin. In *Proceedings of the International Conference on Computer-Aided Design*, 2002.

[YVB+01] Nina Yevtushenko, Tiziano Villa, Robert K. Brayton, Alexandre Petrenko, and Alberto L. Sangiovanni-Vincentelli. Solution of parallel language equations for logic synthesis. In *Proceedings of the International Conference on Computer-Aided Design*, pages 103–110, 2001.