Towards Formalizing QoS of Web Services with Weighted Automata

Pierre-Cyrille Heam, Olga Kouchnarenko, Jérôme Voinot

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Towards Formalizing QoS of Web Services with Weighted Automata

P.-C. Héam, O. Kouchnarenko and J. Voinot

N° 6218

June 2007

THÈMES 4 et 3

ISSN 0249-6399

Towards Formalizing QoS of Web Services with Weighted Automata

P.-C. Héam, O. Kouchnarenko and J. Voinot

Thèmes 4 et 3 — Simulation et optimisation
de systèmes complexes — Interaction homme-machine,
images, données, connaissances
Projets CASSIS

Rapport de recherche n° 6218 — June 2007 — 22 pages

Abstract: Web services (WSs) are used more and more as components of distributed applications with a goal to resolve complex tasks that simple services cannot. This use of WSs is connected to the emergence of languages like WS-BPEL which allows describing the external behaviour of WSs on top of the service interfaces. The use of WSs as components of distributed applications implies the possibility to change a failing service for another which can do at least the same things as the replaced service. The composition issues are also of particular interest to WSs users. Different solutions have been proposed during the last years to check such properties, but, to our knowledge, none of them takes QoS aspects into account. This paper introduces underpinnings and a tool for verifying WSs substitutivity and well-formed composition while considering WSs costs such as the execution time of the different operations provided by WSs.

Key-words: Web Services, Verification, Composition, Quality of Service

This work was partially granted by the ARA-COPS project

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,

Vers une formalisation des services Web par des automates à poids

Résumé : Les services Web sont de plus en plus utilisés comme des composants d'une même application distribuée visant à résoudre une tache complexe que les services seuls ne peuvent effectuer. Cette utilisation des Web services est à liée à la mise en place de langages comme WS-BPEL permettant de décrire le comportement externe d'applications distribuées impliquant la possibilité de remplacer un service défectueux par un autre pouvant faire au moins la même chose. Les problèmes de composition sont aussi particulièrement intéressants pour les utilisateurs de services Web. Différentes solutions ont été proposées ces dernières années pour vérifier de telles propriétés, mais, à notre connaissance, aucune d'elles ne prend en considération les aspects de qualité de service. Cet article introduit les fondements théoriques et un outil pour vérifier la substitutivité des services Web ainsi que leur bonne formation dans une composition, tout cela en prenant en compte des coûts comme le temps d'execution des différentes opérations proposées par les différents services Web.

Mots-clés : Services Web, vérification, composition, qualité de service

1 Introduction

This paper is dedicated to the verification of substitutivity and well-formed composition of Web services while considering a new factor – Quality of Service (QoS).

Web services are increasingly used as components of distributed applications with an aim to resolve complex tasks that simple services cannot. This use of WSs is connected to the emergence of language like WS-BPEL (Web Services Business Process Execution Language, previously known as BPEL4WS, BPEL for short) which allows describing the external behaviour of WSs on top of the service interfaces defined in their WSDL (Web Services Description Language) specifications. With the success of WSs compositions, a new factor – Quality of Service (QoS) – quickly appeared in their definitions [Tia05, D'A06]. However, current Web service standards do not provide mechanisms for specifying services with QoS requirements, such as execution time or financial cost.

To make up for this lack, the starting point of our work and the first contribution of this paper is BPEL and WSDL language extensions including several service cost notions. In [HKV07], we have proposed to extend BPEL with a notion of service costs. In this paper we go further and consider both BPEL and WSDL specifications for being closer to the WSs reality. The main purpose of these extensions is to be able to simply specify QoS aspects of WSs. Moreover, in this paper more verification problems, e. g. strong substitutivity, well-formed composition, etc., are studied for different models, and new decision results are presented, as detailed below.

Following the use of WSs as components, various research orientations have appeared. Of particular interest to us within the framework of this paper are those relating to the checking of properties starting from BPEL and/or WSDL processes, and more especially WSs substitutivity and composition. This research is motivated by the fact that the use of WSs as components of distributed applications requires the possibility for the user to change (possibly dynamically) a failing service for another, which at least performs the same tasks as the replaced service. Different solutions have been proposed during the last years to check such properties. Some of them consist of the translation of BPEL into different classes of automata, like guarded automata in [FBS04], or finite state machines in [BCD$^+$05] and [Fos06]. But, to our knowledge, none of them introduces a model that allows taking service costs of WSs into account. In this direction, the second contribution of the paper is a formal model, called WSs weighted automata (WSWA for short), and a theory underpinning the proposed languages extensions.

Weighted automata – an extension of max-plus automata – is a formalism widely used in computer science for applications in images compression [IvR99, KMT04], speech-to-text proceeding [MPR05, BGW01] or discrete event systems [Gau95]. These large application areas make them intensively studied from the theoretical point of view [Kro94, Web94, HIJ02, KLJP04]. See [BR88] for more detail.

With the increasing importance of QoS in the design of WSs compositions, it is of great interest for users and providers of WSs to be able to say that a WS does the same things as another (possibly failing) service, with comparable/higher quality. This is why this paper gives formal definitions of substitutivity and (strong) well-formed composition problems

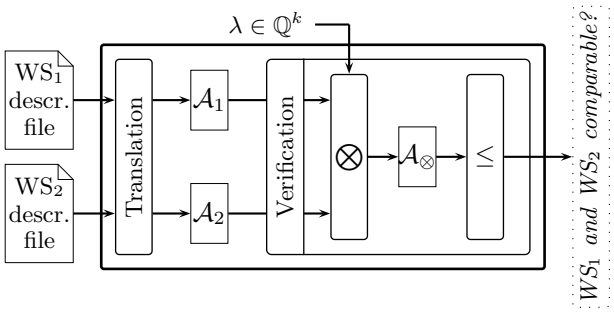


Figure 1: Automata-based WSs verification tool

based on languages inclusion of WSWA, and new decision results for different classes of WSWA. The third contribution of the paper is the verifications proceeding on WSWA thanks to an automatic tool we have been developing. This tool whose structure is given by Fig. 1 allows us to automate the translation from extended BPEL/WSDL specifications to WSWA and to automatically check WSs substitutivity and composition.

The remainder of the paper is organised as follows. Section 2 defines WSWA and substitutivity/composition notions based on them. It introduces extensions of BPEL and WSDL, and explains how these extensions can be translated into WSWA. The verification issues on WSs substitutivity and composition are presented in Section 3, and the operation of the tool support is described in Section 4. Finally, Section 5 gives conclusions and future works.

2 Modelling Web Services and QoS aspects

In this paper WSs and composed WSs are modelled by weighted automata. Section 2.1 introduces the formalism of finite weighted automata and of Web services executions. Section 2.2 explains how to model the substitutivity and the composition properties in this theoretical framework. Finally, Section 2.3 presents BPEL/WSDL extensions to model QoS aspects of WSs, and the computation of weighted automata from extended specifications.

2.1 Modelling Web Services with Finite Weighted Automata

In this paper, Σ denotes a finite set of actions and k is a fixed positive integer. We first introduce the notion of weighted automata.

Definition 1 *A* finite weighted automata $\mathcal{A}$ *is a quintuplet* $\mathcal{A} = (Q, \Sigma, E, I, F)$ *where* Q *is the finite set of states,* $E \subseteq Q \times \Sigma \times \mathbb{Z}^k \times Q$ *is the set of transitions,* $I \subseteq Q$ *is the set of initial states and* $F \subseteq Q$ *is the set of final states.*

Notice that there is a restriction on E: for every action a, every pair of states p, q there exists in E at most one transition of the form (p, a, c, q). For every pair of elements

$c = [c_1, \ldots, c_k], d = [d_1, \ldots, d_k]$ of $\mathbb{Q}^k$ and for every $\otimes \in \{+, -, *, /, \max\}$, we denote by $c \otimes d$ the element $[c_1 \otimes d_1, \ldots, c_k \otimes d_k]$ of $\mathbb{Q}^k$. Now we formally define what is an execution of a weighted automaton and related notions.

