Component Simulation-based Substitutivity Managing QoS and Composition Issues

Pierre-Cyrille Héam[a,b]

Olga Kouchnarenko[b]

Jérôme Voinot[b]

[a] *LSV CNRS/INRIA*
ENS Cachan
61 av. du Président Wilson
F-94235 Cachan Cedex
pcheam@lsv.ens-cachan.fr

[b] *INRIA/CASSIS and LIFC*
University of Franche-Comté
16 route de Gray
F-25030 Besançon Cedex
{okouchnarenko,jvoinot}@lifc.univ-fcomte.fr

Abstract

Several scientific bottlenecks have been identified in existing component-based approaches. Among them, we focus on the identification of a relevant abstraction for the component expression and verification of properties like substitutivity: When is it possible to formally accept or reject the substitution of a component in a composition? This paper suggests integer weighted automata to tackle this problem when considering a new factor – Quality of Service (QoS). Four notions of simulation-based substitutivity managing QoS aspects are proposed, and related complexity issues on integer weighted automata are investigated. Furthermore, the paper defines composition operators: sequential, strict-sequential and parallel compositions, bringing path costs into the analysis. New results on the compatibility of proposed substitutivity notions w.r.t. sequential and parallel composition operators are established.

Key words: Substitutivity, Component, Simulation, weighted automata, Quality of Service[1]

1. Introduction

This paper is dedicated to the verification of substitutivity of components modelled by integer weighted automata while considering a new factor – Quality of Service (QoS). In this context modelling and verifying both functional and non-functional properties is possible. For these verification problems, we provide new theoretical decidability results. Furthermore, the paper defines composition operators: sequential, strict-sequential and parallel compositions, bringing path costs into the analysis. We point out how compatible proposed substitutivity notions and sequential and parallel composition operators really are.

Component-based development provides significant advantages – portability, adaptability, re-usability, etc. – when developing, e.g., Java Card smart card applications or when composing Web services within Service Component Architecture (SCA). Several scientific bottlenecks have been identified in existing component-based approaches. Among them,

[1] This work is partially funded by the French ANR projects ARA COPS and ACI TACOS.

we focus on the identification of a relevant abstraction for the component expression and verification. When is it possible to accept or reject the substitution of a component in a composition? Moreover, with the increasing importance of QoS in the design of component-oriented applications, like Web services, it is of great interest for users and developers to be able to determine, possibly dynamically, that a Web service performs the same tasks as another possibly failing service, with comparable/higher quality.

1.1. Contributions

Most of prior and current works on component and service composition focus on either the functional aspect or the QoS aspect alone, it is very difficult to address both. This paper takes an approach of modelling components and services and QoS descriptions by integer weighted finite state automata, and studies the complexity of substitutivity of one such automaton by another.

More precisely, the present paper makes the following contributions: The *first contribution* is formal definitions of four – (partial) substitutivity and (partial) strong substitutivity – problems based on a simulation of automata taking path costs into account. For these substitutivity problems new decision/complexity results for different classes of integer weighted automata are presented.

The *second contribution* is formal definitions of composition operators: sequential, strict-sequential and parallel compositions, bringing path costs into the analysis. New results on the compatibility of proposed substitutivity notions with relation to sequential and parallel composition operators are established.

The *third contribution* concerns some practical issues on service and component substitutivity. We briefly situate component substitutivity w.r.t. various compositions in the context of a new type of urban, possibly driverless, vehicles. These examples illustrate why the topic is very important in practice, especially given the need to bring costs into consideration.

Notice that the first contribution was presented in [HKV08]. The second contribution is completely new. The third contribution follows and develops the examples in [HKV08].

1.2. Related Work

Weighted automata, trace-equivalence, simulations. Weighted automata – an extension of integer weighted automata – is a formalism widely used in computer science for applications in images compression [IvR99, KMT04], speech-to-text processing [MPR02, MPR05, BGW01] or discrete event systems [Gau95]. These large application areas make them intensively studied from the theoretical point of view [Kro94, Web94, HIJ02, KLMP04]. See [BR88] for more detail on weighted automata.

To compare processes or components, trace equivalences are in general not expressive enough and there are stronger equivalence relations permitting to consider deadlocks, livelocks, branching behaviours, causality, etc. Among them, the strong bisimulation equivalence by Milner [Mil80] and Park [Par81] is widely used in computer science because of its numerous advantages: It preserves branching behaviours and, consequently, most of the dynamic properties; there is a link between the strong bisimulation and modal logics [HM85]; this is a congruence for a number of composition operators, e.g. parallel composition, prefixing by an action, etc. The reader is referred to the survey [vG01] on simulation-preorder relations.

Bisimulation relations over weighted automata were investigated in [BK03]. In that paper authors consider that a max/plus automaton simulates another one if it can perform at the same moment the same action with the same weight. Our main purpose is to handle QoS aspects which are global notions over components. This is why in our paper, unlike [BK03], weights are related to successful paths of automata.

In the recent survey [tBBG07], the authors pointed out that, let us quote, “automata-based models are increasingly being used to formally describe, compose, and verify service compositions”. The role of automata-based analysis is also emphasised in [BABC$^+$09] for distributed components (Fractal, GCM and ProActive components). The main advantage of numerous works on component/service composition based on the use of automata or Labeled Transition Systems (LTSs) (see for instance [FUMK07, MR08]) is that their formal basis allows automatic tool support. However, extending automata (finite state automata, timed automata, I/O automata, team automata, etc.) with costs makes various verification problems undecidable in general [BBBR07]. In this framework, the present work defines four component/service substitutivity notions based on simulation relations of integer weighted automata, and provides constructive proofs for deciding substitutivity verification problems over those automata. Moreover, the article shows that the proposed notions are compatible with sequential and parallel composition operators which are essential for building new applications.

Modelling of QoS and of non functional properties of systems. The term non-functional requirement has been in use for more than 20 years, but there is still no consensus in the software engineering community on what non-functional requirements are, and on how we should elicit, document, and validate them [Gli07]. On the other hand, there is a unanimous consensus that non-functional requirements and properties are important and are critical for the success of a software development project. Hundreds of works exist based on the well-known quality models in [MRW77, BBK$^+$78] and those developed since 1977. In all these works, non-functional requirements and properties are a significant part of the software quality. A synthesis and a classification of existing requirements for the description of a component in order to use it in a component-based approach is in [CCH$^+$07].

Within the SCA initiative[2], a recent set of specifications describes a language-neutral model for building applications and systems using a Service-Oriented Architecture. SCA is claimed to be extendable and user friendly with:

- multiple implementation types including Java, C++, BPEL, PHP, Spring, etc.
- multiple bindings including Webservice, JMS, EJB, JSON RPC, etc.
- multiple hosting environments such as Tomcat, Jetty, Geronimo, OSGI, etc.

The policy framework provided with SCA supports specifications of constraints, capabilities and QoS expectations, from component design to concrete deployment.

Recently, minimum-cost delegation in service composition through the integration of existing services was studied in [GIRS08]. In this work, services are modelled as finite state machines augmented with linear counters, and service requirements are specified in a sequence form. Activity processing costs are integrated into the delegation computation, and promising polynomial time delegation techniques are developed. The main difference between this study and ours is that our goal is to verify if a service/component can be substituted by another one w.r.t. sequential and parallel compositions, while theirs is to compute a way to delegate desired actions to available services. Their automated composition synthesis task is closely related to planning.

Verifying the substitutivity of components and Web services. There are numerous works dealing with component substitutivity or interoperability [SCHS07, CVZ07, CHS06, BV06, Bra03]. Our work is close to that in [CVZ07], where the authors addressed component substitutability using equivalences between component-interaction automata, which are defined

[2] The first official specification of SCA providing hierarchical components is the 1.0 version, published in march 2007.

with respect to a given set of observable labels. In the present work, in addition to a set of labels, path costs are taken into account when comparing integer weighted automata.

In [BCH05, BCHS07], the authors defined three substitutivity notions over interface automata modelling Web services. First two notions deal with signatures and propositional constraints on the consistency between various method calls and return values. They are stateless and cannot be handled in our framework. The third substitutivity notion on protocol interfaces is based on a simulation relation over labelled transition systems like in the present paper. It is shown to be polynomial time decidable but it does not manage costs.

Different solutions have been proposed to allow taking QoS into account while specifying Web services and their compositions [LKD$^+$03, Tia05, d'A06, BRL07, HKV07]. In [HKV07] the substitutivity problem has been investigated for the trace equivalence over integer weighted automata.

In [LMW07, SW09] the authors studied the correct interaction between services modelled by open nets (uncoloured Petri nets with interfaces). The behaviour semantics of a set of open nets is given by annotated automata. These works on the correct interaction between services have been mainly inspired by the notion of soundness for workflow nets [vdA98]. Extending an annotated automaton with global constraints over its states proposed in [SW09] gives an operating guideline to characterise all correctly interacting partners of a service. Then simulation relations are used for deciding service composition and substitutability.

In [LVOS09] the authors compared and evaluated two different Petri net semantics for BPEL. Both implemented semantics abstract from data (messages and the content of variables). The properties that can be verified on the resulting models are (based on) soundness [vdA98], relaxed soundness [DvdA04], and also temporal logic properties.

The recent work in [CCSS08] is dedicated to the verification of a dynamic substitutability problem: can a component replace another component during an execution? The verification approach is based on recent model-checking techniques. Notice that action costs are not taken into account in [CCSS08]. In that setting, i.e. without considering costs, their substitutivity notion is stronger than the notion defined in the present paper.

The integration of (abstractions of) QoS properties into component models is supported several component-based approaches and tools, such as KLAPER [GMRS07], Palladio [BKR07] and RoboCop [FEHC02]. As these component models do not define any refinement notion, they are clearly distinguishable form our work. However, these models already provide very well validated abstractions on performance. Let us notice that the protocol for using a component is often context-dependent. It is due to automated component adaptation and architectural dependency analysis. Parametric contracts [Reu03, RHH05] for software components allow addressing this aspect and were successfully used for automated protocol adaptation and quality of service prediction.

Finally, in [MSK05, FM07] authors show how to use automata and concurrent logic to model component-based systems. In these works, finite automata are derived from UML descriptions and synchronisations are performed using interface constraints.

1.3. Layout of the paper

The remainder of the paper is organised as follows. A motivating example is given in Sect. 2. Section 3 recalls integer weighted automata and defines four simulation-oriented substitutivity notions based on them. The verification issues on components substitutivity are presented in Sect. 4 and 5. Section 6 puts the substitutivity problems in the composition context. Section 7 exposes how the theoretical results would be exploited in practice. Finally, Section 8 concludes and gives some prospectives.

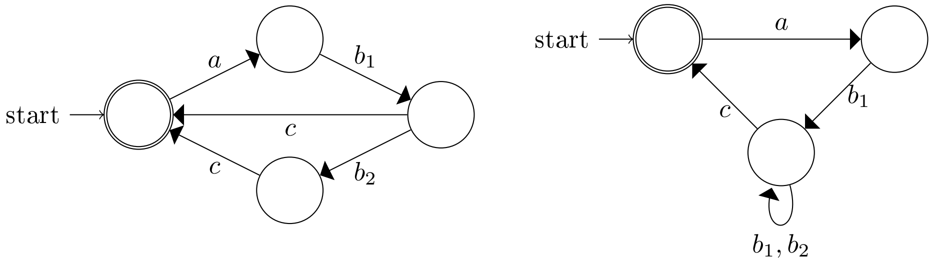

Figure 1: Components C_1 and C_2

2. Motivating Example: Localisation Component

This section quickly presents the substitutivity problem on a characteristic example. It is inspired from a real case study in the land transportation domain.

Context. The TACOS project[3] concerns the development of a new type of urban vehicles with new functionalities and services. The project follows the Cybercar concept, a public transport system with fully or partially automated driving capabilities, aimed at replacing the private car. One of the major cornerstones is the development, the validation and the certification of vehicles, like Cristal or Cycab.

A positioning system is a critical part of a land transportation system. Many positioning systems have been proposed over the past few years. Among them, let's quote GPS, GALILEO or GLONASS positioning systems which belong to the Global Navigation Satellite Systems (GNSS, for short). However, currently only some mobile terminals (laptops, PDAs, cell phones, etc.) are embedded with GNSS receivers. In addition, positioning systems are often dedicated to a particular environment; e.g., the GNSS systems generally do not work indoors. To solve these problems, numerous alternatives relying on different technologies, have arisen (see [SE06, EFPC04, HNS03, RMG05, OG00] for more details on issues related to positioning systems).

The present section and Section 7 briefly describe how such heterogeneous positioning systems, encapsulated as components, called localisation components, are used together to provide positioning data satisfying some non functional requirements. Note that positioning data can be given in different formats. The most used format is the geographic one, like that usually obtained from a GPS positioning system. But other systems give semantic location data, like 'You are near the station Place Stanislas'.

In this framework, let us consider the two following positioning components where Wireless networks are exploited to extend the use of the GNSS. Their abstract representations are given by finite automata in Fig. 1. The question, the positioning component user is interested in, is: 'When is it possible to accept or reject the substitution of a component by another component?'

- Component C1 works as follows. Action a encodes that C1 receives a positioning request; at this stage, C1 performs either only action b_1 or both b_1 and b_2 depending on the (abstracted) value passed through the a request. The action b_1 corresponds to a geographic location computing where as b_2 encodes a semantic location computing.

[3] The French National Research Agency TACOS project, ANR-06-SETI-017 (`http://tacos.loria.fr`).

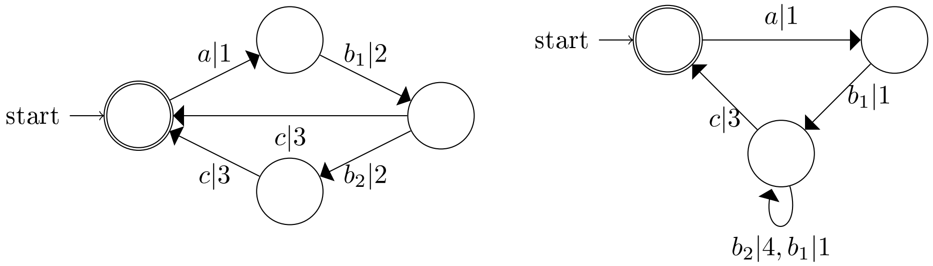

Figure 2: Components C_3 and C_4 with costs

The abstracted value may depend on an environment where the available power or the power consumption must be taken into account/reduced. For example, once the geographic location obtained, a vehicle whose available power is not enough to reach the next station because of a critical environment, must compute semantic location data to offer to its passengers. Then C1 performs the action c to acknowledge that its positioning task is successfully executed.