Definition 2 *A* partial execution *of a finite weighted automaton $\mathcal{A}$ is a sequence $\pi = (p_0, a_0, c_0, q_0), (p_1, a_1, c_1, q_1), \ldots, (p_n, a_n, c_n, q_n)$ of transitions of $\mathcal{A}$ such that for every $0 \leq i < n$, $q_i = p_{i+1}$. If we add the conditions: p_0 is an initial state, q_n is a final state, then we call π an* execution*. The* label *of the (partial) execution π is the word $a_0 a_1 \ldots a_n$ and the* cost *of the (partial) execution π is the sum of the c_i's: $\mathrm{cost}_{\mathcal{A}}(\pi) = \sum_{i=0}^{n} c_i$.*

The cost *of a word u which is the label of at least one execution of $\mathcal{A}$ is the maximum of all costs of executions of label u:*

$$\mathrm{cost}_{\mathcal{A}}(u) = \max_{\pi \text{ is labelled by } u} \mathrm{cost}(\pi).$$

Notice that since u is finite, there are finitely many path executions labelled by u. The cost of u presented above is so well defined. Classically we denote by $L(\mathcal{A})$ the set of labels of executions of $\mathcal{A}$. We say that two partial executions π_1 and π_2 of two weighted automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are equivalent, denoted $\pi_1 \sim \pi_2$, if they have the same label. This relation is trivially an equivalence relation on partial executions inducing an order relation on weighted automata: we say that $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ if for every execution π of $\mathcal{A}_1$ there exists an equivalent execution π_2 of $\mathcal{A}_2$. Notice that $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ if and only if $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$.

For the rest of the paper, if there is no special mention, all considered automata are finite weighted automata.

2.2 Modelling Substitutivity and Composition Managing QoS Aspects

A problem occurring while managing WSs is to replace a failed service by another. Formally, for two Web services given by their weighted automata $\mathcal{A}_1$ and $\mathcal{A}_2$ the question is to decide whether $\mathcal{A}_2$ can have the same behaviour than $\mathcal{A}_1$ with a similar quality. In this case we say that $\mathcal{A}_2$ can substitute $\mathcal{A}_1$.

Definition 3 *Let λ be a positive element of $\mathbb{Q}^k$, and $\mathcal{A}_1$ and $\mathcal{A}_2$ be two weighted automata. We say that $\mathcal{A}_1$ is* substitutable *by $\mathcal{A}_2$ if for every execution π_1 of $\mathcal{A}_1$ there exists an execution π_2 of $\mathcal{A}_2$ such that $\pi_1 \sim \pi_2$ and $\mathrm{cost}_{\mathcal{A}_2}(\pi_2) \leq \lambda \mathrm{cost}_{\mathcal{A}_1}(\pi_1)$. We say that $\mathcal{A}_1$ is* strongly substitutable *by $\mathcal{A}_2$ if for every execution π_1 of $\mathcal{A}_1$ there exists an execution π_2 of $\mathcal{A}_2$ such that $e_1 \sim e_2$ and for all execution π_2' of $\mathcal{A}_2$ such that $\pi_1 \sim \pi_2'$, $\mathrm{cost}_{\mathcal{A}_2}(\pi_2') \leq \lambda \mathrm{cost}_{\mathcal{A}_1}(\pi_1)$.*

The notion of substitutivity means that a service S_1 can be subsituated by a service S_2 if S_2 has a way to act as S_1 with a higher quality. The notion of strong substitutivity means that a service S_1 can be substituted by a service S_2 if S_2 has a way to act as S_1 and whatever the way chosen by S_2 to act as S_1, its quality is higher.

The *composition* issue is also a central problem in the framework of Web services: considering a list of Web services, the problem is to decide whether they can be associated together

```
<costs>
  <executionTime min="..." max="..." average="..." unit="..." />+
  <financialCost value="..." unit="..." />+
</costs>
```

Figure 2: Syntax of the costs element

in order to provide a complex service required by a client. Composition of Web services are described by BPEL files. We show in the next section how to translate this kind of files into weighted automata. This leads to the following definition of a well-formed composition of Web services.

Definition 4 *Let λ be a positive element of $\mathbb{Q}^k$ and $\mathcal{A}_1$ and $\mathcal{A}_2$ be two automata ($\mathcal{A}_2$ represents a composition of Web services). We say that $\mathcal{A}_2$ is* (strongly) well-formed *with respect to $\mathcal{A}_1$ if $\mathcal{A}_1$ is* (strongly) substitutable *by $\mathcal{A}_2$.*

Notice that in the definitions we choose that $\text{cost}(\pi_2) \leq \lambda\text{cost}(\pi_1)$ modelling that the lower is the cost the better is the service, what is intuitive for connection time or financial cost. One can give a dual definition if the lower is the cost the worse is the service by changing $\text{cost}(\pi_2) \leq \lambda\text{cost}(\pi_1)$ in $\text{cost}(\pi_2) \geq \lambda\text{cost}(\pi_1)$. All notions, algorithms, etc. described in this paper may be trivially adapted to this dual definition. In order to not overload the reader, we do not consider that case.

2.3 From WSDL and BPEL to Finite Weighted Automata

Unfortunately current Web service standards do not provide mechanisms for specifying WSs with QoS requirements, such as execution time or financial cost. In [HKV07], we have proposed to extend BPEL with a notion of service costs. In this paper we go further and extend both BPEL and WSDL specifications. BPEL files are produced by WSs users while WSDL files are given WSs providers. Adding service costs in WSDL enables joining a WSs audit framework, and makes it possible to set dynamically service costs, and to produce new extended WSDL files from composed WSs to be able to provide those like new WSs.

In this section, QoS aspects are first introduced in WSs specifications, and second the new extended descriptions are translated into WSs weighted automata by extracting behaviours/costs from extended BPEL specifications and extended WSDL interface files.

2.3.1 Introducing QoS Aspects in Web Services Description

Our proposal consists in defining a description element called costs. As shown in Fig. 2, each costs element may contain several elements which represent the different QoS aspects to be taken into account. The types of the service costs that can be specified, for the moment, are:

- executionTime, the minimal, maximal and average execution time of an operation expressed in number of time units (seconds, milliseconds, ...).
- financialCost, the financial cost of the execution of an operation expressed in number of monetary units (USD, EUR, JPY, ...).

This list can be easily extended by other valuable operation costs. In comparison with [HKV07], the QoS requirements we propose to consider here are more advanced. Indeed,

- the executionTime element allows considering time intervals in addition to the average execution time of an operation;
- the costs element can be added in BPEL (extension of invoke, receive and reply BPEL activities and of onMessage BPEL activity element) and in WSDL (extension of operation WSDL element).

This last point directly concerns the applicability of our approach in a Web services audit framework. Indeed, BPEL files being produced by Web services users, they can be modified by those. However, for confidence reasons, WSDL files should be modified by an audit entity rather than Web services providers which produce them. These considerations are not in the scope of this paper.

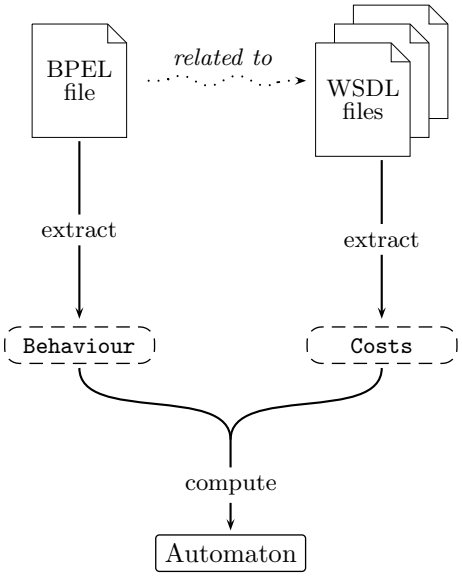
To compute a WSWA representing a Web service from its specification, the translation has to do:

- The extraction of the Web service behaviours to define the automaton transitions thanks to the basic activities (invoke, receive, ...), and the sequences of the automaton transitions thanks to the structured activities (sequence, while, ...) used in the corresponding BPEL description.
- The extraction of the costs of each Web service operation to define the cost of each automaton transition from the costs element corresponding to the transition operation.

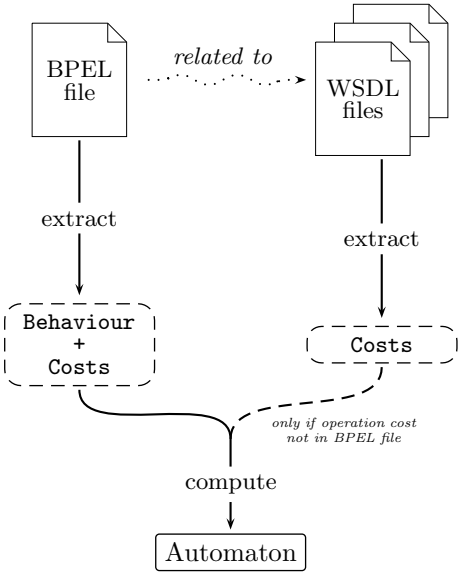
Though separated, these two parts are closely related since there are two possibilities to deal with costs in WSs specifications. In the case when the costs are only given in the WSDL files related to the considered BPEL file, for each transition its cost is defined from the operation costs element of the corresponding WSDL files. Otherwise, i.e. when the costs are given in WSDL and/or BPEL descriptions files (see Fig. 3), we give top priority to the values in the BPEL description because it is defined by the Web service user.

While extracting the Web service behaviours, the two cases above have no impact on the result of the translation. Because of lack of space, we only give the translation rules for invoke BPEL activities. Others translations are similar (see `http://lifc.univ-fcomte.fr/~heampc/wiselong.pdf` for more detail).

The invocation of a Web service operation can be synchronous or asynchronous. Consequently, there are two different rules for the translation of an invoke activity into weighted automata. In an extended BPEL specification, an asynchronous invoke activity is of the following form:



(a) Costs in WSDL description files



(b) Costs in WSDL and BPEL description files

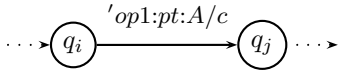
Figure 3: Translation process schema

```
<invoke partnerLink="A" portType="pt" operation="op1" inputVariable="x">
  <costs>
    <financialCost value="20" unit="EUR" />
  </costs>
</invoke>
```

In the corresponding WSDL description file, the called operation is as follows.

```
<portType name="pT">
  <operation name="op1">
    <input message="msgType1" />
    <output message="msgType2" />
    <costs>
      <executionTime min="0.3" max="1.2" average="0.9" unit="milliseconds" />
      <financialCost value="30" unit="EUR" />
    </costs>
  </operation>
  ...
</portType>
```

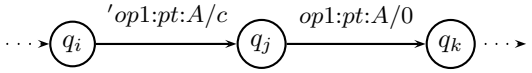
>From the forms above the following piece of WSWA is generated where c is a vector of the cost element values from WSDL or BPEL files.



For the example above, the vector c contains the values from the WSDL description for the execution times of the operation, and from the BPEL specification for the financial cost. A synchronous invoke activity is of the following form:

```
<invoke partnerLink="A" portType="pt" operation="op2" inputVariable="x" outputVariable="y">
  <costs>
    <executionTime min="0.3" max="0.8" average="0.5" unit="milliseconds" />
    <financialCost value="20" unit="EUR" />
  </costs>
</invoke>
```

Notice that this operation is linked to the same WSDL description file as the one above. From this activity definition we generate the following piece of weighted automata where c is the vector containing cost element values.



In this case the vector c contains only the values specified in the BPEL specification in order to give top priority to the user. The second transition have the nil cost since we consider the return of an invocation to be free.

3 Verifying Substitutivity and Composition

According to Definitions 3 and 4, once we have modelled Web services by weighted automata, the formal algorithmic problem for substitutivity and well-formation on one hand, and for strong substitutivity and strong well-formation are identical. Let λ be a positive element of $\mathbb{Q}^k$ and $\mathbf{C_1}$ and $\mathbf{C_2}$ be two classes of finite weighted automata. We define the **generic substitutivity/well-formed composition** problems by $P^{\lambda}[\mathbf{C_1}, \mathbf{C_2}]$ by:

Input: Two automata $\mathcal{A}_1 \in \mathbf{C_1}$ and $\mathcal{A}_2 \in \mathbf{C_2}$.
Question: for every execution π_1 of $\mathcal{A}_1$ does there exist an execution π_2 of $\mathcal{A}_2$ such that $\pi_1 \sim \pi_2$ and $\text{cost}_{\mathcal{A}_2}(\pi_2) \leq \lambda \text{cost}_{\mathcal{A}_1}(\pi_1)$?

Similarly, we define the generic problem $P^{\lambda}_{\text{strong}}[\mathbf{C_1}, \mathbf{C_2}]$ related to the **strong substitutivity/strong well-formed composition**:

Input: Two automata $\mathcal{A}_1 \in \mathbf{C_1}$ and $\mathcal{A}_2 \in \mathbf{C_2}$.
Question: for every execution π_1 of $\mathcal{A}_1$ does there exist an execution π_2 of $\mathcal{A}_2$ such that $\pi_1 \sim \pi_2$ and for every π_2' of $\mathcal{A}_2$ such that $\pi_1 \sim \pi_2'$, $\text{cost}_{\mathcal{A}_2}(\pi_2') \leq \lambda \text{cost}_{\mathcal{A}_1}(\pi_1)$?

3.1 Preliminary Background and Notation on Weighted Automata

Let $\mathcal{A}_1 = (Q_1, \Sigma, E_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, E_2, I_2, F_2)$ be two automata. The *product* of $\mathcal{A}_1$ by $\mathcal{A}_2$ is the automaton on Σ, denoted $\mathcal{A}_1 \times \mathcal{A}_2$, whose set of states is $Q_1 \times Q_2$, set of initial states is $I_1 \times I_2$, set of final states $F_1 \times F_2$ and set of transition is
$\left\{\big((p_1, p_2), a, c_1 + c_2, (q_1, q_2)\big) \mid (p_1, a, c_1, q_1) \in E_1,\ (p_2, a, c_2, q_2) \in E_2,\ a \in \Sigma\right\}$.
It is well-known that $L(\mathcal{A}_1 \times \mathcal{A}_2) = L(\mathcal{A}_1) \times L(\mathcal{A}_2)$. Let $\mathcal{A} = (Q, \Sigma, E, I, F)$ be an automaton and $\alpha \in \mathbb{Q}^k$. We define the automaton $\alpha\mathcal{A}$ obtained from $\mathcal{A}$ by multiplying all transition costs by α; formally $\alpha\mathcal{A} = (Q, \Sigma, E', I, F)$ with $E' = \{(p, a, \alpha c, q) \mid (p, a, c, q) \in E\}$.