- Component C2 works similarly but after having done first b_1, it can perform actions b_1 or b_2 as many times as it is required. For example, depending on the speed of the vehicle, the localisation system must give the position more or less frequently.

Obviously, the C1 component can be functionally substituted by C2. Furthermore, when considering, e.g., energy costs over components represented by finite automata C3 and C4 in Fig. 2, the cost of each action is put on each transition.

For both C3 and C4, receiving a positioning request a costs 1 energy unit and performing c costs 3 energy units. However, for C3 each action b_1 and b_2 costs 2 energy units. For C4, performing the b_1 action costs only 1 energy unit but all b_2 actions cost 4 energy units. The intuition behind this modelling is as follows. C3 has a low-cache memory allowing it to locally compute actions b_1 and b_2. C4 has a high performance low-cache memory that allows it to locally compute action b_1 with a cost of 1 energy unit. C4 also has a local hard drive that makes b_2 computations possible. However, reading and writing on hard drives has a high energy cost of 4 energy units. In this situation, we do not want to say that C4 can substitute C3 since performing ab_1b_2c on C3 has the cost of 8 energy units whereas the same sequence of actions costs 9 energy units on C4.

3. Simulation-based Component Substitutivity

3.1. Theoretical Background

In this paper, Σ denotes a finite set of actions. We first introduce the notion of integer weighted automata. To simplify the presentation the results are given for integer weighted automata but can be easily extended to any weights in a semi-ring.

Definition 1. *A* finite integer weighted automaton $\mathcal{A}$ *over* Σ *is a quintuplet*

$$\mathcal{A} = (Q, \Sigma, E, I, F)$$

where Q *is the finite set of states,* $E \subseteq Q \times \Sigma \times \mathbb{Z} \times Q$ *is the set of transitions,* $I \subseteq Q$ *is the set of initial states, and* $F \subseteq Q$ *is the set of final states. Finite integer weighted automata are often simply called automata in the sequel.*

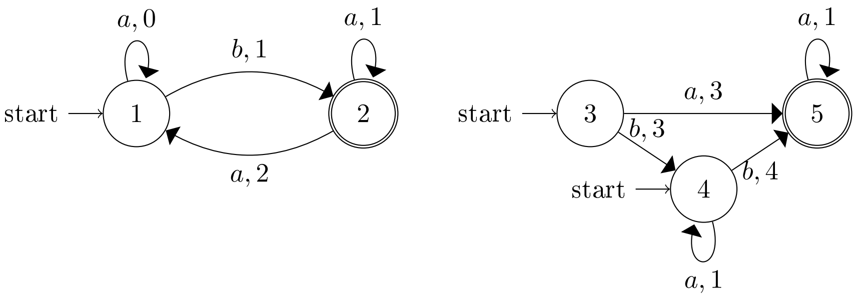

Figure 3: Automata $\mathcal{A}_{exe1}$ and $\mathcal{A}_{exe2}$

Figure 3 gives two examples of finite integer weighted automata. Initial states are represented with an input arrows and final states with a double circle.

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), also written $p \xrightarrow{a,c}_{\mathcal{A}} q$. Now we formally define an execution of a integer weighted automaton and related notions.

A *partial execution* or a *path* of a finite integer 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* or *a successful path*. The *trace/label* $tr(\pi)$ 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: $\text{cost}_{\mathcal{A}}(\pi) = \sum_{i=0}^{n} c_i$. For instance, $(1, a, 0, 1), (1, a, 0, 1), (1, b, 1, 2), (2, a, 2, 1)$ is a successful path of $\mathcal{A}_{exe1}$, whose trace is $aaba$ and whose weight is $0 + 0 + 1 + 2 = 3$.

A state p of a integer weighted automaton is *accessible/reachable* (resp. *co-accessible/co-reachable*) if there exists a path from an initial state to p (resp. from p to a final state). For instance, in the automaton depicted in Fig. 10, the state $2, 3$ is not accessible. Basically, given $\mathcal{A}$, $L(\mathcal{A})$ denotes its set of execution traces.

An automaton is *trim* if its states are all both accessible and co-accessible. It is well known that for every automaton $\mathcal{A}$, there exists a trim automaton with the same set of successful executions. Moreover, computing this trim automaton can be done in polynomial time. For instance, the trim automaton in Fig. 11 is obtained from the automaton in Fig. 10.

An automaton $\mathcal{A}$ is *finitely ambiguous* if there exists a positive integer k such that for every word w there exists at most k successful paths in $\mathcal{A}$ labelled by w. For example, the automaton $\mathcal{A}_{exe2}$ is finitely ambiguous whereas the automaton $\mathcal{A}_{exe1}$ is not: the word $ba^n b$ is accepted by n different successful paths, depending when the transition from 2 to 1 is fired.

Definition 2. *Let $\mathcal{A}_1 = (Q_1, A, E_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q_2, A, E_2, I_2, F_2)$ be two automata. A binary relation $\preceq_{\mathcal{A}_1,\mathcal{A}_2} \subseteq Q_1 \times Q_2$ is a simulation if $(p_1, p_2) \in \preceq_{\mathcal{A}_1,\mathcal{A}_2}$ implies, for all a in A and all c_1 in $\mathbb{Q}$,*

i) for every $q_1 \in Q_1$, if $(p_1, a, c_1, q_1) \in E_1$ then there exist $q_2 \in Q_2$ and $c_2 \in \mathbb{Q}$ such that $(p_2, a, c_2, q_2) \in E_2$ and $(q_1, q_2) \in \preceq_{\mathcal{A}_1,\mathcal{A}_2}$, and

ii) if p_1 is final, then p_2 is final too.

If there is no ambiguity on $\mathcal{A}_1$ and $\mathcal{A}_2$, we just say that p_2 $\preceq$- simulates p_1, written $p_1 \preceq p_2$, when there is a simulation containing (p_1, p_2). It is easy to see that the largest simulation on $Q_1 \times Q_2$ exists. To simplify the notations, the largest simulation on $Q_1 \times Q_2$ is also denoted by $\preceq_{\mathcal{A}_1,\mathcal{A}_2}$.

The above relation is extended to paths of $\mathcal{A}_1$ and $\mathcal{A}_2$ in the following way: an execution π_2 of $\mathcal{A}_2$ $\preceq$- simulates an execution π_1 of $\mathcal{A}_1$ if and only if they have the same label (and consequently the same length) and for every i, $\pi_1[i] \preceq \pi_2[i]$. Finally, we write $\mathcal{A}_1 \preceq \mathcal{A}_2$ if for every co-accessible initial state i_1 of $\mathcal{A}_1$ there exists an initial state i_2 of $\mathcal{A}_2$ such that $i_1 \preceq i_2$. For our example in Sect. 2, it is easy to see that C3 $\preceq$ C4.

3.2. Modelling Substitutivity

A problem occurring while managing components/services is to determine that a component/service performs the same tasks as another possibly failing service, with comparable or higher quality. More formally, for two Web services modelled by their integer weighted automata $\mathcal{A}_1$ and $\mathcal{A}_2$, the problem is to decide whether $\mathcal{A}_2$ can have the same behaviour as $\mathcal{A}_1$ with a similar or higher quality. To address this problem, four notions of simulation-based substitutivity managing QoS aspects are proposed in this section.

The notion of 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 the *cost* of this way is comparable or better that the cost in S_1. Intuitively, the substitutivity is an existential notion: for each sequence of actions that can be done by S_1, there exists in S_2 an equivalent sequence of actions with a smaller cost. 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 is, its quality is similar or higher. Intuitively, the strong substitutivity notion requires a stronger universal quantification ensuring that not only S_2 can do better that S_1, but that it will always do better.

Substitutivity Problem
Input: Two automata $\mathcal{A}_1$ and $\mathcal{A}_2$.
Output: True if for every successful path π_1 of $\mathcal{A}_1$ there exists a successful path π_2 of $\mathcal{A}_2$ such that $\pi_1 \preceq \pi_2$ and $\mathrm{cost}_{\mathcal{A}_2}(\pi_2) \leq \mathrm{cost}_{\mathcal{A}_1}(\pi_1)$, false otherwise.

We write $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ when $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the substitutivity problem.

Strong Substitutivity Problem
Input: Two automata $\mathcal{A}_1$ and $\mathcal{A}_2$.
Output: True if for every successful path π_1 of $\mathcal{A}_1$ there exists a successful path π_2 of $\mathcal{A}_2$ such that $\pi_1 \preceq \pi_2$ and for every π_2' of $\mathcal{A}_2$ such that $\pi_1 \preceq \pi_2'$, $\mathrm{cost}_{\mathcal{A}_2}(\pi_2') \leq \mathrm{cost}_{\mathcal{A}_1}(\pi_1)$, false otherwise.

We write $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}} \mathcal{A}_2$ when $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the strong substitutivity problem.

It is sometime fruitful to compare successful executions costs only on subtraces. This leads to the following *partial* substitutivity problems that are similar to the ones above. For these problems, we want to compare parts of executions, not paths that cannot be related to a successful path. Consequently, automata are required to be trim, and comparisons are done for all paths, not only for successful paths.

Partial Substitutivity Problem
Input: Two trim automata $\mathcal{A}_1$ and $\mathcal{A}_2$.
Output: True if for every path π_1 of $\mathcal{A}_1$ there exists a path π_2 of $\mathcal{A}_2$ such that $\pi_1 \preceq \pi_2$ and $\mathrm{cost}_{\mathcal{A}_2}(\pi_2) \leq \mathrm{cost}_{\mathcal{A}_1}(\pi_1)$, false otherwise.

We note $\mathcal{A}_1 \sqsubseteq_{\mathrm{p}} \mathcal{A}_2$ when $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the partial substitutivity problem.

Partial Strong Substitutivity Problem
Input: Two trim automata $\mathcal{A}_1$ and $\mathcal{A}_2$.
Output: True if for every path π_1 of $\mathcal{A}_1$ there exists a path π_2 of $\mathcal{A}_2$ such that $\pi_1 \preceq \pi_2$ and for every π_2' of $\mathcal{A}_2$ such that $\pi_1 \preceq \pi_2'$, $\text{cost}_{\mathcal{A}_2}(\pi_2') \leq \text{cost}_{\mathcal{A}_1}(\pi_1)$, false otherwise.

We write $\mathcal{A}_1 \sqsubseteq_{\text{p}}^{\text{st}} \mathcal{A}_2$ when $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the partial strong substitutivity problem.

Notice that in the above definitions we choose that $\text{cost}(\pi_2) \leq \text{cost}(\pi_1)$ 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 $\text{cost}(\pi_2) \leq \text{cost}(\pi_1)$ into $\text{cost}(\pi_2) \geq \text{cost}(\pi_1)$. 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.

We end this section by recalling some results on decision procedures for finite integer weighted automata.

Theorem 3. *Given two integer weighted automata $\mathcal{A}_1$ and $\mathcal{A}_2$, it is*

- *undecidable to test whether for every $u \in L(\mathcal{A}_1)$, $\text{cost}_{\mathcal{A}_1}(u) \leq \text{cost}_{\mathcal{A}_2}(u)$ [Kro94]; the 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 $u \in L(\mathcal{A}_1)$, $\text{cost}_{\mathcal{A}_1}(u) \leq \text{cost}_{\mathcal{A}_2}(u)$ 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 finitely ambiguous [WS91].*
- *PSPACE-complete to decide whether $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ [AHU74, BJ06].*

4. Strong Substitutivity Problems

This section provides decidability results for the strong substitutivity and the partial strong substitutivity problems.

Lemma 4. *One has $\mathcal{A}_1 \preceq \mathcal{A}_2$ if and only if for every successful path π_1 of $\mathcal{A}_1$ there exists a successful path π_2 of $\mathcal{A}_2$ such that $\pi_1 \preceq \pi_2$.*

Proof. Assume first that for every successful path π_1 of $\mathcal{A}_1$ there exists a successful path π_2 of $\mathcal{A}_2$ such that $\pi_1 \preceq \pi_2$. Let i_1 be a co-accessible state of $\mathcal{A}_1$. By definition of co-accessibility, there exists a successful path π_1 in $\mathcal{A}_1$ starting from i_1. By hypothesis, there exists a successful path π_2 of $\mathcal{A}_2$ such that $\pi_1 \preceq \pi_2$. Therefore, $\pi_1[1] \preceq \pi_2[1]$. But $\pi_1[1] = i_1$ and since π_2 is a successful path, $\pi_2[1]$ is an initial state of $\mathcal{A}_2$. Consequently, $\mathcal{A}_1 \preceq \mathcal{A}_2$.

Assume now that $\mathcal{A}_1 \preceq \mathcal{A}_2$. Let π_1 be a successful path of $\mathcal{A}_1$. Since $\pi_1[1]$ is an initial state and since $\mathcal{A}_1 \preceq \mathcal{A}_2$, there exists an initial state q_1 in $\mathcal{A}_2$ such that $\pi_1[1] \preceq q_1$. Therefore, if we denote by $(\pi_1[1], a_1, c_1, \pi_1[2])$ the first transition of π_1, there exists a state q_2 in $\mathcal{A}_2$ and $d_1 \in \mathbb{Z}$, such that (q_1, a_1, d_1, q_2) is a transition of $\mathcal{A}_2$ and $\pi_1[2] \preceq q_2$. Iterating this construction, one can, by a direct induction, build a successful path π_2 of $\mathcal{A}_2$ such that $\pi_1 \preceq \pi_2$, which concludes the proof. □

Theorem 5. *The strong substitutivity problem is P-complete.*

Proof. Let $\mathcal{A}_1 = (Q_1, A, E_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q_2, A, E_2, I_2, F_2)$ be two automata. We denote by $\mathcal{B}$ the automaton (Q, A, E, I, F) where

- $Q = \{(q_1, q_2) \in Q_1 \times Q_2 \mid q_1 \preceq q_2\}$,
- $E = \{((p_1, p_2), a, c, (q_1, q_2)) \mid (p_1, a, c_1, q_1) \in E_1,\ (p_2, a, c_2, q_2) \in E_2,\ c = c_1 - c_2,\ a \in A\}$,
- $I = (I_1 \times I_2) \cap Q$ and $F = (F_1 \times F_2) \cap Q$.