An automaton is *deterministic* if it has a unique initial state and if for every state and every action there exists at most one transition starting from this state and labelled by this action. The product of two deterministic automata is also deterministic. An automaton is *ℓ-ambiguous* ($\ell \in \mathbb{N}$) if for every accepted word u there exists at most ℓ executions labelled by u. A 1-ambiguous automaton is called *unambiguous*. Deterministic automata are trivially unambiguous. An automaton is *finitely ambiguous* if there exists an integer ℓ such that the automaton is ℓ-ambiguous. An automaton is *uniformly weighted* if for every pair (p_1, a, c_1, q_1) and (p_2, a, c_2, q_2) of transitions labelled by the same action, $c_1 = c_2$.

We denote by **WA** the class of all finite weighted automata. We also denote by **D** (resp. **UA**) (resp. **FA**) (resp. **UW**) the subclasses of **WA** of deterministic (resp. non-ambiguous) (resp. finitely ambiguous) (resp. uniformly weighted) weighted automata. One has: $\mathbf{D} \subseteq \mathbf{UA} \subseteq \mathbf{FA} \subseteq \mathbf{WA}$.

We finish this section by recalling some results on decision procedures for finite (weighted) automata.

Theorem 5 *Given an integer λ and two weighted automata $\mathcal{A}_1$ and $\mathcal{A}_2$, it is*

- *undecidable to test whether for every* $u \in L(\mathcal{A}_1)$, $\text{cost}_{\mathcal{A}_1}(u) \leq \lambda \text{cost}_{\mathcal{A}_2}(u)$ *[Kro94]; same problem is decidable if* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are both finitely ambiguous [HIJ02, Web94],*
- *undecidable to test whether for every* $u \in L(\mathcal{A}_1)$*, there exists an execution* π *of label* u *in* $\mathcal{A}_1$ *such that* $\text{cost}_{\mathcal{A}_1}(\pi) \geq 0$ *(resp.* $\text{cost}_{\mathcal{A}_1}(\pi) \leq 0$*) [Kro94],*
- *decidable in polynomial time to test whether for every* $u \in L(\mathcal{A}_1)$, $\text{cost}_{\mathcal{A}_1}(u) \leq \lambda \text{cost}_{\mathcal{A}_2}(u)$ *if* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are both finitely ambiguous [HIJ02, Web94],*
- *decidable in polynomial time to test whether* $\mathcal{A}_1$ *is in* **UA** *or if* $\mathcal{A}_1$ *is in* **FA** *[WS91].*
- *PSPACE-complete to decide whether* $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ *[AHU74].*

3.2 On the Complexity of the $P^{\lambda}_{\text{strong}}[\mathbf{C}_1, \mathbf{C}_2]$ and $P^{\lambda}[\mathbf{C}_1, \mathbf{C}_2]$ Problems

Set $\lambda = \frac{x}{y}$ where x and y are two positive relatively prime numbers. $\mathbf{C_1}$ and $\mathbf{C_2}$ be two classes of finite weighted automata. Omitted proofs are given at `http://lifc.univ-fcomte.fr/∼heampc/wiselong.pdf`.

Theorem 6 *Two automata* $\mathcal{A}_1 \in \mathbf{C}_1$ *and* $\mathcal{A}_2 \in \mathbf{C_2}$ *satisfy the problem* $P^{\lambda}_{\text{strong}}[\mathbf{C_1}, \mathbf{C_2}]$ *if and only if they satisfy* $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ *and for every path execution* π *of* $x\mathcal{A}_1 \times (-y\mathcal{A}_2)$, $\text{cost}(\pi) \geq 0$.

Corollary 7 *The problems* $P^{\lambda}_{\text{strong}}[\mathbf{WA}, \mathbf{WA}]$ *and* $P^{\lambda}_{\text{strong}}[\mathbf{WA}, \mathbf{D}]$ *are respectively PSPACE-complete and polynomial. The problem* $P^{\lambda}_{\text{strong}}[\mathbf{UA}, \mathbf{UA}]$ *is polynomial too.*

Using the inclusions $\mathbf{D} \subseteq \mathbf{UA} \subseteq \mathbf{FA} \subseteq \mathbf{WA}$, previous results on the complexity of $P^{\lambda}_{\text{strong}}[\mathbf{C_1}, \mathbf{C_2}]$ may be synthesised in the following table.

$P^{\lambda}_{\text{strong}}[\mathbf{C_1}, \mathbf{C_2}]$	$\mathbf{C_1} = \mathbf{D}$	$\mathbf{C_1} = \mathbf{UA}$	$\mathbf{C_1} = \mathbf{FA}$	$\mathbf{C_1} = \mathbf{WA}$
$\mathbf{C_2} = \mathbf{D}$	P	P	P	P
$\mathbf{C_2} = \mathbf{UA}$	P	P	P	PSPACE
$\mathbf{C_2} = \mathbf{FA}$	P	P	P	PSPACE
$\mathbf{C_2} = \mathbf{WA}$	PSPACE	PSPACE	PSPACE	PSPACE-complete

Theorem 8 *The problem* $P^{\lambda}[\mathbf{D}, \mathbf{WA}]$ *is undecidable.*

Proof. Let $\mathcal{A}$ be in **WA**. Let $\mathcal{A}_1$ be the automaton of **D** obtained from the minimal unweighted automaton of $L(\mathcal{A})$ by adding the nil weight on each transition. By construction, one has $\mathcal{A}_1 \sqsubseteq \mathcal{A}$.

We claim that $\mathcal{A}_1$ and $\mathcal{A}$ satisfy $P^{\lambda}[\mathbf{D}, \mathbf{WA}]$ if and only if for every $u \in L(\mathcal{A})$, there exists an execution π of label u in $\mathcal{A}$ such that $\text{cost}_{\mathcal{A}}(\pi) \leq 0$.

- Assume first that $\mathcal{A}_1$ and $\mathcal{A}$ satisfy $P^{\lambda}[\mathbf{D}, \mathbf{WA}]$. Let $u \in L(\mathcal{A})$. Since $L(\mathcal{A}_1) = L(\mathcal{A})$, there exists an execution π_1 in $\mathcal{A}_1$ of label u. Furthermore, by construction of $\mathcal{A}_1$, $\text{cost}_{\mathcal{A}_1}(\pi_1) = 0$. Now, since $\mathcal{A}_1$ and $\mathcal{A}$ satisfy $P^{\lambda}[\mathbf{D}, \mathbf{WA}]$, there exists an execution π in $\mathcal{A}$ such that $\text{cost}_{\mathcal{A}}(\pi) \leq 0$.

- Assume now that $\mathcal{A}_1$ and $\mathcal{A}$ are such that for every $u \in L(\mathcal{A})$, there exists an execution π of label u in $\mathcal{A}$ with $\text{cost}_{\mathcal{A}}(\pi) \leq 0$. Let π_1 be an execution of $\mathcal{A}_1$, and u_1 the label of π_1. By hypothesis, there exists an execution π in $\mathcal{A}$ of label u_1 such that $\text{cost}_{\mathcal{A}}(\pi) \leq 0$. Since $\text{cost}_{\mathcal{A}_1}(\pi_1) = 0$ by construction, $\mathcal{A}_1$ and $\mathcal{A}$ satisfy $P^{\lambda}[\mathbf{D}, \mathbf{WA}]$, proving the claim.