We claim that $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}} \mathcal{A}_2$ if and only if $\mathcal{A}_1 \preceq \mathcal{A}_2$ and for every successful path π of $\mathcal{B}$, $\mathrm{cost}_{\mathcal{B}}(\pi) \geq 0$.

($\Rightarrow$) Assume that $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}} \mathcal{A}_2$. By Lemma 4, for every successful path of $\mathcal{A}_1$ there exists an $\preceq$-related path in $\mathcal{A}_2$. Thus $\mathcal{A}_1 \preceq \mathcal{A}_2$. Consider now a successful path π in $\mathcal{B}$,

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

By definition of $\mathcal{B}$, there exist $p_0, p_1, \ldots, p_n$ states of $\mathcal{A}_1$, $q_0, q_1, \ldots, q_n$ states of $\mathcal{A}_2$, integers $c_1, c_2, \ldots, c_n, d_1, d_2, \ldots, d_n$ such that

- $\pi_1 = (p_0, a_1, c_1, p_1), (p_1, a_2, c_2, p_2), \ldots, (p_{n-1}, a_n, c_n, p_n)$ is a successful path in $\mathcal{A}_1$,
- $\pi_2 = (q_0, a_1, d_1, q_1), (q_1, a_2, d_2, q_2), \ldots, (q_{n-1}, a_n, d_n, q_n)$ is a successful path in $\mathcal{A}_2$,
- for every $1 \leq i \leq n$, $\alpha_i = c_i - d_i$,
- for every $0 \leq i \leq n$, $\overline{p_i} = (p_i, q_i)$ and $p_i \preceq q_i$.

Thus, one has $\pi_1 \preceq \pi_2$. Therefore, since $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the strong substitutivity problem, the following inequality holds:

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

$$\text{Consequently,} \quad \mathrm{cost}_{\mathcal{B}}(\pi) = \sum_{i=1}^{n} \alpha_i \geq 0.$$

($\Leftarrow$) Assume now that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $\mathcal{A}_1 \preceq \mathcal{A}_2$ and for every successful path π of $\mathcal{B}$, $\mathrm{cost}_{\mathcal{B}}(\pi) \geq 0$.

Since $\mathcal{A}_1 \preceq \mathcal{A}_2$, by Lemma 4, for every successful path in $\mathcal{A}_1$ there exists a $\preceq$-related successful path in $\mathcal{A}_2$.

Finally, consider two successful paths

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

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 $\pi_1 \preceq \pi_2$.

By definition there exists an successful path π in $\mathcal{B}$,

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

such that

- for every $1 \leq i \leq n$, $\alpha_i = c_i - d_i$,
- for every $0 \leq i \leq n$, $\overline{p_i} = (p_i, q_i)$ and $p_i \preceq q_i$.

Moreover, by hypotheses, one has $\mathrm{cost}_{\mathcal{B}}(\pi) \geq 0$:

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

$$\text{Consequently,}\quad \sum_{i=1}^{n} d_i \leq \sum_{i=1}^{n} c_i.$$

It follows that $\mathrm{cost}_{\mathcal{A}_2}(\pi_2) \leq \mathrm{cost}_{\mathcal{A}_1}(\pi_1)$, proving the claim.

Deciding whether $\mathcal{A}_1 \preceq \mathcal{A}_2$ is known to be P-complete [SJ01, SJ05]. Now deciding whether for every successful path π of $\mathcal{B}$, $\mathrm{cost}_{\mathcal{B}}(\pi) \geq 0$ is a basic polynomial problem on weighted graphs which can be solved for instance by Bellman-Ford's algorithm.

The P-completeness is trivially obtained using the claim on automata with nil weights and the P-completeness of testing whether $\mathcal{A}_1 \preceq \mathcal{A}_2$.

□

Theorem 6. *The partial strong substitutivity problem is P-complete.*

Proof. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two trim automata. Let $\mathcal{B}$ be the automaton constructed as in the proof of Theorem 5. We claim that $\mathcal{A}_1 \sqsubseteq_{\mathrm{p}}^{\mathrm{st}} \mathcal{A}_2$ if and only if $\mathcal{A}_1 \preceq \mathcal{A}_2$ and if every transition of $\mathcal{B}$ has a positive weight.

The proof is quite similar to the one of Theorem 5: if $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the partial strong substitutivity problem, then using the property on paths of length 1, each transition of $\mathcal{B}$ has to be positively weighted. Conversely, if every transition of $\mathcal{B}$ has a positive weight, it is clear by a direct induction on paths lengths, that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the partial strong substitutivity problem.

The P-completeness is also trivially obtained using the claim on automata with nil weights and the P-completeness of testing whether $\mathcal{A}_1 \preceq \mathcal{A}_2$.

□

5. Substitutivity Problems

This section provides decidability results for the substitutivity and the partial substitutivity problems.

Theorem 7. *The substitutivity problem is polynomial time decidable if $\mathcal{A}_2$ is finitely ambiguous.*

Proof. Let $\mathcal{A}_2 = (Q_2, \Sigma, E_2, I_2, F_2)$ a finitely ambiguous integer weighted automaton and $\mathcal{A}_1 = (Q_1, \Sigma, E_1, I_1, F_1)$ be a integer weighted automaton. 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:

- $E_3 = \{(p, [a, p, q], c, q) \mid (p, a, c, q) \in E_1\}$,
- $E_4 = \{(p, [a, r, s], c, q) \mid (p, a, c, q) \in E_2,\ \exists x \in \mathbb{Z},\ (r, a, x, s) \in E_1$,$r, s \in Q_1$and $r \preceq p$ and $s \preceq q\}$.

Notice that $\mathcal{A}_3$ is unambiguous and that $\mathcal{A}_4$ is 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 because of restriction on E in Sect. 3. 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 ℓ executions in $\mathcal{A}_4$ accepting u. Thus $\mathcal{A}_4$ is finitely ambiguous.

Let $\mathcal{B} = \mathcal{A}_3 \times (-\mathcal{A}_4)$, where $-\mathcal{A}_4$ is obtained from $\mathcal{A}_4$ by multiplying the weight of each transition by -1.

We claim that $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$ if and only if $\mathcal{A}_1 \preceq \mathcal{A}_2$ and for every $u \in L(\mathcal{B})$, there exists an execution π in $\mathcal{B}$ such that $\text{cost}_{\mathcal{B}}(\pi) \geq 0$.

($\Rightarrow$) Assume first that $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$. Then $\mathcal{A}_1 \preceq \mathcal{A}_2$. Now let $u \in L(\mathcal{B})$.

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 $\mathcal{A}_1$.

Since $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$, there exists an execution π_2 in $\mathcal{A}_2$ of label $a_1 a_2 \dots a_n$ such that

$$\text{cost}_{\mathcal{A}_2}(\pi_2) \leq \text{cost}_{\mathcal{A}_1}(\pi_1) \quad \text{and} \quad \pi_1 \preceq \pi_2. \tag{1}$$

Set

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

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 $\mathcal{A}_4$. 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.

($\Leftarrow$) Let assume now that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $\mathcal{A}_1 \preceq \mathcal{A}_2$ and for every $u \in L(\mathcal{B})$, there exists an execution π in $\mathcal{B}$ such that $\text{cost}_{\mathcal{B}}(\pi) \geq 0$.

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

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

Consequently, since $\mathcal{A}_1 \preceq \mathcal{A}_2$, there exists a successful path π_4 in $\mathcal{A}_4$ such that $\pi_3 \preceq \pi_4$. It follows that $u = [a_1, q_1, q_2][a_2, q_2, q_3] \dots [a_n, q_n, q_{n+1}]$ is in $L(\mathcal{B})$. By hypothesis, there exists an execution π in $\mathcal{B}$ of label u such that

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

Let π'_3 and π'_4 be the corresponding executions of respectively $\mathcal{A}_3$ and $\mathcal{A}_4$ corresponding to π. Using (2), one has:

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

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

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

Set

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

By construction of $\mathcal{A}_4$, there exists an execution π_2 of $\mathcal{A}_2$ of the form:

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

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

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

Since by construction $\pi_2 \preceq \pi_1$, the proof of the claim is completed.

This finishes the proof of the theorem, the polynomial time decidability resulting from Theorem 3. □

Theorem 8. *The partial substitutivity problem is decidable in polynomial time.*

Proof. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two trim automata. We claim that automata $\mathcal{A}_1 \sqsubseteq_{\mathrm{p}} \mathcal{A}_2$ if for every transition (p_1, a, c_1, q_1) of $\mathcal{A}_1$ there exists a transition (p_2, a, c_2, q_2) of $\mathcal{A}_2$ such that $c_2 \leq c_1$, $p_1 \preceq p_2$ and $q_1 \preceq q_2$. Indeed, if $\mathcal{A}_1 \sqsubseteq_{\mathrm{p}} \mathcal{A}_2$ then, using the property on paths of length 1, one has the desired result. Conversely, if for every transition (p_1, a, c_1, q_1) of $\mathcal{A}_1$ there exists a transition (p_2, a, c_2, q_2) of $\mathcal{A}_2$ such that $c_2 \leq c_1$, $p_1 \preceq p_2$ and $q_1 \preceq q_2$, a direct induction on paths lengths shows that $\mathcal{A}_1 \sqsubseteq_{\mathrm{p}} \mathcal{A}_2$.

Computing relation $\preceq$ can be done in polynomial time. Next, it suffices to check the above property by a simple walk of the transitions list. □

6. Substitutivity and Composition

In this section we put the substitutivity problems introduced in this paper in the composition context. We define three natural composition operators: sequential, strict-sequential and parallel compositions. To motivate composition operators, let us mention ATP rules formalising BPEL in [MR08], in discrete-time. Another example comes from applications where CSP controllers are used for B machines modelling the components. Indeed, in CSP||B approach, the CSP sequential and parallel composition operators are allowed [ET07] to control B machines. A lot of process algebraic approaches allow such composition operators. In addition to these well-known operators, we consider the strict sequential composition operator allowing to observe when the control goes from the first component to the second one. This operator is useful, e.g., for the architectural description of the composite Fractal components [BABC$^+$09]. Notice also that our parallel composition operator is the same as the operator studied in [CCSS08], but in addition our operator handles action costs.

We demonstrate that considering path costs when verifying simulation relations in a composition manner, does have a cost: some (but not all) substitutivity notions introduced in this paper are not compatible with several composition operators. New positive composition results are also provided.

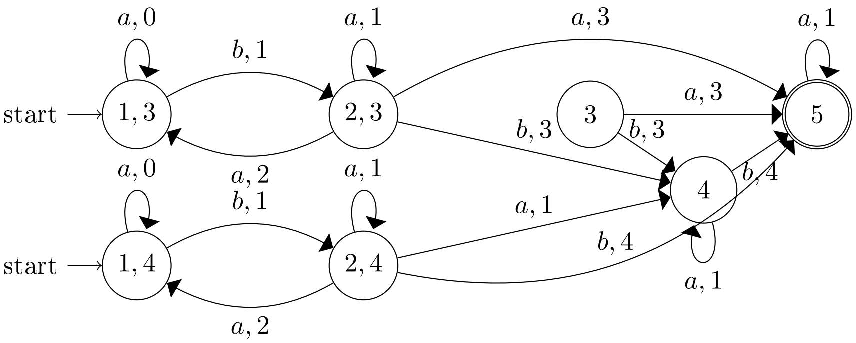

Figure 4: Automaton $\mathcal{A}_{exe1}.\mathcal{A}_{exe2}$

6.1. Substitutivity and Sequential Composition

Definition 9. *Let $\mathcal{A}_1 = (Q_1, A_1, E_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q_2, A_2, E_2, I_2, F_2)$ be two integer weighted automata. The sequential composition of $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted $\mathcal{A}_1.\mathcal{A}_2$, is the automaton $\mathcal{A}_{12} = (Q_{12}, A_{12}, E_{12}, I_{12}, F_{12})$ where*

- $Q_{12} = \{(p_1, p_2) \mid p_1 \in Q_1, p_2 \in I_2\} \cup Q_2$,
- $A_{12} = A_1 \cup A_2$,
- $I_{12} = \{(p_1, p_2) \mid p_1 \in I_1, p_2 \in I_2\}$,
- $F_{12} = F_2$,

and where the transition relation E_{12} obeys the following rules:

$$\textit{[SEQ1]}\ \frac{p_1 \xrightarrow{a_1,c_1}_{\mathcal{A}_1} q_1}{(p_1,p_2) \xrightarrow{a_1,c_1}_{\mathcal{A}_1.\mathcal{A}_2} (q_1,p_2)} \qquad \textit{[SEQ2]}\ \frac{p_2 \xrightarrow{a_2,c_2}_{\mathcal{A}_2} q_2}{(p_1,p_2) \xrightarrow{a_2,c_2}_{\mathcal{A}_1.\mathcal{A}_2} q_2}\ p_1 \in F_1$$

$$\textit{[SEQ3]}\ \frac{p_2 \xrightarrow{a_2,c_2}_{\mathcal{A}_2} q_2}{p_2 \xrightarrow{a_2,c_2}_{\mathcal{A}_1.\mathcal{A}_2} q_2}$$

States of the form (p_1, p_2), with $p_1 \in Q_1$ and $p_2 \in Q_2$ are called composed states.

This definition means that all moves of sequential composition are moves of either $\mathcal{A}_1$, or of $\mathcal{A}_2$ if $\mathcal{A}_1$ is in a final state, or of $\mathcal{A}_2$ if the state is a non composed one.