It follows that if $P^{\lambda}[\mathbf{D}, \mathbf{WA}]$ is decidable, then the problem to decide whether for every $u \in L(\mathcal{A})$, there exists an execution π of label u in $\mathcal{A}$ such that $\text{cost}_{\mathcal{A}}(\pi) \leq 0$ is decidable; a contradiction (Theorem 5). □

Proposition 9 *One has* $P^{\lambda}[\mathbf{C_1}, \mathbf{D}] = P^{\lambda}_{\text{strong}}[\mathbf{C_1}, \mathbf{D}]$ *and* $P^{\lambda}[\mathbf{C_1}, \mathbf{UA}] = P^{\lambda}_{\text{strong}}[\mathbf{C_1}, \mathbf{UA}]$

Proposition 10 *The problem* $P^{\lambda}[\mathbf{WA}, \mathbf{FA}]$ *is PSPACE while the problem* $P^{\lambda}[\mathbf{FA}, \mathbf{FA}]$ *is decidable in polynomial time.*

Using the inclusions $\mathbf{D} \subseteq \mathbf{UA} \subseteq \mathbf{FA} \subseteq \mathbf{WA}$, the previous results on the complexity of $P^{\lambda}[\mathbf{C_1}, \mathbf{C_2}]$ may be synthesised in the following table.

$P^{\lambda}[\mathbf{C_1}, \mathbf{C_2}]$	$\mathbf{C_1} = \mathbf{D}$	$\mathbf{C_1} = \mathbf{UA}$	$\mathbf{C_1} = \mathbf{FA}$	$\mathbf{C_1} = \mathbf{WA}$
$\mathbf{C_2} = \mathbf{D}$	P	P	P	P
$\mathbf{C_2} = \mathbf{UA}$	P	P	P	PSPACE
$\mathbf{C_2} = \mathbf{FA}$	P	P	P	PSPACE
$\mathbf{C_2} = \mathbf{WA}$	undecidable	undecidable	undecidable	undecidable

If we consider weighted automata generated by WSDL files, with no moderation by BPEL specifications, all considered automata are in **UW**. Indeed, WSDL files describes each action with a cost. This cost does not depend on where the action is called in the composed service.

Proposition 11 *For every pair of classes of automata* $\mathbf{C_1}, \mathbf{C_2}$*, the problems* $P^{\lambda}[\mathbf{C_1} \cap \mathbf{UW}, \mathbf{C_2} \cap \mathbf{UW}]$ *and* $P^{\lambda}_{\text{strong}}[\mathbf{C_1} \cap \mathbf{UW}, \mathbf{C_2} \cap \mathbf{UW}]$ *are equals.*

So using the results of Section 3.2, one has:

$P^{\lambda}[\mathbf{C_1} \cap \mathbf{UW}, \mathbf{C_2} \cap \mathbf{UW}]$	$\mathbf{C_1} = \mathbf{D}$	$\mathbf{C_1} = \mathbf{UA}$	$\mathbf{C_1} = \mathbf{FA}$	$\mathbf{C_1} = \mathbf{WA}$
$\mathbf{C_2} = \mathbf{D}$	P	P	P	P
$\mathbf{C_2} = \mathbf{UA}$	P	P	P	PSPACE
$\mathbf{C_2} = \mathbf{FA}$	P	P	P	PSPACE
$\mathbf{C_2} = \mathbf{WA}$	PSPACE	PSPACE	PSPACE	PSPACE-complete

4 Implementation

A tool for verifying different kinds of substitutivity based on weighted automata has been implemented. Its structure is shown in Fig. 1.

The translation module is used to translate extended BPEL specifications and WSDL descriptions into weighted automata. Applied to the result of the specification files parsing, the corresponding algorithm is based on the set of rules introduced in Section 2.3.1. This algorithm is a recursive algorithm in which the automaton is built activities after activities.

The verification module is then used to verify substitutivity between two WSs following Definition 3. The inputs of this module are the WSWA produced by the translation module. First, an abstraction of each automaton is computed. This abstraction consists in removing the τ-transitions covering the assign and empty activities. Those transitions can be removed without loosing behaviours since they do not take part in services exchanges. Second, the verification process starts by the computation of the product of the two abstract automata. Third, all the verification algorithms are applied to the result of this product. Those algorithms are based on well-known graph theory algorithms like depth-first search or minimum spanning tree algorithms.

This implementation has been tested on a Dell Latitude D600 with an Intel Pentium M 1.4Ghz and 512Mo of RAM. The tests have been achieved on different versions of several examples like a book store example provided by Oracle[1] or the classical loan approval example. Different versions of those examples have been developed in order to test the different possibilities to compute automata from description files. The following table shows some examples of the approximative computing time needed for building automata from BPEL and WSDL description files.

Example	Costs location	State space	Time to build	Time to check
Loan approval	WSDL	10	$<$ 1 sec.	$<$ 1 sec.
Loan approval	WSDL + BPEL	10	$<$ 1 sec.	$<$ 1 sec.
Book store	WSDL	50	$<$ 3 sec.	$<$ 1 sec.
Book store	WSDL + BPEL	50	$<$ 3 sec.	$<$ 1 sec.

The increase of the time to build automata is due to the presence of flow activities to put in parallel more activities in the book store example than in the loan approval example, and not to the size of the state space of the produced automaton. Also, this table shows that the costs location is not a factor to increase the time. Currently the verification module allows us to check whether two WSs are substitutable following Definition 3. The table presents the time needed to check the WSs substitutivity on some of the handled examples.

5 Conclusion

Awaiting a new standard including QoS aspects, this paper presented an approach to model WSs, provided as extended BPEL/WSDL specifications, and to decide the substitutivity and composition problems while considering QoS aspects. The tool developed enables us to automate the different steps of the verification process from extended BPEL4WS/WS-BPEL specifications or from extended WSDL interface files.

[1] `http://www.oracle.com/technology/pub/articles/matjaz_bpel2.html`

In the future we intend, on one hand, to extend WSWA with guards on transitions, close to what is done in [FBS04]. This extension should enable modelling WSs more finely. In addition, this should allow us to consider conditional service costs. By conditional service costs we mean service costs which are dependent of the message (or the number of messages) received by the concerned WS. These evolutions should make it possible to use our framework and the dedicated tool in the realistic context with the proviso that it is possible to discern two functionally equivalent services (i.e. to rename operations which performs the same task to obtain equivalent WSWA).

On the other hand, we plan to consider other relations between WSWA, e.g. semi-commutative relations or different kinds of simulation relations. Unfortunately, within these settings the problems of the substitutivity/well-formation might become undecidable.

We are grateful to Ines Klimann and to Sylvain Lombardy for their help.

References

[AHU74] A. Aho, J. Hopcroft, and J. Ullman. *The design and analysis of computer algorithms*, pages 395–400. Addison-Wesley, 1974.

[BCD+05] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic Composition of Transition-based Semantic Web Services with Messaging. In *VLDB'05, Trondheim, Norway*, 2005.

[BGW01] A.L. Buchsbaum, R. Giancarlo, and J. Westbrook. An approximate determinization algorithm for weighted finite-state automata. *Algorithmica*, 30(4):503–526, 2001.

[BR88] J. Berstel and Ch. Reutenauer. *Rational Series and Their Languages*. Springer-Verlag, 1988.

[D'A06] A. D'Ambrogrio. A Model-driven WSDL Extension for Describing the QoS of Web Services. In *ICWS'06, Chicago, Illinois, USA*, 2006.

[FBS04] X. Fu, T. Bultan, and J. Su. Analys of Interacting BPEL Web Services. In *WWW'04, New York, NY, USA*, 2004.

[Fos06] H. Foster. *A Rigourous Approach to Engineering Web Service Compositions*. PhD thesis, Imperial College London, 2006.

[Gau95] S. Gaubert. Performance Evaluation of (max,+) Automata. *IEEE Trans. on Automatic Control*, 40(12), 1995.

[HIJ02] K. Hashiguchi, K. Ishiguro, and S. Jimbo. Decidability of the Equivalence Problem for Finitely Ambiguous Finance Automata. *IJAC*, 12(3), 2002.

[HKV07] P.-C. Héam, O. Kouchnarenko, and J. Voinot. How to Handle QoS Aspects in Web Services Substitutivity Verification. In *WETICE'07,*, 2007.

[HU80] J. Hopcroft and J. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley, 1980.

[IvR99] K. Culik II and P.C. von Rosenberg. Generalized weighted finite automata based image compression. *J. UCS*, 5(4):227–242, 1999.

[KLJP04] I. Klimann, S. Lombardy, J., and Ch. Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *TCS*, 327(3):349–373, 2004.

[KMT04] F. Katritzke, W. Merzenich, and M. Thomas. Enhancements of partitioning techniques for image compression using weighted finite automata. *TCS*, 313(1):133–144, 2004.

[Kro94] D. Krob. The Equality Problem for Rational Series with Multiplicities in the Tropical Semiring is Undecidable. *IJAC*, 4(3), 1994.

[MPR05] M. Mohri, F. Pereira, and M. Riley. Weighted automata in text and speech processing. March 28 2005.

[Tia05] Min Tian. *QoS integration in Web services with the WS-QoS framework*. PhD thesis, Freie Universitat Berlin, 2005.

[Web94] A. Weber. Finite-valued Distance Automata. *TCS*, 134, 1994.

[WS91] A. Weber and H. Seidl. On the degree of ambiguity of finite automata. *TCS*, 88(2):325–349, 1991.

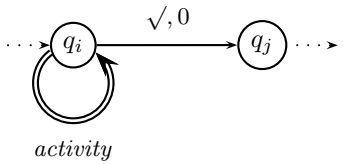
Annexes

6 Translation rules

Translation rule for a recursive call of activities. This rule corresponds to the translation of a while activity into cost automata. In a BPEL specification, a while activity is of the following form:

<while condition="*booleanCondition*">
  activity
</while>

where *activity* can be a basic or structured activity like a sequence of activities. >From this activity definition we generate the following pieces of weighted automata:

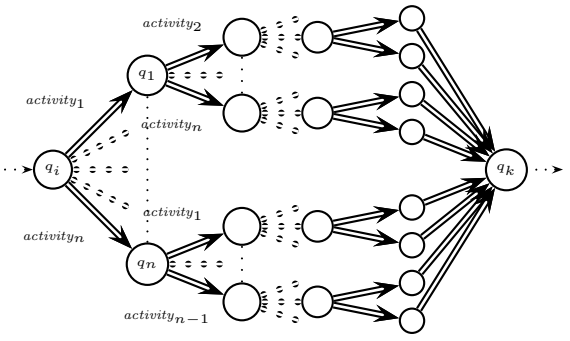


The transition labelled with $\surd$ represent the exit of the loop. This transition is introduced to verify some properties easier while allowing to detect the state which is the origin of a loop.

Translation rule for parallel activities. This rule corresponds to the translation of a flow activity into cost automata. In a BPEL specification, a flow activity is of the following form:

<flow>
  activity$_1$
  activity$_2$
  ...
  activity$_n$
</flow>

where *activity*$_1$, *activity*$_2$, ..., *activity*$_n$ can be basic or structured activities. >From this activity definition we generate the following piece of weighted automata:



Notice that this translation rule produces a piece of WSWA containing $(n*((n-1)!+n))+2$ states where n is the number of activities. This number of states is reduced if some links between activities are defined thanks to the source and target elements of BPEL activities.

7 Omitted proofs

7.1 Proof of Theorem 6

Proof. The proof will is divided into two parts.

- Assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $P^{\lambda}_{\text{strong}}$. By definition, for every execution of $\mathcal{A}_1$ there exists an equivalent execution in $\mathcal{A}_2$. Thus $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$. Consider now an execution π in $x\mathcal{A}_1 \times (-y\mathcal{A}_2)$,

$$\pi = (\overline{p_0}, a_1, \alpha_1, \overline{p_1}), (\overline{p_1}, a_2, \alpha_2, \overline{p_2}) \dots (\overline{p_{n-1}}, a_n, \alpha_n, \overline{p_n}).$$

  By definition of $x\mathcal{A}_1 \times (-y\mathcal{A}_2)$, there exist $p_0, p_1, \dots, p_n$ states of $\mathcal{A}_1$, $q_0, q_1, \dots, q_n$ states of $\mathcal{A}_2$, integers $c_1, c_2, \dots, c_n, d_1, d_2, \dots, d_n$ such that

  - $\pi_1 = (p_0, a_1, c_1, p_1), (p_1, a_2, c_2, p_2), \dots, (p_{n-1}, a_n, c_n, p_n)$ is an execution in $\mathcal{A}_1$,
  - $\pi_2 = (q_0, a_1, d_1, q_1), (q_1, a_2, d_2, q_2), \dots, (q_{n-1}, a_n, d_n, q_n)$ is an execution in $\mathcal{A}_2$,
  - for every $1 \leq i \leq n$, $\alpha_i = xc_i - yd_i$,
  - for every $0 \leq i \leq n$, $\overline{p_i} = (p_i, q_i)$.

  One has $\pi_1 \sim \pi_2$. Therefore, since $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $P^{\lambda}_{\text{strong}}$, the following inequality holds:

$$\sum_{i=1}^{n} d_i \leq \frac{x}{y} \sum_{i=1}^{n} c_i.$$

  Consequently,

$$\sum_{i=1}^{n} \alpha_i \geq 0.$$

- Assume now that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ and for every execution π of $(x\mathcal{A}_1 \times (-y\mathcal{A}_2))$, $\text{cost}(\pi) \geq 0$.

  Since $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$, for every execution in $\mathcal{A}_1$ there exists an equivalent execution in $\mathcal{A}_2$.

  Finally consider two executions

  $$\pi_1 = (p_0, a_1, c_1, p_1), (p_1, a_2, c_2, p_2), \ldots, (p_{n-1}, a_n, c_n, p_n)$$

  in $\mathcal{A}_1$ and

  $$\pi_2 = (q_0, a_1, d_1, q_1), (q_1, a_2, d_2, q_2), \ldots, (q_{n-1}, a_n, d_n, q_n)$$

  in $\mathcal{A}_2$ such that $\pi \sim \pi_2$.

  By definition there exists an execution π in $y\mathcal{A}_1 \times (-x\mathcal{A}_2)$,

  $$\pi = (\overline{p_0}, a_1, \alpha_1, \overline{p_1}), (\overline{p_1}, a_2, \alpha_2, \overline{p_2}) \ldots (\overline{p_{n-1}}, a_n, \alpha_n, \overline{p_n}).$$

  such that

  - for every $1 \leq i \leq n$, $\alpha_i = yc_i - xd_i$,
  - for every $0 \leq i \leq n$, $\overline{p_i} = (p_i, q_i)$.