Given the two automata $\mathcal{A}_{exe1}$ and $\mathcal{A}_{exe2}$ depicted in Fig. 3, their sequential composition $\mathcal{A}_{exe1}.\mathcal{A}_{exe2}$ is given in Fig. 4.

Lemma 10. *Let $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ be four automata such that there exist two simulation relations $\preceq_1$ and $\preceq_2$, and $\mathcal{A}_1 \preceq_1 \mathcal{A}_3$ and $\mathcal{A}_2 \preceq_2 \mathcal{A}_4$. We define the relation $\mathcal{R}$ between the states of $\mathcal{A}_1.\mathcal{A}_2$ and the states of $\mathcal{A}_3.\mathcal{A}_4$ by*

- *$(p_1, p_2)\mathcal{R}(p_3, p_4)$ if and only if $(p_1 \preceq_1 p_3)$ and $(p_2 \preceq_2 p_4)$ and,*
- *$p_2\mathcal{R}p_4$ if and only if $p_2 \preceq_2 p_4$ and,*
- *there is neither state of the form (p_1, p_2) related by $\mathcal{R}$ to a state of $\mathcal{A}_4$, nor state of $\mathcal{A}_2$ related by $\mathcal{R}$ to a state of the form (q_3, q_4).*

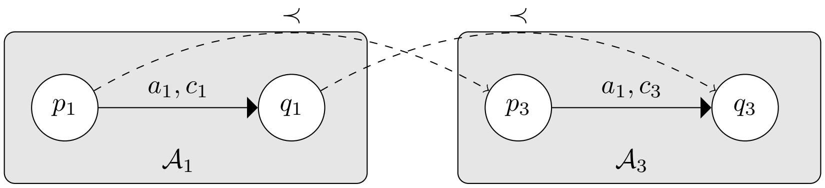

Figure 5: Proof of Lemma 11.

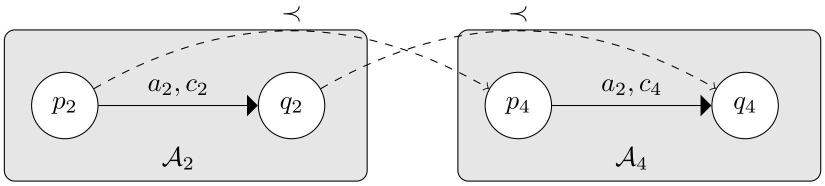

Figure 6: Proof of Lemma 11.

The relation $\mathcal{R}$ is a simulation relation.

PROOF. Since final states of $\mathcal{A}_1.\mathcal{A}_2$ are final states of $\mathcal{A}_2$ and final states of $\mathcal{A}_3.\mathcal{A}_4$ are final states of $\mathcal{A}_4$, and by definition of $\mathcal{R}$, the relation $\mathcal{R}$ satisfies the condition ii) of Definition 2.

There are three kinds of transitions obeying either the rule $[SEQ1]$, or $[SEQ2]$, or $[SEQ3]$.

- For non composed states, since every transition from a non composed state of $\mathcal{A}_1.\mathcal{A}_2$ targets, by the rule $[SEQ3]$, a non composed state of $\mathcal{A}_1.\mathcal{A}_2$, the condition i) of Definition 2 is satisfied for states of $\mathcal{A}_2$ and $\mathcal{A}_4$.
- For composed states, assume that $p_1 \preceq_1 p_3$ and $p_2 \preceq_2 p_4$. Two kinds of transitions can be fired from (p_1, p_2).
  - If there is a transition (p_1, a_1, c_1, q_1) in $\mathcal{A}_1$, then by $[SEQ1]$ there is a transition in $\mathcal{A}_1.\mathcal{A}_2$ of the form $((p_1, p_2), a_1, c_1, (q_1, p_2))$ (see Fig. 5). Since $p_1 \preceq_1 p_3$, by Definition 2 there is a transition (p_3, a_1, c_3, q_3) in $\mathcal{A}_3$ such that $q_1 \preceq_1 q_3$. Thus $(q_1, p_2)\mathcal{R}(q_3, p_4)$.
  - If there is a transition (p_2, a_2, c_2, q_2) in $\mathcal{A}_2$, then by $[SEQ2]$ there is a transition $((p_1, p_2), a_2, c_2, q_2)$ in $\mathcal{A}_1.\mathcal{A}_2$ (see Fig. 6). Since $p_2 \preceq_2 p_4$, by Definition 2 there is a transition (p_4, a_2, c_4, q_4) in $\mathcal{A}_4$ such that $q_2 \preceq_2 q_4$. Therefore $q_2\mathcal{R}q_4$, proving the lemma.

□

Proposition 11. *Let $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ be finite trim automata. If $\mathcal{A}_1 \sqsubseteq \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq \mathcal{A}_4$ [resp. $\mathcal{A}_1 \sqsubseteq_{\mathrm{p}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq_{\mathrm{p}} \mathcal{A}_4$], then the pair $\mathcal{A}_1.\mathcal{A}_2 \sqsubseteq \mathcal{A}_3.\mathcal{A}_4$ [resp. $\mathcal{A}_1.\mathcal{A}_2 \sqsubseteq_{\mathrm{p}} \mathcal{A}_3.\mathcal{A}_4$].*

PROOF. Let π be a successful path of $\mathcal{A}_1.\mathcal{A}_2$. By definition of the sequential product, π can be decomposed into $\pi = \pi_1, ((p_1, p_2), a_2, c_2, q_2), \pi_2$, where π_1 is a path built up using only composed states, and $(p_2, a_2, c_2, q_2), \pi_2$ is a successful path of $\mathcal{A}_2$. Let φ be the projection that maps each transition $((p_1, p_2), a, c, (q_1, q_2))$ of $\mathcal{A}_1.\mathcal{A}_2$ between composed states

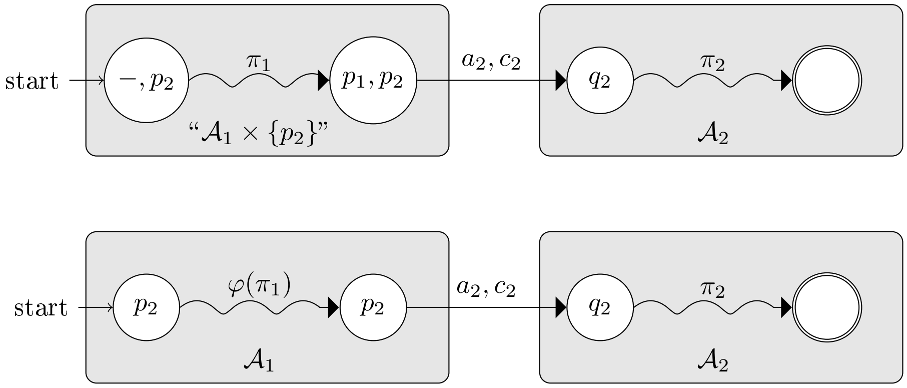

Figure 7: Proof of Proposition 11.

to (p_1, a, c, q_1). The function φ can be naturally extended to paths. Decompositions are illustrated in Fig. 7: the first line represents the decomposition of π and the second line the decomposition using φ.

By $[SEQ1]$, $\varphi(\pi_1)$ is a successful path of $\mathcal{A}_1$. Since $\mathcal{A}_1 \sqsubseteq \mathcal{A}_3$, there exists a path π_3 of $\mathcal{A}_3$ such that $\pi_1 \preceq_1 \pi_3$ and $\text{cost}(\pi_3) \leq \text{cost}(\pi_3)$. Similarly, since $\mathcal{A}_2 \sqsubseteq \mathcal{A}_4$, there exists a path π_4 such that $((p_2, a_2, c_2, q_2), \pi_2) \preceq_2 \pi_4$ and $\text{cost}((p_2, a_2, c_2, q_2), \pi_2) \leq \text{cost}(\pi_4)$. Let q_4 be the starting state of π_4, p_f be the ending state of π_3, and k be the length of π_3 (which is also the length of π_1). The sequence π' of transitions of $\mathcal{A}_3.\mathcal{A}_4$ defined by: if i is smaller than or equal to k, and if the i-th transition of π_3 is (r_i, a_i, c_i, r_{i+1}), then the i-th transition of π' is $((r_i, q_4), b_i, d_i, (r_{i+1}, q_4))$. If i is equal to $k+1$, then the i-th transition of π' is $((p_f, q_4), a_2, c_2, q_2)$. For the values of i greater than $k+1$, the i-th transition of π' is the $(i+k)$-th transition of π_4. Using $[SEQ1], [SEQ2]$ and $[SEQ3]$, one can easily check that π' is a successful path of $\mathcal{A}_3.\mathcal{A}_4$ such that that $\text{cost}(\pi') \leq \text{cost}(\pi)$. Moreover, by Lemma 10, $\pi \preceq \pi'$, proving the lemma for the substitutivity problem.

The proof still works for the partial substitutivity problem. □

Unfortunately, Proposition 11 does not hold for (partial) strong substitutivity problems. Indeed, let us consider the following four automata:

$\mathcal{A}_1 =$ $(\{p_1\}, \{a\}, \{(p_1, a, 1, p_1)\}, \{p_1\}, \{p_1\})$,
$\mathcal{A}_3 =$ $(\{q_1\}, \{a\}, \{(q_1, a, 0, q_1)\}, \{q_1\}, \{q_1\})$,
$\mathcal{A}_2 =$ $(\{p_2\}, \{a\}, \{(p_2, a, 4, p_2)\}, \{p_2\}, \{p_2\})$,
$\mathcal{A}_4 =$ $(\{q_2\}, \{a\}, \{(q_2, a, 3, q_2)\}, \{q_2\}, \{q_2\})$,

Pairs of automata $\mathcal{A}_1, \mathcal{A}_3$ and $\mathcal{A}_2, \mathcal{A}_4$ both trivially satisfy the strong substitutivity problem and the partial strong substitutivity problems. However, when considering the pair $\mathcal{A}_1.\mathcal{A}_2, \mathcal{A}_3.\mathcal{A}_4$, one has

$$\{((p_1, p_2), (q_1, q_2)), ((p_1, p_2), q_2), (p_2, q_2)\} \subseteq \preceq_{\mathcal{A}_1.\mathcal{A}_2, \mathcal{A}_3.\mathcal{A}_4} .$$

Consequently,

$$((p_1, p_2), a, 1, (p_1, p_2))((p_1, p_2), a, 4, p_2) \preceq ((q_1, q_2), a, 3, q_2)(q_2, a, 3, q_2).$$

But these paths do not satisfy the weight conditions of the strong and the partial strong substitutivity problems. Intuitively, a sequential composition of automata may create new ways to perform a sequence of actions: these new ways may have costs that do not fulfil the universal weight condition required by the strong substitutivity.

However, when the automata in the pair have disjoint alphabets, the following composition result holds.

Proposition 12. *Let $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ be finite trim automata [resp. are finite trim automata such that $\mathcal{A}_1.\mathcal{A}_2$ and $\mathcal{A}_3.\mathcal{A}_4$ are both trim] such that $\mathcal{A}_1$ and $\mathcal{A}_2$ have disjoint alphabets. If $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_4$ [resp. $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}}_{\mathrm{p}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq^{\mathrm{st}}_{\mathrm{p}} \mathcal{A}_4$], then the pair $\mathcal{A}_1.\mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_3.\mathcal{A}_4$ [resp. $\mathcal{A}_1.\mathcal{A}_2 \sqsubseteq^{\mathrm{st}}_{\mathrm{p}} \mathcal{A}_3.\mathcal{A}_4$].*

Proof. Assume that $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_4$. Let π be a successful path of $\mathcal{A}_1.\mathcal{A}_2$. By Proposition 11, there exists a successful path of $\mathcal{A}_3.\mathcal{A}_4$ similar to π with a lower cost.

First we claim that the relation $\mathcal{R}$ defined in Lemma 10 is the largest simulation relation. Remark that since transitions that can be fired from non composed states of $\mathcal{A}_1.\mathcal{A}_2$ are exactly the transitions of $\mathcal{A}_2$ and since $\mathcal{A}_2$ and $\mathcal{A}_1.\mathcal{A}_2$ have the same set of final states, if $p_2 \preceq_{\mathcal{A}_1.\mathcal{A}_2,\mathcal{A}_3.\mathcal{A}_4} p_4$, then $p_2 \preceq_{\mathcal{A}_2,\mathcal{A}_4} p_4$. Now if $(p_1,p_2) \preceq_{\mathcal{A}_1.\mathcal{A}_2,\mathcal{A}_3.\mathcal{A}_4} (p_3,p_4)$, then $p_1 \preceq_{\mathcal{A}_1,\mathcal{A}_3} p_3$ and $p_2 \preceq_{\mathcal{A}_2,\mathcal{A}_4} p_4$. For every transition (p_2, a_3, c_3, r_2) of $\mathcal{A}_2$, there exists a state transition in $\mathcal{A}_3.\mathcal{A}_4$ from (p_3,p_4) labelled by a_3 to a state related to r_2 by $\preceq_{\mathcal{A}_1.\mathcal{A}_2,\mathcal{A}_3.\mathcal{A}_4}$. According to the assumption on the alphabet, this state, denoted r_4, is not a composed state. Therefore (using the above remark) $r_2 \preceq_{\mathcal{A}_2,\mathcal{A}_4} r_4$, proving the claim.

One can now prove the proposition. Consider a path π' of $\mathcal{A}_3.\mathcal{A}_4$ such that $\pi \preceq \pi'$. The path π' can be decomposed into $\pi' = \pi_3, ((p_3,p_4),a,c,q_4), \pi_4$ such that π_3 is a successful path of $\mathcal{A}_3$ and $((p_3,p_4),a,c,q_4), \pi_4$ is a successfully path of $\mathcal{A}_4$. Similarly π can be decomposed into $\pi = \pi_1, ((p_1,p_2),b,d,q_2), \pi_2$ such that π_1 is a success-full path of $\mathcal{A}_1$ and $((p_1,p_2),b,d,q_2), \pi_2$ is a success-full path of $\mathcal{A}_2$. Since $\pi \preceq \pi'$ and by alphabet conditions, π_1 and π_3 have the same length, $a = b$ and, π_2 and π_4 have the same length. Now inductively using the claim (resp. the remark) on states of π_1 and π_3 (resp. of $((p_1,p_2),b,d,q_2)\pi_2$ and $((p_3,p_4),a,c,q_4), \pi_4)$, one has $\pi_1 \preceq_{\mathcal{A}_1,\mathcal{A}_3} \pi_3$ (resp. $((p_1,p_2),b,d,q_2), \pi_2 \preceq_{\mathcal{A}_2,\mathcal{A}_4} ((p_3,p_4),a,c,q_4), \pi_4)$. Since $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_4$, one has $\mathrm{cost}(\pi_3) \leq \mathrm{cost}(\pi_1)$ and $\mathrm{cost}(((p_3,p_4),a,c,q_4),\pi_4) \leq \mathrm{cost}(((p_1,p_2),b,d,q_2),\pi_2)$. Therefore, $\mathrm{cost}(\pi') \leq \mathrm{cost}(\pi)$, which concludes the proof.

The proof for the strong substitutivity problem is very close to the above proof. □

Let us consider a variant of the sequential composition of automata, called the strict sequential product, where additional transitions with a special label are introduced. This label allows one to identify parts of a path w.r.t. composed automata.

Definition 13. *Let $\mathcal{A}_1 = (Q_1, A_1, E_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q_2, A_2, E_2, I_2, F_2)$ be two integer weighted automata. The strict sequential composition of $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted $\mathcal{A}_1 \rightarrow \mathcal{A}_2$, is the automaton $\mathcal{A}_{12} = (Q_1 \cup Q_2, A_1 \cup A_2 \cup \{\delta\}, E_1 \cup E_2 \cup E_{12}, I_1, F_2)$ where $\delta \notin A_1 \cup A_2$ and $E_{12} = \{(p, \delta, 0, q) \mid p \in F_1,\ q \in I_2\}$.*

For our running automata $\mathcal{A}_{exe1}$ and $\mathcal{A}_{exe2}$ in Fig. 3, their strict sequential product $\mathcal{A}_{exe1} \rightarrow \mathcal{A}_{exe2}$ is depicted in Fig. 8.

Proposition 14. *Let $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ be finite trim automata. If $\mathcal{A}_1 \sqsubseteq \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq \mathcal{A}_4$ [resp. $\mathcal{A}_1 \sqsubseteq_{\mathrm{p}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq \mathcal{A}_4$] [resp. $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_4$] [resp. $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}}_{\mathrm{p}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_4$] , then $\mathcal{A}_1 \rightarrow \mathcal{A}_2 \sqsubseteq \mathcal{A}_3 \rightarrow \mathcal{A}_4$ [resp.$\mathcal{A}_1 \rightarrow \mathcal{A}_2 \sqsubseteq_{\mathrm{p}} \mathcal{A}_3 \rightarrow \mathcal{A}_4$] [resp. $\mathcal{A}_1 \rightarrow \mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_3 \rightarrow \mathcal{A}_4$] [resp. $\mathcal{A}_1 \rightarrow \mathcal{A}_2 \sqsubseteq^{\mathrm{st}}_{\mathrm{p}} \mathcal{A}_3 \rightarrow \mathcal{A}_4$] .*

Proof. The relation $\mathcal{R}$ between states of $\mathcal{A}_1 \rightarrow \mathcal{A}_2$ and states of $\mathcal{A}_3 \rightarrow \mathcal{A}_4$ is defined as follows: $p\mathcal{R}q$ if and only if either p is a state of $\mathcal{A}_1$ and q a state of $\mathcal{A}_3$ and $p \preceq_{\mathcal{A}_1,\mathcal{A}_3} q$, or p is a state of $\mathcal{A}_2$ and q a state of $\mathcal{A}_4$ and $p \preceq_{\mathcal{A}_2,\mathcal{A}_4} q$. One can easily check (as for Lemma 10) that $\mathcal{R}$ is a simulation relation.

The proof is structured as follows: firstly, (part 1), we prove the proposition for the substitutivity problem. Secondly, (part 2), we show that $\mathcal{R}$ is the largest simulation relation

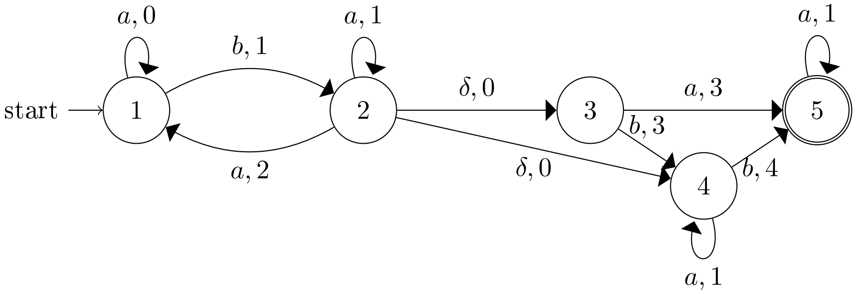

Figure 8: Automaton $\mathcal{A}_{exe1} \to \mathcal{A}_{exe2}$

between $\mathcal{A}_1 \to \mathcal{A}_2$ and $\mathcal{A}_3 \to \mathcal{A}_4$. This leads to the final third step (part 3), where we prove the proposition for the strong substitutivity problem. Proofs for partial (strong) substitutivity problems are very similar and left to the reader. Notice that since $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ are finite trim automata, so are $\mathcal{A}_1 \to \mathcal{A}_2$ and $\mathcal{A}_3 \to \mathcal{A}_4$.

(Part 1):
Let π be a successful path in $\mathcal{A}_1 \to \mathcal{A}_2$. By construction of $\mathcal{A}_1 \to \mathcal{A}_2$, π can be decomposed into $\pi_1, (p_1, \delta, p_2), \pi_2$, where π_1 is a successful path of $\mathcal{A}_1$, π_2 is a successful path of $\mathcal{A}_2$, p_1 is a final state of $\mathcal{A}_1$ and p_2 an initial state of $\mathcal{A}_2$.

Assume that $\mathcal{A}_1 \sqsubseteq \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq \mathcal{A}_4$, then there exist successful paths π_3 of $\mathcal{A}_3$ and π_4 of $\mathcal{A}_4$ such that $\pi_1 \preceq_{\mathcal{A}_1,\mathcal{A}_3} \pi_2$, $\pi_3 \preceq_{\mathcal{A}_3,\mathcal{A}_4} \pi_4$, $\mathrm{cost}(\pi_3) \leq \mathrm{cost}(\pi_1)$ and $\mathrm{cost}(\pi_4) \leq \mathrm{cost}(\pi_2)$. Let p_3 be the ending state of π_3, and p_4 the starting state of p_4. Since π_3 is a successful path in $\mathcal{A}_3$, p_3 is a final state of $\mathcal{A}_3$. Similarly, p_4 is an initial state of $\mathcal{A}_4$. Consequently, $\pi_3, (p_3, \delta, 0, p_4), \pi_4$ is a successful path of $\mathcal{A}_3 \to \mathcal{A}_4$. Moreover, by construction, $\pi \mathcal{R} (\pi_3, (p_3, \delta, 0, p_4), \pi_4)$. Thus $\pi \preceq \pi_3, (p_3, \delta, 0, p_4), \pi_4$. Furthermore, $\mathrm{cost}(\pi_2) \leq \mathrm{cost}(\pi_1)$ and $\mathrm{cost}(\pi_4) \leq \mathrm{cost}(\pi_3)$ ensure that $\mathrm{cost}(\pi_3, (p_3, \delta, 0, p_4), \pi_4) \leq \mathrm{cost}(\pi)$, proving the proposition for the substitutivity problem.

(Part 2):
We claim that $\mathcal{R} = \preceq$, i.e. that $\mathcal{R}$ is the largest simulation relation between $\mathcal{A}_1 \to \mathcal{A}_2$ and $\mathcal{A}_3 \to \mathcal{A}_4$. Indeed, let p be a state of $\mathcal{A}_1 \to \mathcal{A}_2$, and q be a state of $\mathcal{A}_3 \to \mathcal{A}_4$ such that $p \preceq_{\mathcal{A}_1 \to \mathcal{A}_2, \mathcal{A}_3 \to \mathcal{A}_4} q$. Following cases arise:

(1) Assume that p is a state of $\mathcal{A}_2$. Since $\mathcal{A}_2$ is trim, there exists a path in $\mathcal{A}_2$ from p to a final state of $\mathcal{A}_2$. Now, the assumption $p \preceq_{\mathcal{A}_1 \to \mathcal{A}_2, \mathcal{A}_3 \to \mathcal{A}_4} q$ implies that there is a similar path in $\mathcal{A}_4$. Since δ doesn't occur in the label of this path, q is a state of $\mathcal{A}_4$. Since the restriction of $\preceq_{\mathcal{A}_1 \to \mathcal{A}_2, \mathcal{A}_3 \to \mathcal{A}_4}$ to states of $\mathcal{A}_2$ and states of $\mathcal{A}_4$ is a simulation relation, one has $p \preceq_{\mathcal{A}_2,\mathcal{A}_4} q$. Therefore $p\mathcal{R}q$.

(2) Assume that p is a state of $\mathcal{A}_1$. We will show by contradiction that q is a state of $\mathcal{A}_3$. Assume that q is a state of $\mathcal{A}_4$. Since $\mathcal{A}_1$ and $\mathcal{A}_3$ are trim, there is a path in $\mathcal{A}_1 \to \mathcal{A}_2$ from p to a final state. By construction, δ occurs in the label of this path. Since $p \preceq_{\mathcal{A}_1 \to \mathcal{A}_2, \mathcal{A}_3 \to \mathcal{A}_4} q$, there is a similar path in $\mathcal{A}_3 \to \mathcal{A}_4$. But q is a state of $\mathcal{A}_4$, thus there is no path from q whose label contains δ, a contradiction. Therefore, $q \in \mathcal{A}_3$. As for case (1), this ensures that $p\mathcal{R}q$, proving the claim.

(Part 3):
Assume that $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_4$. According to above proof, it remains to show that for every successful path π of $\mathcal{A}_1 \to \mathcal{A}_2$ and every successful path π' of $\mathcal{A}_3 \to \mathcal{A}_4$, if $\pi \preceq \pi'$, then $\mathrm{cost}(\pi') \leq \mathrm{cost}(\pi)$. Assuming that π is a successful path of $\mathcal{A}_1 \to \mathcal{A}_2$

and that π' is a successful path of $\mathcal{A}_3 \to \mathcal{A}_4$, the path π can be decomposed into $\pi = \pi_1, (p_1, \delta, p_2), \pi_2$ and the path π' into $\pi' = \pi_3, (p_3, \delta, p_4), \pi_4$. Symbol δ occurs only once in the label of π and in the label of π'. Thus, by length argument, if $\pi \preceq \pi'$, using the claim (point 2), one has $\pi_1 \preceq_{\mathcal{A}_1,\mathcal{A}_3} \pi_3$ and $\pi_2 \preceq_{\mathcal{A}_2,\mathcal{A}_4} \pi_4$. It follows that $\mathrm{cost}(\pi_3) \leq \mathrm{cost}(\pi_1)$ and $\mathrm{cost}(\pi_4) \leq \mathrm{cost}(\pi_2)$. Consequently, $\mathrm{cost}(\pi') \leq \mathrm{cost}(\pi)$, proving the proposition for the strong substitutivity problem.

□

6.2. Substitutivity and Parallel Composition

We now define a parallel composition operator and offer the positive and negative results on the compatibility of the substitutivity with relation to the parallel composition.

Definition 15. *Let $\mathcal{A}_1 = (Q_1, A_1, E_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q_2, A_2, E_2, I_2, F_2)$ be two integer weighted automata. The parallel product of $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted $\mathcal{A}_1 \otimes \mathcal{A}_2$, is the automaton $\mathcal{A}_{12} = (Q_{12}, A_{12}, E_{12}, I_{12}, F_{12})$ where*

- $Q_{12} = \{(p_1, p_2) \mid p_1 \in Q_1,\ p_2 \in Q_2\}$,
- $A_{12} = A_1 \cup A_2$,
- $I_{12} = I_1 \times I_2$,
- $F_{12} = F_1 \times F_2$,

and where the transition relation E_{12} obeys the following rules:

$$\textit{[SYNC]}\ \ \frac{p_1 \xrightarrow{a_1,c_1}_{\mathcal{A}_1} q_1, p_2 \xrightarrow{a_2,c_2}_{\mathcal{A}_2} q_2}{(p_1,p_2) \xrightarrow{a,c_1+c_2}_{\mathcal{A}_1\otimes\mathcal{A}_2} (q_1,q_2)}\ \ a \in A_1 \cap A_2$$

$$\textit{[PAR1]}\ \ \frac{p_1 \xrightarrow{a_1,c_1}_{\mathcal{A}_1} q_1}{(p_1,p_2) \xrightarrow{a_1,c_1}_{\mathcal{A}_1\otimes\mathcal{A}_2} (q_1,p_2)}\ \ a_1 \in A_1 \setminus A_2$$

$$\textit{[PAR2]}\ \ \frac{p_2 \xrightarrow{a_2,c_2}_{\mathcal{A}_2} q_2}{(p_1,p_2) \xrightarrow{a_2,c_2}_{\mathcal{A}_1\otimes\mathcal{A}_2} (p_1,q_2)}\ \ a_2 \in A_2 \setminus \mathcal{A}_1$$

The parallel composition *of $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted $\mathcal{A}_1 \| \mathcal{A}_2$, is the automaton obtained by deleting in $\mathcal{A}_1 \otimes \mathcal{A}_2$ states (and related transitions) that are not co-accessible.*

Consider, for instance, the two automata $\mathcal{A}_{exe3}$ and $\mathcal{A}_{exe4}$ depicted in Fig. 9. The automata $\mathcal{A}_{exe3} \otimes \mathcal{A}_{exe4}$ and $\mathcal{A}_{exe3} \| \mathcal{A}_{exe4}$ are respectively displayed in Fig. 10 and Fig. 11.