  Moreover, by hypotheses, one has $\text{cost}(\pi) \geq 0$:

  $$\text{cost}(\pi) = \sum_{i=1}^{n} \alpha_i \geq 0.$$

  Consequently,

  $$\sum_{i=1}^{n} d_i \leq \frac{x}{y} \sum_{i=1}^{n} c_i.$$

  It follows that $\text{cost}_{\mathcal{A}_2}(\pi_2) \leq \lambda \text{cost}_{\mathcal{A}_1}(\pi_1)$, which concludes the proof.

□

7.2 Proof of Corollary 7

Proof. Deciding whether $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ is equivalent to decide whether $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ which is classically PSPACE-complete[AHU74] in the general case and polynomial if either $\mathcal{A}_2$ is deterministic [HU80] or if both $\mathcal{A}_1$ and $\mathcal{A}_2$ are finitely ambiguous [Web94]. Now deciding whether for every execution π of $c_2\mathcal{A}_1 \times (-c_1\mathcal{A}_2)$, $\text{cost}(\pi) \geq 0$ is a classical polynomial problem on weighted graphs which can be solved for instance by Bellman-Ford algorithm (see any graph algorithms book).

The PSPACE-completeness of the general case is trivially obtained by Theorem 6 using automata with zero weights and the PSPACE-completeness of testing whether $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$. □

7.3 Proof of Proposition 9

PROOF. Since for every word $u \in L(\mathcal{A}_2)$ there is a unique execution in $\mathcal{A}_2$ labelled by π, the result is obvious. □

7.4 Proof of Proposition 10

PROOF. Let $\mathcal{A}_1 = (Q_1, \Sigma, E_1, I_1, F_1) \in \mathbf{WA}$ and $\mathcal{A}_2 = (Q_2, \Sigma, E_2, I_2, F_2) \in \mathbf{FA}$. Set $\mathcal{A}_3 = (Q_1, \Sigma \times Q_1 \times Q_1, E_3, I_1, F_1)$ and $\mathcal{A}_4 = (Q_2, \Sigma \times Q_1 \times Q_1, E_4, I_2, F_2)$ where:

- $E_3 = \{(p, [a, p, q], c, q) \mid (p, a, c, q) \in E_1\}$,
- $E_4 = \{(p, [a, r, s], c, q) \mid (p, a, c, q) \in E_2,\ \exists x \in \mathbb{Q}^k,\ (r, a, x, s) \in E_1\}$.

Notice that $\mathcal{A}_3$ is unambiguous and that $\mathcal{A}_4$ are finitely ambiguous. Indeed if $u = [a_1, q_1, q_2][a_2, q_2, q_3] \dots [a_n, q_n, q_{n+1}]$ is accepted by $\mathcal{A}_3$, then there is a unique execution $(q_1, a_1, c_1, q_2) \dots (q_n, a_n, c_n, q_{n+1})$ labelled by u (remind that in Section 2.1 that for each pair of states and each action there exists at most one transition between these two states label by this action). Now assume that $\mathcal{A}_2$ is ℓ-ambiguous and that the word $u = [a_1, q_1, q_2][a_2, q_2, q_3] \dots [a_n, q_n, q_{n+1}]$ is accepted by $\mathcal{A}_4$. Since there are at most ℓ executions in $\mathcal{A}_2$ accepting $a_1 a_2 \dots a_n$, there is at most $\ell \mathrm{card}(Q_1)^2$ executions in $\mathcal{A}_4$ accepting u. Thus $\mathcal{A}_4$ is finitely ambiguous.

Let $\mathcal{B} = x\mathcal{A}_3 \times (-y\mathcal{A}_4)$.

We claim that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $P^\lambda[\mathbf{WA}, \mathbf{FA}]$ if and only if $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ and for every $u \in L(\mathcal{B})$, there exists an execution π in $\mathcal{B}$ such that $\mathrm{cost}_{\mathcal{B}}(\pi) \geq 0$.

($\Rightarrow$) Assume first that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $P^\lambda[\mathbf{WA}, \mathbf{FA}]$. Then $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$. Now let $u \in L(\mathcal{B})$. This part of the proof is represented on Fig. 7.4.

By definition of the product, one also has $u \in L(\mathcal{A}_3)$. Consequently, there exists an execution π_3 in $\mathcal{A}_3$ of label u of the form

$$\pi_3 = (q_1, [a_1, q_1, q_2], c_1, q_2), (q_2, [a_2, q_2, q_3], c_2, q_3) \dots (q_n, [a_n, q_n, q_{n+1}], c_n, q_{n+1}).$$

Consequently, by construction of $\mathcal{A}_3$,

$$\pi_1 = (q_1, a_1, c_1, q_2), (q_2, a_2, c_2, q_3) \dots (q_n, a_n, c_n, q_{n+1})$$

is an execution in $\mathcal{A}_1$ (Step 1 of Fig. 7.4).

Since $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $P^\lambda[\mathbf{WA}, \mathbf{FA}]$, there exists an execution π_2 in $\mathcal{A}_2$ of label $a_1 a_2 \dots a_n$ such that (Step 2 of Fig. 7.4):

$$\mathrm{cost}_{\mathcal{A}_2}(\pi_2) \leq \lambda \mathrm{cost}_{\mathcal{A}_1}(\pi_1). \tag{1}$$

Set

$$\pi_2 = (p_1, a_1, d_1, p_2), (p_2, a_2, d_2, p_3) \dots (p_n, a_n, d_n, p_{n+1}).$$

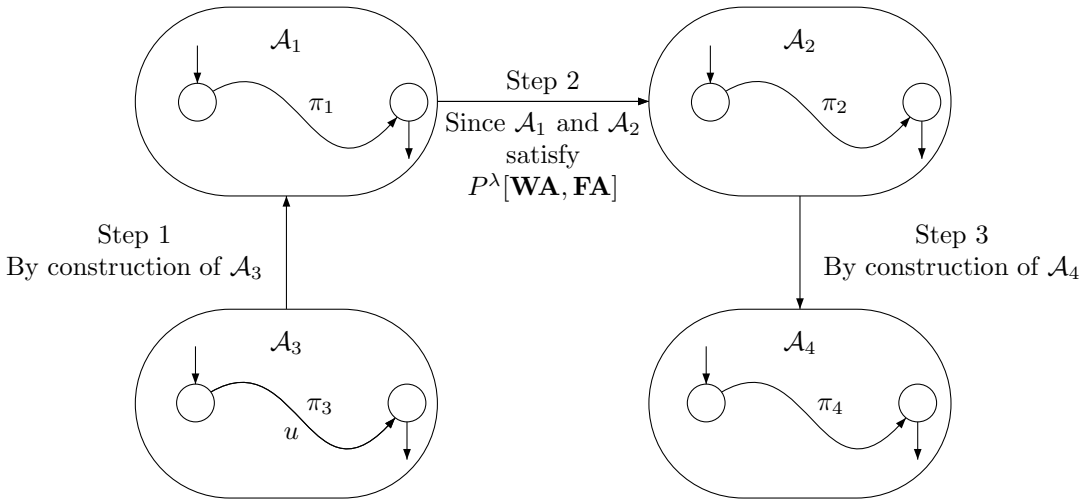


Figure 4: Proof of Proposition 10, part 1.