Proposition 16. *Let $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ be finite trim automata [resp. are finite trim automata such that $\mathcal{A}_1 \| \mathcal{A}_2$ and $\mathcal{A}_3 \| \mathcal{A}_4$ are both trim]. If $\mathcal{A}_1 \sqsubseteq \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq \mathcal{A}_4$ [resp. $\mathcal{A}_1 \sqsubseteq_{\mathrm{p}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq_{\mathrm{p}} \mathcal{A}_4$], then $\mathcal{A}_1 \| \mathcal{A}_2 \sqsubseteq \mathcal{A}_3 \| \mathcal{A}_4$ satisfies the substitutivity problem [resp. $\mathcal{A}_1 \| \mathcal{A}_2 \sqsubseteq_{\mathrm{p}} \mathcal{A}_3 \| \mathcal{A}_4$].*

Proof. In this proof A_1 is the common alphabet of $\mathcal{A}_1$ and $\mathcal{A}_3$ and A_2 is the common alphabet of $\mathcal{A}_2$ and $\mathcal{A}_4$

The relation $\mathcal{R}$ between states of $\mathcal{A}_1 \| \mathcal{A}_2$ and states of $\mathcal{A}_3 \| \mathcal{A}_4$ is defined as follows: $(p_1, p_2) \mathcal{R} (p_3, p_4)$ if and only if $p_1 \preceq_{\mathcal{A}_1,\mathcal{A}_3} p_3$ and $p_2 \preceq_{\mathcal{A}_2,\mathcal{A}_4} p_4$. The proof is divided into two

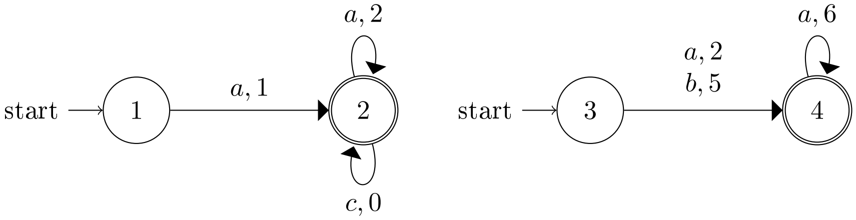

Figure 9: Automata $\mathcal{A}_{exe3}$ and $\mathcal{A}_{exe4}$

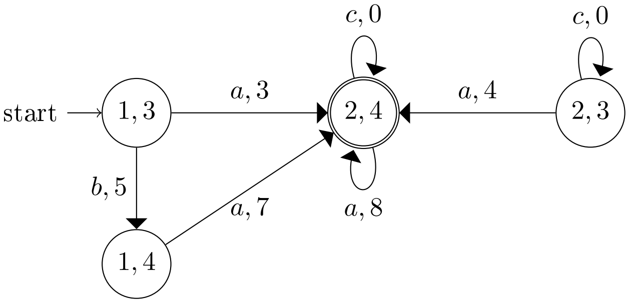

Figure 10: Automaton $\mathcal{A}_{exe3} \otimes \mathcal{A}_{exe4}$

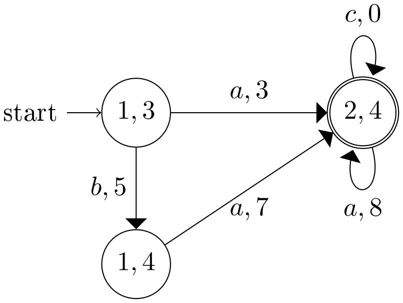

Figure 11: Automaton $\mathcal{A}_{exe3} \| \mathcal{A}_{exe4}$

parts: Firstly, in (Part 1), we prove that $\mathcal{R}$ is a simulation relation. Secondly, in (Part 2), we prove the proposition for the substitutivity problem.

(Part 1):

We first prove that relation $\mathcal{R}$ is a simulation relation. Indeed, if (p_1, p_2) is final then, by definition of $\mathcal{A}_1 \otimes \mathcal{A}_2$, p_1 and p_2 are respectively final states of $\mathcal{A}_1$ and $\mathcal{A}_2$. Then, if $p_1 \preceq_{\mathcal{A}_1,\mathcal{A}_3} p_3$, p_3 is final, and if $p_2 \preceq_{\mathcal{A}_2,\mathcal{A}_4} p_4$, p_4 is final, proving $\mathcal{R}$ satisfies condition *ii*) of Definition 2. Now it remains to prove condition *i*). Assume that $(p_1, p_2)\mathcal{R}(p_3, p_4)$. The following three cases arise:

(1) If there exists a transition $((p_1, p_2), a_1, c_1, (q_1, p_2))$ in $\mathcal{A}_1 \| \mathcal{A}_2$, with $a_1 \in A_1 \setminus A_2$, it is obtained by applying $[PAR1]$. So, there exist a transition (p_1, a_1, c_1, q_1) in $\mathcal{A}_1$ and a state q_3 of $\mathcal{A}_3$ such that $p_1 \preceq_{\mathcal{A}_1,\mathcal{A}_3} q_3$. Since (q_1, p_2) is accessible and co-accessible in $\mathcal{A}_1 \otimes \mathcal{A}_2$, so is (q_3, p_4) in $\mathcal{A}_3 \otimes \mathcal{A}_4$. It follows that (q_3, p_4) is a state of $\mathcal{A}_3 \| \mathcal{A}_4$ satisfying $(q_1, p_2)\mathcal{R}(q_3, p_4)$.

(2) If a transition from (p_1, p_2) is fired by applying $[PAR2]$, one can prove, as for case (1), that condition *i*) of Definition 2 is satisfied.

(3) If a transition from (p_1, p_2) is fired by applying $[SYNC]$, then there exist a transition (p_1, a, c_1, q_1) in $\mathcal{A}_1$ and a transition (p_2, a, c_2, q_2) such that $a \in A_1 \cap A_2$. Since $(p_1, p_2)\mathcal{R}(p_3, p_4)$, there are q_3 in $\mathcal{A}_3$ and q_4 in $\mathcal{A}_4$ and transitions (p_3, a, c_3, q_3) and (p_4, a, c_4, q_4) in respectively $\mathcal{A}_3$ and $\mathcal{A}_4$ such that $q_1 \preceq_{\mathcal{A}_1,\mathcal{A}_3} q_3$ and $q_2 \preceq_{\mathcal{A}_2,\mathcal{A}_4} q_4$. Since (q_3, q_4) is both an accessible and co-accessible state of $\mathcal{A}_1 \otimes \mathcal{A}_2$, $(q_1, q_2)\mathcal{R}(q_3, q_4)$, proving that $\mathcal{R}$ is a simulation relation.

(Part 2):

Now we will prove the proposition for the substitutivity problem. Assume that $\mathcal{A}_1 \sqsubseteq \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq \mathcal{A}_4$. Let π be a successful path in $\mathcal{A}_1 \| \mathcal{A}_2$. We denote by φ_i ($i \in \{1, 2\}$), the partial function that maps transitions of $\mathcal{A}_1 \| \mathcal{A}_2$ to transitions of $\mathcal{A}_i$ as follows: a transition $((p_1, p_2), a, c, (q_1, q_2))$ of $\mathcal{A}_1 \| \mathcal{A}_2$ is mapped to (p_i, a, c, q_i) if $a \in A_i$. Partial functions φ_i ($i \in \{1, 2\}$) are extended to sequences of transitions: $\varphi_i(t_1, \ldots, t_k) = \varphi_i(t_1), \ldots, \varphi_i(t_k)$ with the convention that if $\varphi_i(t)$ is not defined, then $\varphi_i(t)$ is mapped to the empty path. For instance, if t_1, t_2, t_3 are three transition respectively labelled by letter of $A_1 \cap A_2$, $A_1 \setminus A_2$ and $A_2 \setminus A_1$, then $\varphi_1(t_1, t_2, t_3) = \varphi_1(t_1), \varphi_1(t_2)$ and $\varphi_2(t_1, t_2, t_3) = \varphi_1(t_1), \varphi_1(t_3)$.

Let π be a successful path in $\mathcal{A}_1 \| \mathcal{A}_2$. One can easily check that $\varphi_i(\pi)$ is a successful path of $\mathcal{A}_i$. Therefore there are successful paths π_3 and π_4 of respectively $\mathcal{A}_3$ and $\mathcal{A}_4$ such that $\varphi_1(\pi) \preceq_{\mathcal{A}_1,\mathcal{A}_3} \pi_3$, $\varphi_2(\pi) \preceq_{\mathcal{A}_1,\mathcal{A}_3} \pi_4$, $\text{cost}(\pi_3) \leq \text{cost}(\varphi_1(\pi))$ and $\text{cost}(\pi_4) \leq \text{cost}(\varphi_2(\pi))$. We inductively define the finite sequences of integers α_i and β_i by

- $\alpha_1 = 1$ and $\beta_1 = 1$,
- if the i-th transition of π is labelled by a letter in $A_1 \cap A_2$, then $\alpha_{i+1} = 1 + \alpha_1$ and $\beta_{i+1} = 1 + \beta_i$,
- if the i-th transition of π is labelled by a letter in $A_1 \setminus A_2$, then $\alpha_{i+1} = 1 + \alpha_1$ and $\beta_{i+1} = \beta_i$,
- if the i-th transition of π is labelled by a letter in $A_2 \setminus A_1$, then $\alpha_{i+1} = \alpha_1$ and $\beta_{i+1} = 1 + \beta_i$.

Informally, when running the path π, each time a transition labelled by a letter in A_i is met, the corresponding counter (α for A_1 and β for A_2) increases.

Now we define the sequence of transitions π' of $\mathcal{A}_3 \otimes \mathcal{A}_4$ by:

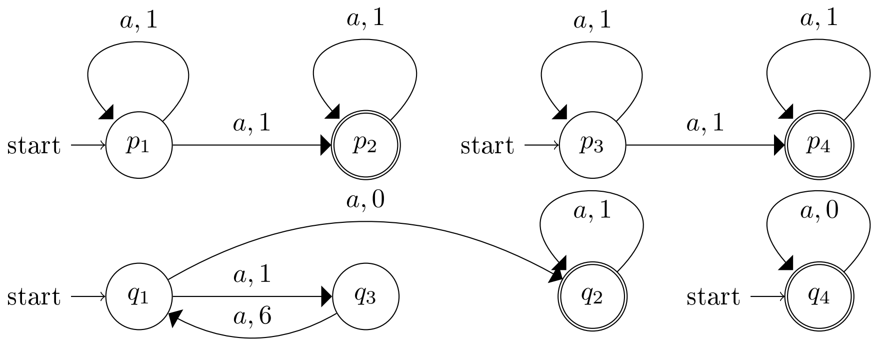

Figure 12: Automata for a counter-example on Proposition 16

- If the i-th transition of π is labelled by a letter a in $A_1 \cap A_2$, then the i-th transition of π' is $((p_3, p_4), a, c_3 + c_4, (q_3, q_4))$ where (p_3, a, c_3, q_3) is the α_i-th transition of π_3 and (p_4, a, c_4, q_4) is the β_i-th transition of π_4.
- If the i-th transition of π is labelled by a letter a in $A_1 \setminus A_2$, then the i-th transition of π' is $((p_3, p_4), a, c_3, (q_3, p_4))$ where (p_3, a, c_3, q_3) is the α_i-th transition of π_3.
- If the i-th transition of π is labelled by a letter a in $A_2 \setminus A_1$, then the i-th transition of π' is $((p_3, p_4), a, c_4, (p_3, q_4))$ where (p_4, a, c_4, q_4) is the β_i-th transition of π_4.

One can easily check that α_i is less or equal to the length of π_1 (equivalently the length of π_3) and that β_i is less or equal to the length of π_2 (equivalently the length of π_4). Thus, following rules $[PAR1]$, $[PAR2]$ and $[SYNC]$, π' is well-defined. By a direct induction, one can prove that π' is a successful path of $\mathcal{A}_3 \| \mathcal{A}_4$ satisfying $\pi \mathcal{R} \pi'$ and $\text{cost}(\pi') \leq \text{cost}(\pi)$. Since $\mathcal{R}$ is, by definition, included in $\preceq_{\mathcal{A}_1 \| \mathcal{A}_2, \mathcal{A}_3 \| \mathcal{A}_4}$, the proof for the substitutivity problem is complete.

□

Unfortunately, Proposition 16 does not hold for (partial) strong substitutivity problems. Consider, for instance, the following automata (depicted in Fig. 12):

$\mathcal{A}_1 =$ $(\{p_1, p_2\}, \{a\}, \{(p_1, a, 1, p_1), (p_1, a, 1, p_2), (p_2, a, 1, p_2)\}, \{p_1\}, \{p_2\})$,
$\mathcal{A}_3 =$ $(\{q_1, q_2, q_3\}, \{a\}, \{(q_1, a, 1, q_3), (q_3, a, 6, q_1), (q_1, a, 0, q_2), (q_2, a, 1, q_2)\}, \{q_1\}, \{q_2\})$,
$\mathcal{A}_2 =$ $(\{p_3, p_4\}, \{a\}, \{(p_3, a, 1, p_3), (p_3, a, 1, p_4), (p_4, a, 1, p_4)\}, \{p_3\}, \{p_4\})$,
$\mathcal{A}_4 =$ $(\{q_4\}, \{a\}, \{(q_4, a, 0, q_4)\}, \{q_4\}, \{q_4\})$.