Now, by construction of $\mathcal{A}_4$,

$$\pi_4 = (p_1, [a_1, q_1, q_2], d_1, p_2), (p_2, [a_2, q_2, q_3], d_2, p_3) \dots (p_n, [a_n, q_n, q_{n+1}], d_n, p_{n+1})$$

is an execution of $\mathcal{A}_4$ (Step 3 of Fig. 7.4). Since $\text{cost}_{\mathcal{A}_2}(\pi_2) = \text{cost}_{\mathcal{A}_4}(\pi_4)$ and $\text{cost}_{\mathcal{A}_1}(\pi_1) = \text{cost}_{\mathcal{A}_3}(\pi_3)$ and by (1), the execution π in $\mathcal{B}$ corresponding to π_3 and π_4 has label u and a positive cost (the detailed proof of this last claim is quite similar to the one of Theorem 6 and is left to the reader).

($\Leftarrow$) Let assume now that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ and for every $u \in L(\mathcal{B})$, there exists an execution π in $\mathcal{B}$ such that $\text{cost}_{\mathcal{B}}(\pi) \geq 0$. This part of the proof is sketched in Fig. 7.4.

Let

$$\pi_1 = (q_1, a_1, c_1, q_2), (q_2, a_2, c_2, q_3) \dots (q_n, a_n, c_n, q_{n+1})$$

be an execution of $\mathcal{A}_1$. By construction of $\mathcal{A}_3$, one has in $\mathcal{A}_3$ the following execution (Step 1 in Fig. 7.4):

$$\pi_3 = (q_1, [a_1, q_1, q_2], c_1, q_2), (q_2, [a_2, q_2, q_3], c_2, q_3) \dots (q_n, [a_n, q_n, q_{n+1}], c_n, q_{n+1}).$$

Consequently, $u = [a_1, q_1, q_2][a_2, q_2, q_3] \dots [a_n, q_n, q_{n+1}]$ is in $L(\mathcal{A}_3)$. Since $\mathcal{A}_1 \sqsubseteq A_2$ and by construction of $\mathcal{A}_3$ and $\mathcal{A}_4$, $u \in L(\mathcal{A}_4)$. Consequently, by hypothesis, there exists an execution π in $\mathcal{B}$ of label u such that

$$\text{cost}_{\mathcal{B}}(\pi) \geq 0. \tag{2}$$

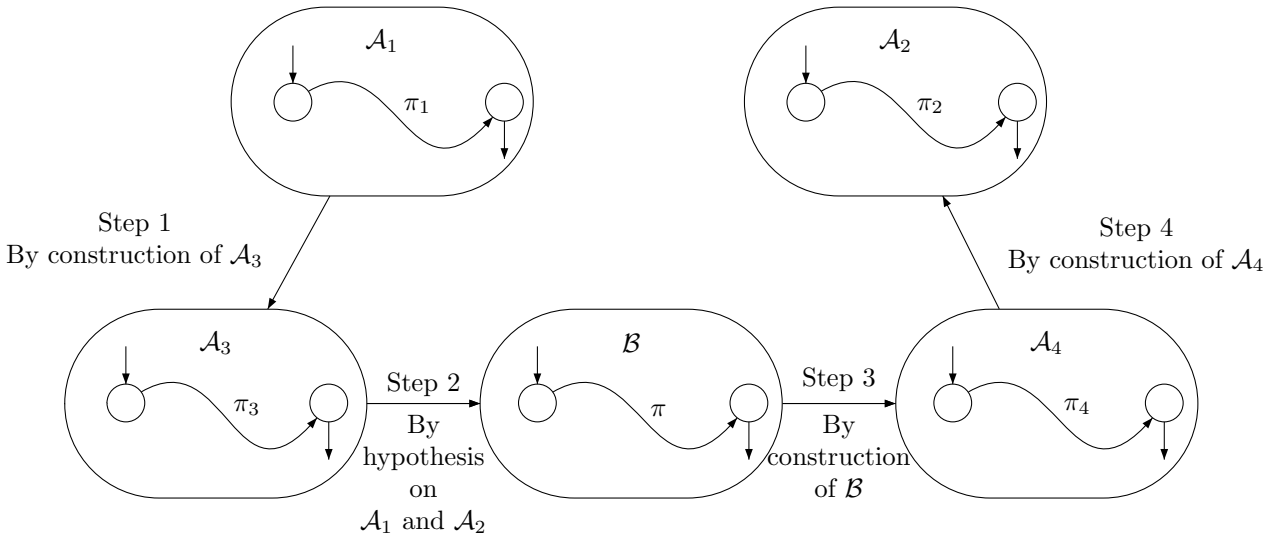


Figure 5: Proof of Proposition 10, part 2.

Let π_3' and π_4 be the corresponding executions of respectively $\mathcal{A}_3$ and $\mathcal{A}_4$ corresponding to π (Step 3 on Fig. 7.4). Using (2), one has (the proof is similar to the proof of Theorem 6):

$$\text{cost}_{\mathcal{A}_4}(\pi_4) \leq \lambda \text{cost}_{\mathcal{A}_3}(\pi_3').$$

Therefore, since $\mathcal{A}_3$ is unambiguous, $\pi_3 = \pi_3'$ and one has:

$$\text{cost}_{\mathcal{A}_4}(\pi_4) \leq \lambda \text{cost}_{\mathcal{A}_3}(\pi_3). \tag{3}$$

Set

$$\pi_4 = (p_1, [a_1, q_1, q_2], d_1, p_2), (p_2, [a_2, q_2, q_3], d_2, p_3) \ldots (p_n, [a_n, q_n, q_{n+1}], d_n, p_{n+1}).$$

By construction of $\mathcal{A}_4$, there exists an execution π_2 of $\mathcal{A}_2$ of the form (Step 4 in Fig 7.4):

$$\pi_2 = (p_1, a_1, d_1, p_2), (p_2, a_2, d_2, p_3) \ldots (p_n, a_n, d_n, p_{n+1}).$$

Since $\text{cost}_{\mathcal{A}_4}(\pi_4) = \text{cost}_{\mathcal{A}_2}(\pi_2)$ and by (3) one has:

$$\text{cost}_{\mathcal{A}_2}(\pi_2) \leq \lambda \text{cost}_{\mathcal{A}_3}(\pi_3).$$

Since $\pi_2 \sim \pi_1$, the proof of the claim is completed.

This finishes the proof of the proposition, the polynomial time decidability resulting from Theorem 5. □

7.5 Proof of Proposition 10

PROOF. By definition, one has $P^{\lambda}_{\text{strong}}[\mathbf{C_1} \cap \mathbf{UW}, \mathbf{C_2} \cap \mathbf{UW}] \subseteq P^{\lambda}[\mathbf{C_1} \cap \mathbf{UW}, \mathbf{C_2} \cap \mathbf{UW}]$.

Now, if $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $P^{\lambda}[\mathbf{C_1} \cap \mathbf{UW}, \mathbf{C_2} \cap \mathbf{UW}]$, then for every execution π_1 of $\mathcal{A}_1$ there exists an equivalent execution π_2 of $\mathcal{A}_2$ such that $\text{cost}_{\mathcal{A}_2}(\pi_2) \leq \lambda \text{cost}_{\mathcal{A}_1}(\pi_1)$. Let π_2' be an execution of $\mathcal{A}_2$ equivalent to π_1. By transitivity, π_2' and π_2 are equivalent. Moreover since $\mathcal{A}_2 \in \mathbf{UW}$, $\text{cost}_{\mathcal{A}_2}(\pi_2) = \text{cost}_{\mathcal{A}_2}(\pi_2')$. It follows that $P^{\lambda}[\mathbf{C_1} \cap \mathbf{UW}, \mathbf{C_2} \cap \mathbf{UW}] \subseteq P^{\lambda}_{\text{strong}}[\mathbf{C_1} \cap \mathbf{UW}, \mathbf{C_2} \cap \mathbf{UW}]$.

□

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399