Both pairs of automata $\mathcal{A}_1, \mathcal{A}_3$ and $\mathcal{A}_2, \mathcal{A}_4$ satisfy the strong substitutivity problem. But $(p_1, p_3) \preceq_{\mathcal{A}_1 \| \mathcal{A}_2, \mathcal{A}_3 \| \mathcal{A}_4} (q_1, q_4)$, $(p_1, p_3) \preceq_{\mathcal{A}_1 \| \mathcal{A}_2, \mathcal{A}_3 \| \mathcal{A}_4} (q_3, q_4)$ and $(p_2, p_4) \preceq_{\mathcal{A}_1 \| \mathcal{A}_2, \mathcal{A}_3 \| \mathcal{A}_4} (q_2, q_4)$. Therefore the successful paths

$$\pi_{12} = ((p_1, p_3), a, 2, (p_1, p_3))((p_1, p_3), a, 2, (p_1, p_3))((p_1, p_3), a, 2, (p_2, p_4))$$

and

$$\pi_{34} = ((q_1, q_4), a, 1, (q_3, q_4))((q_3, q_4), a, 6, (q_1, q_4))((q_1, q_4), a, 0, (q_2, q_4))$$

satisfy $\pi_{12} \preceq_{\mathcal{A}_1 \| \mathcal{A}_2, \mathcal{A}_3 \| \mathcal{A}_4} \pi_{34}$. But $\text{cost}(\pi_{12}) = 6$ and $\text{cost}(\pi_{34}) = 7$.

However, as for the sequential composition, one has the following result for pairs of automata with disjoint alphabets.

Proposition 17. *Let $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ be finite trim automata such that $\mathcal{A}_1$ and $\mathcal{A}_2$ have disjoint alphabets and, $\mathcal{A}_3$ and $\mathcal{A}_4$ have disjoint alphabets. If $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_4$ [resp. $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}}_{\mathrm{p}} \mathcal{A}_3$ and $\mathcal{A}_2, \sqsubseteq^{\mathrm{st}}_{\mathrm{p}} \mathcal{A}_4$], then the pair $\mathcal{A}_1\|\mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_3\|\mathcal{A}_4$ [resp. $\mathcal{A}_1\|\mathcal{A}_2 \sqsubseteq^{\mathrm{st}}_{\mathrm{p}} \mathcal{A}_3\|\mathcal{A}_4$].*

Proof. Assume that both couples $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_4$. Let π be a successful path of $\mathcal{A}_1\|\mathcal{A}_2$. By Proposition 16, there exists a successful path of $\mathcal{A}_3\|\mathcal{A}_4$ similar to π with a smaller cost.

We claim that if $(p_1, p_2) \preceq_{\mathcal{A}_1\|\mathcal{A}_2,\mathcal{A}_3\|\mathcal{A}_4} (p_3, p_4)$, then $p_1 \preceq_{\mathcal{A}_1\|\mathcal{A}_2} p_3$ and $p_2 \preceq_{\mathcal{A}_3\|\mathcal{A}_4} p_4$. It suffices to prove that $p_1 \preceq_{\mathcal{A}_1\|\mathcal{A}_2} p_3$ because of the case symmetry. Assume that $p_1 \not\preceq_{\mathcal{A}_1\|\mathcal{A}_2} p_3$. Then the following cases may arise:

(1) p_1 is a final state of $\mathcal{A}_1$ whereas p_3 is not. Since $\mathcal{A}_2$ is trim, there exists a path in $\mathcal{A}_2$ from p_2 to a final state q_2 of $\mathcal{A}_2$. This path is labelled by letters in the $\mathcal{A}_2$ alphabet. Therefore there is a path in $\mathcal{A}_1\|\mathcal{A}_2$ from (p_1, p_2) to (p_1, q_2). Since $(p_1, p_2) \preceq_{\mathcal{A}_1\|\mathcal{A}_2,\mathcal{A}_3\|\mathcal{A}_4} (p_3, p_4)$, there is a similar path in $\mathcal{A}_3\|\mathcal{A}_4$ to a state of the form (p_3, q_4). Now since q_2 is a final state in $\mathcal{A}_2$, so is (p_1, q_2) in $\mathcal{A}_1\|\mathcal{A}_2$. But $(p_1, q_2) \preceq_{\mathcal{A}_1\|\mathcal{A}_2,\mathcal{A}_3\|\mathcal{A}_4} (p_3, q_4)$, so, (p_3, q_4) is final in $\mathcal{A}_3\|\mathcal{A}_4$. Consequently, p_3 is final, a contradiction.

(2) There is a transition in $\mathcal{A}_1$ starting from p_1 labelled by a, but no transition labelled by a starts from p_3 in $\mathcal{A}_3$. Therefore there is a transition in $\mathcal{A}_1\|\mathcal{A}_2$ starting from (p_1, p_2) labelled by a. Since a is not a letter from the alphabet of $\mathcal{A}_3$, no transition labelled by a in $\mathcal{A}_3\|\mathcal{A}_4$ can be fired from (p_3, p_4), a contradiction.

(3) There is a transition (p_1, a_1, c_1, q_1) of $\mathcal{A}_1$ such that for every transition of the form (p_3, a_1, c_3, q_3) of $\mathcal{A}_2$, $q_1 \not\preceq_{\mathcal{A}_1,\mathcal{A}_3} q_3$. Iterating this construction, one can reach states $(p_1', p2)$ and (p_3', p_3) satisfying conditions of case (1), proving the claim.

Now let π' be a path in $\mathcal{A}_3\|\mathcal{A}_4$ such that $\pi \preceq_{\mathcal{A}_1\|\mathcal{A}_2,\mathcal{A}_3\|\mathcal{A}_4} \pi'$. Let π_3' be the sequence of transitions obtained by deleting in π' all the transitions labelled by a letter in the $\mathcal{A}_4$ alphabet. Let also π_4' be the sequence of transitions obtained by deleting in π' all the transitions labelled by a letter in the alphabet of $\mathcal{A}_3$. Using the hypotheses on the alphabets, one can easily check that the projection π_3' of π' on $\mathcal{A}_3$ is a successful path of $\mathcal{A}_3$. Similarly, the projection π_4' of π' on $\mathcal{A}_4$ is a successful path of $\mathcal{A}_4$. Following the same way, π can be projected to produce a successful path π_1 of $\mathcal{A}_1$ and a successful path π_2 of $\mathcal{A}_2$. The claim ensures that $\pi_1 \preceq_{\mathcal{A}_1,\mathcal{A}_3} \pi_3'$ and that $\pi_2 \preceq_{\mathcal{A}_1,\mathcal{A}_3} \pi_4'$. Now remind that both couples $\mathcal{A}_1 \sqsubseteq^{\mathrm{st}} \mathcal{A}_3$ and $\mathcal{A}_2 \sqsubseteq^{\mathrm{st}} \mathcal{A}_4$ satisfy the strong substitutivity problem. Thus $\mathrm{cost}(\pi_3') \leq \mathrm{cost}(\pi_1)$ and $\mathrm{cost}(\pi_4') \leq \mathrm{cost}(\pi_2)$. Since $\mathrm{cost}(\pi) = \mathrm{cost}(\pi_1) + \mathrm{cost}(\pi_2)$ and since $\mathrm{cost}(\pi') = \mathrm{cost}(\pi_3') + \mathrm{cost}(\pi_4')$, one has $\mathrm{cost}(\pi') \leq \mathrm{cost}(\pi)$, proving the proposition.

The proof for the partial strong substitutivity problem is similar and left to the reader. □

7. Practical Issues

As explained in Section 1, this paper is dedicated to component and service substitutivity with a special emphasis on their assembly. The challenge is to build trustworthy systems which satisfy both functional and non functional requirements. The obtained theoretical results have practical applications. Indeed, the methodological and practical approaches we have been developing through various project collaborations rest on them. These approaches can be summarised by:

1. The construction of trustworthy software systems from existing components.
2. An incremental approach to specify and verify component assembling.
3. The elicitation of non-functional requirements and their integration in the specification.

7.1. Application to Web Services with QoS

There are numerous works on automata-based analyses of service composition (see [tBBG07] for a survey). In the setting of the present paper, i.e. without silent τ-transitions, the $\preceq$-simulation relation is compatible with a sequential composition operator modelling e.g. the sequence BPEL structured activities, and with an asynchronous parallel composition operator implementing e.g. the flow BPEL structured activities. Notice that for the flow activities, the encoding would work without the source/target links that would somehow be encoded through a synchronisation. Both BPEL operators are important in practice since they allow building complex services by a composition of services.

An algorithm for the trace-based substitutivity problem has been implemented. The tests have been performed on different versions of a movie store example, a book store example provided by Oracle [Jur05], and the classical loan approval example. We intend to continue the implementation and extend that work to simulation-based substitutivity problems presented in this paper.

7.2. Application to Embedded Components

Thousands of systems in very various domains such as telecommunications, transportation, home automation (also called smart homes or domotics), system-on-chip, etc. are equipped with smart devices or "intelligent" components. They embed a growing software part which is often critical for the safety of the global system. Embedded systems whose resources are in general limited must satisfy both functional and non functional properties to optimise the use of their resources (memory, energy, etc.).

Within the application domain of land transportation systems, different models of a localisation system are proposed[4]. A localisation composite component, which is a critical part of land transportation systems, is made up of different positioning systems, like GPS, Wifi or GPS+Wifi. The use of more than one localisation system is required in a driverless vehicle like Cristal or Cycab, because no system is efficient enough to be used alone. Indeed, a localisation based on the GPS data cannot be used in certain contexts, and the localisation component must respond even if no satellite data can be captured. These requirements allow the vehicle to set a real trustworthy level and to improve the confidence in the reliability.

The composite localisation component includes several positioning systems, a controller, and a merger. Figure 13 gives a very abstract view of the composite localisation component we have been developing using the Fractal component model [BCL$^+$06]. For building the behavioural model, we follow a two-fold approach proposed in [BABC$^+$09] for Fractal, GCM and ProActive distributed components: (1) the architecture and hierarchy information are extracted from the ADL and (2) each of the primitive component's functional behaviour is specified by the user in an automata-based language.

Each positioning system is composed of an atomic positioning component and a software component to validate perceived data. The validation components transfer the positioning data to the merger if they are precise enough. The merger applies a particular algorithm to merge data obtained from positioning systems. The goal of this algorithm is to ensure that the level of reliability must not decrease between two localisations unless the operation to update the context is called. Finally, the controller's purpose is to request and to acknowledge the receipt of positioning data. In addition to mentioned requirements, other non functional requirements such as environment context, time-constrained response, cost of used networks, privacy, etc. must be taken into account when specifying and implementing a localisation component.

[4] `http://tacos.loria.fr`.

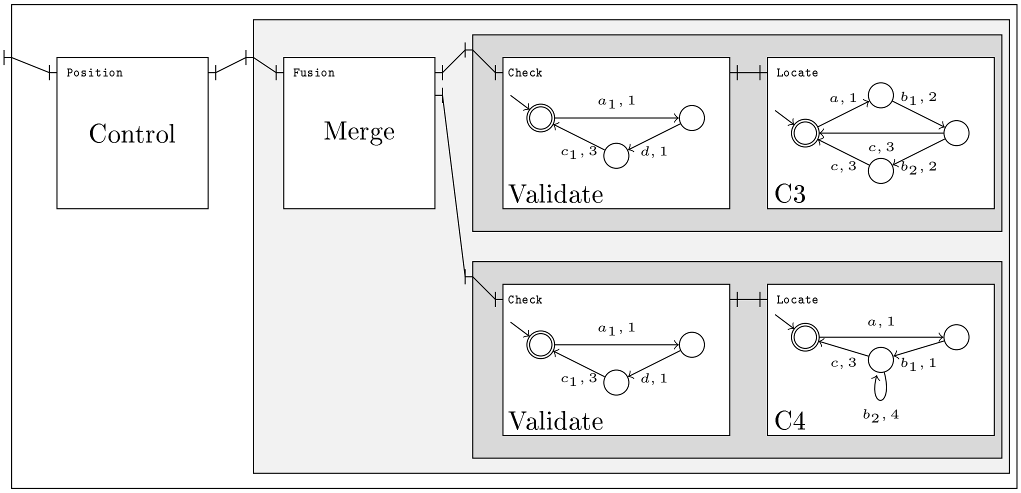

Figure 13: Sequential and parallel composition

In spite of their simplification on the QoS measures, sequential and parallel composition operators managing both functional and non functional aspects can assemble the above mentioned components.

It is easy to see how important substitutivity, sequential and parallel compositions are, especially given the need to bring costs into the analysis. Moreover, within the Fractal framework, [NBL09, PMSD07] proposed dynamic reconfiguration strategies to optimise the used memory space. It is done thanks to an implementation, called Think[5], especially dedicated to embedded Fractal components. That implementation continues with the separation of concerns principle to ease portability, reusability, and code optimisation while deploying components. Moreover, Think proposes components for services frequently used in embedded systems.

In addition, the proposed framework seems to be well-adapted to handle energy dispersion associated with actions, that is particularly relevant for sensor networks (see for instance [WCdL07]).

8. Conclusion

In this paper we proposed to manage both functional and non functional aspects of components. To sum up, this paper exposes how integer weighted automata can be used to address substitutivity issues in the context of component-based applications. We defined four kinds of substitutivity managing QoS aspects. Several complexity results for these substitutivity problems were provided. They are summed up in the following table. Provided proofs being constructive, above complexity results are tractable in practice.

[5] http://think.ow2.org/

	deterministic automata	finitely ambiguous automata	non-deterministic automata
substitutivity	Polynomial time	Polynomial time	open
partial substitutivity	Polynomial time	Polynomial time	Polynomial time
strong substitutivity	Polynomial time	Polynomial time	P-complete
partial strong substitutivity	Polynomial time	Polynomial time	P-complete

In addition, the substitutivity notions were considered in the composition context. Three natural composition operators – sequential, strict-sequential and parallel compositions – were defined. For these composition operators, new – positive and negative – results on the substitutivity vs. composition compatibility were provided. We demonstrated that considering path costs when verifying simulation relations in a composition manner, has a cost. To sum up, the composition results are given in the chart below.

	sequential	sequential disj. alphabets	strict	parallel	parallel disj. alphabets
substitutivity	yes	yes	yes	yes	yes
partial substitutivity	yes	yes	yes	yes	yes
strong substitutivity	no	yes	yes	no	yes
partial strong substitutivity	no	yes	yes	no	yes

We are well aware that there are other possibilities for defining compositions. Nevertheless, our definitions are general enough, so the present paper can be seen as a step towards more sophisticated settings to be of use in real-life applications (see, e.g., [MR08, ET07, BABC$^+$09]).

In this paper, there is no distinction between inputs, outputs and internal actions because we want the substitutivity to deal with all kinds of actions. In our approach, distinguishing actions will just lead to divide the alphabet into three parts: results will be exactly the same.

Distinguishing actions would be useful either for composition purposes or for simulation definitions. On the one hand, the parallel composition defined in Section 6 can manage different kinds of actions that can be synchronised or not. In this context, it is possible to manage synchronisation on external actions. On the other hand, using internal actions may lead to several simulation relation definitions. At this step, our work does not handle tau-based simulation. We plan to investigate this direction in a future work. Several information on that was pointed out at the beginning of Section 7.1. Notice that synchronisations for the parallel composition operator we consider can manage tau-actions as another action.

In the paper, we consider that automata represent compositions already built up from components/services. This approach seems to be not contradictory with works in [LMW07, SW09] where the behaviour semantics of a set of open nets (uncoloured Petri nets with interfaces modelling services) is given by automata whose states are annotated by boolean formula over states. In those automata, interactions/communications are already performed. Once the interactions/synchronisations are hidden in composition automata, the only remained piece of information we are interested in concerns action costs.

Note that the answer to the substitutivity problem proposed in this paper depends on the chosen abstraction. In fact, the results obtained in our framework, as well as for all abstraction-based approaches, depend on the expressive power of the formalism and on the quality of the model. It would be interesting to address the same problem with finer abstractions. In the future, following works on automata-based analyses of services [tBBG07] and components [BABC$^+$09], we plan to extend the model to include messages among components. To go further, more expressive formalisms like Mealy machines, process algebra or Petri nets would provide more precise component abstractions. In this context, extending substitutivity definitions to these formalisms is easy, but algorithmic studies have to be performed again: however substitutivity problems may be undecidable or have an intractable complexity for these formalisms. In other respects, the matter of whether the substitutivity problem is decidable in the general case, remains open. In the context of the trace-based substitutivity, this problem is undecidable. We conjecture the same result holds for the simulation-based substitutivity.

Polynomial time decidability shows the substitution notion presented in the paper is reasonable and practical. For example, it would be possible to take into consideration the fact that performance/reliability metrics of a component service are not only a function on the service or the service trace, but also on parameters such as the execution environment, the performance/reliability of externally called services, and the usage profile. In fact, the decidability being polynomial time, it could be possible to apply the algorithms for each of these parameters.

In a more general context, modelling quantitative aspects is of great interest for modelling and verifying component-based applications. Work continues on modelling and verifying properties simpler than substitutivity, and on considering other applications, e.g. business protocols.

We would like to thank the anonymous referees and FACS'08 participants for their interesting and helpful comments and suggestions to improve and to develop the preliminary results in [HKV08]. We are grateful to the anonymous SCP referees for their corrections, interesting comments and helpful advice to improve the preliminary version of this paper.

References

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

[BABC$^+$09] T. Barros, R. Ameur-Boulifa, A. Cansado, L. Henrio, and E. Madelaine. Behavioural models for distributed Fractal components. *Annales des Télécommunications*, 64(1-2):25–43, 2009.

[BBBR07] P. Bouyer, Th. Brihaye, V. Bruyère, and J.-F. Raskin. On the optimal reachability problem of weighted timed automata. *Formal Methods in System Design*, 31(2):135–175, 2007.

[BBK$^+$78] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. J. MacLeod, and M. J. Merritt. *Characteristics of Software Quality.* North-Holland Publishing Company, 1978.

[BCH05] D. Beyer, A. Chakrabarti, and Th. A. Henzinger. Web service interfaces. In A. Ellis and T. Hagino, editors, *WWW*, pages 148–159. ACM, 2005.

[BCHS07] D. Beyer, A. Chakrabarti, Th. A. Henzinger, and S. A. Seshia. An application of web-service interfaces. In *ICWS*, pages 831–838. IEEE Computer Society, 2007.

[BCL$^+$06] E. Bruneton, Th. Coupaye, M. Leclercq, V. Quéma, and J.-B. Stefani. The fractal component model and its support in Java. *Softw., Pract. Exper.*, 36(11-12):1257–1284, 2006.

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

[BJ06] Thomas Ball and Robert B. Jones, editors. *Antichains: A New Algorithm for Checking Universality of Finite Automata*, volume 4144 of *Lecture Notes in Computer Science*. Springer, 2006.

[BK03] P. Buchholz and P. Kemper. Weak bisimulation for (max/+) automata and related models. *Journal of Automata, Languages and Combinatorics*, 8(2):187–218, 2003.

[BKR07] Steffen Becker, Heiko Koziolek, and Ralf Reussner. Model-based performance prediction with the palladio component model. In Vittorio Cortellessa, Sebastián Uchitel, and Daniel Yankelevich, editors, *WOSP*, pages 54–65. ACM, 2007.

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

[Bra03] Premysl Brada. *Specification-Based Component Substitutability and Revision Identification*. PhD thesis, Charles University in Prague, 2003.

[BRL07] F. Baligand, N. Rivierre, and Th. Ledoux. A declarative approach for QoS-aware Web service compositions. In *ICSOC*, pages 422–428, 2007.

[BV06] Premysl Brada and Lukas Valenta. Practical verification of component substitutability using subtype relation. In *EUROMICRO-SEAA*, pages 38–45. IEEE, 2006.

[CCH$^+$07] S. Chouali, S. Colin, A. Hammad, O. Kouchnarenko, A. Lanoix, H. Mountassir, and J. Souquières. Requirements for the description of a component in order to use in a component based approach – Livrable TACOS L2-1.0. 23 pages Available at http://tacos.loria.fr ANR-06-SETI-017 (TACOS), 2007.

[CCSS08] Sagar Chaki, Edmund M. Clarke, Natasha Sharygina, and Nishant Sinha. Verification of evolving software via component substitutability analysis. *Formal Methods in System Design*, 32(3):235–266, 2008.

[CHS06] S. Chouali, M. Heisel, and J. Souquières. Proving component interoperability with B refinement. *Electr. Notes Theor. Comput. Sci.*, 160:157–172, 2006.

[CVZ07] I. Cerná, P. Vareková, and B. Zimmerova. Component substitutability via equivalencies of component-interaction automata. *Electr. Notes Theor. Comput. Sci.*, 182:39–55, 2007.

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

[DvdA04] J. Dehnert and W. M. P. van der Aalst. Bridging the gap between business models and workflow specifications. *Int. J. Cooperative Inf. Syst.*, 13(3):289–332, 2004.

[EFPC04] E. Ermel, A. Fladenmuller, G. Pujolle, and A. Cotton. Improved position estimation in Wireless heterogeneous networks. In *NETWORKING 2004*, May 2004.

[ET07] N. Evans and H. Treharne. Interactive tool support for csp || b consistency checking. *Formal Asp. Comput.*, 19(3):277–302, 2007.

[FEHC02] Alexandre V. Fioukov, Evgeni M. Eskenazi, Dieter K. Hammer, and Michel R. V. Chaudron. Evaluation of static properties for component-based architectures. In *EUROMICRO*, pages 33–39. IEEE Computer Society, 2002.

[FM07] Juliana Küster Filipe and Sotiris Moschoyiannis. Concurrent logic and automata combined: A semantics for components. *Electr. Notes Theor. Comput. Sci.*, 175(2):135–151, 2007.

[FUMK07] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Ws-Engineer: A model-based approach to engineering web service compositions and choreography. In *Test and Analysis of Web Services*, pages 87–119. 2007.

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

[GIRS08] C. E. Gerede, O. H. Ibarra, B. Ravikumar, and J. Su. Minimum-cost delegation in service composition. *Theoretical Computer Science*, 409(3):417–431, 2008.

[Gli07] M. Glinz. On Non-Functional Requirements. In IEEE, editor, *15th IEEE International Requirements Engineering Conference*, pages 21 – 26, 2007.

[GMRS07] Vincenzo Grassi, Raffaela Mirandola, Enrico Randazzo, and Antonino Sabetta. Klaper: An intermediate language for model-driven predictive analysis of performance and reliability. In Andreas Rausch, Ralf Reussner, Raffaela Mirandola, and Frantisek Plasil, editors, *CoCoME*, volume 5153 of *Lecture Notes in Computer Science*, pages 327–356. Springer, 2007.

[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, Paris, France*, pages 333–338. IEEE Computer Society, 2007.

[HKV08] P.-C. Héam, O. Kouchnarenko, and J. Voinot. Component simulation-based substitutivity managing QoS aspects. Technical report, University of Málaga, 2008. Int. Workshop on Formal Aspects of Software Components, FACS'08.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[HNS03] J. Hallberg, M. Nilsson, and K. Synnes. Positioning with Bluetooth. In *10th Int. Conference on Telecommunications (ICT'2003)*, 2003.

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

[Jur05] M. B. Juric. *A Hands-on Introduction to BPEL, Part 2: Advanced BPEL*, 2005. `http://www.oracle.com/technology/pub/articles/matjaz_bpel2.html`.

[KLMP04] I. Klimann, S. Lombardy, J. Mairesse, and Ch. Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theoretical Computer Science*, 327(3):349–373, 2004.

[KMT04] F. Katritzke, W. Merzenich, and M. Thomas. Enhancements of partitioning techniques for image compression using weighted finite automata. *Theoretical Computer Science*, 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.

[LKD$^+$03] H. Ludwig, A. Keller, A. Dan, R.P. King, and R. Franck. *Web Service Level Agreement (WSLA) Language Specification, Version 1.0.* IBM Corporation, January 2003.

[LMW07] N. Lohmann, P. Massuthe, and K. Wolf. Operating guidelines for finite-state services. In J. Kleijn and A. Yakovlev, editors, *ICATPN*, volume 4546 of *Lecture Notes in Computer Science*, pages 321–341. Springer, 2007.

[LVOS09] N. Lohmann, H.M.W. Verbeek, C. Ouyang, and Ch. Stahl. Comparing and evaluating Petri net semantics for BPEL. *Int. J. Business Process Integration and Management*, 4(1):60–73, 2009.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer Verlag, 1980.

[MPR02] M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[MPR05] M. Mohri, F. Pereira, and M. Riley. Weighted automata in text and speech processing. volume abs/cs/0503077, 2005.

[MR08] R. Mateescu and S. Rampacek. Formal modeling and discrete-time analysis of BPEL Web services. In J. L. G. Dietz, A. Albani, and J. Barjis, editors, *CIAO! / EOMAS*, volume 10 of *Lecture Notes in Business Information Processing*, pages 179–193. Springer, 2008.

[MRW77] J. A. McCall, P. K. Richards, and G. F. Walters. Factors in software quality (volume-III). In *Preliminary Handbook on Software Quality for an Acquisition Manager*. 1977.

[MSK05] Sotiris Moschoyiannis, Michael W. Shields, and Paul J. Krause. Modelling component behaviour with concurrent automata. *Electr. Notes Theor. Comput. Sci.*, 141(3):199–220, 2005.

[NBL09] J-F. Navas, J-P. Babau, and O. Lobry. Minimal yet effective reconfiguration infrastructures in component-based embedded systems. In *SINTER'09*, pages 41–48. ACM, 2009.

[OG00] R.J. Orr and G.D.Abowd. The smart floor: A mechanism for natural user identification and tracking. In *Conference on Human Factors in Computing Systems (CHI2000)*, The Netherlands, 2000.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Lecture Notes in Computer Science*, volume 104, pages 167–183. Springer Verlag, 1981.

[PMSD07] J. Polakovic, S. Mazare, J.-B. Stefani, and P.-Ch. David. Experience with safe dynamic reconfigurations in component-based embedded systems. In *CBSE*, pages 242–257, 2007.

[Reu03] P. H. Reussner. Automatic component protocol adaptation with the coconut/j tool suite. *Future Gener. Comput. Syst.*, 19(5):627–639, 2003.

[RHH05] R. Reussner, J. Happe, and A. Habel. Modelling parametric contracts and the state space of composite components by graph grammars. In *Fundamental Approaches to Software Engineering, 8th International Conference, FASE 2005, Edinburgh, UK*, volume 3442 of *Lecture Notes in Computer Science*, pages 80–95. Springer, 2005.

[RMG05] J.A. Royo, E. Mena, and L.C. Gallego. Locating users to develop location-based services in Wireless local area networks. In Thomson, editor, *Symp. on Ubiquitous Computing and Ambient Intelligence*, pages 471–478, Granada (Spain), September 2005.

[SCHS07] H. W. Schmidt, I. Crnkovic, G. T. Heineman, and J. A. Stafford, editors. *Component-Based Software Engineering, 10th International Symposium, CBSE 2007, Medford, MA, USA, July 9-11, 2007, Proceedings*, volume 4608 of *LNCS*. Springer, 2007.

[SE06] M. Scuturici and D. Ejigu. Positioning support in pervasive environments. In *IEEE Int. Conf. on Pervasive Services (ICPS)*, June 2006.

[SJ01] Z. Sawa and P. Jancar. P-hardness of equivalence testing on finite-state processes. In *SOFSEM*, pages 326–335, 2001.

[SJ05] Z. Sawa and P. Jancar. Behavioural equivalences on finite-state systems are ptime-hard. *Computers and Artificial Intelligence*, 24(5), 2005.

[SW09] Ch. Stahl and K. Wolf. Deciding service composition and substitutability using extended operating guidelines. *Data Knowl. Eng.*, 68(9):819–833, September 2009.

[tBBG07] M.H. ter Beek, A. Bucchiarone, and S. Gnesi. Formal methods for service composition. *Annals of Mathematics, Computing & Teleinformatics*, 5(1):1–10, 2007.

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

[vdA98] W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[vG01] R.J. van Glabbeek. *The linear time - branching time spectrum, Handbook of Process Algebra*, chapter 1. Elsevier, 2001.

[WCdL07] Xiaoling Wu, Jinsung Cho, Brian J. d'Auriol, and Sungyoung Lee. Energy-aware routing for wireless sensor networks by AHP. In *Software Technologies for Embedded and Ubiquitous Systems*, volume 4761 of *LNCS*, pages 446–455, 2007.

[Web94] A. Weber. Finite-valued Distance Automata. *Theoretical Computer Science*, 134, 1994.

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