Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 260 (2010) 109–123
www.elsevier.com/locate/entcs

Component Simulation-based Substitutivity Managing QoS Aspects[1]

Pierre-Cyrille Héam[2] Olga Kouchnarenko[2] Jérôme Voinot [2]

INRIA-CASSIS
LIFC
University of Franche-Comté
16 route de Gray, 25030 Besançon Cedex, France

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 max/plus 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 max/plus automata are investigated.

Keywords: Substitutivity, Component, Simulation, max/plus automata, Quality of Service

1 Introduction

This paper is dedicated to the verification of substitutivity of components modelled by max/plus 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.

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) – a relatively new initiative advocated by users of Java technology. 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. When is it possible to accept or reject the substi-

[1] This work is partially funded by the French project ARA COPS.

[2] {heampc,kouchna,voinot}@lifc.univ-fcomte.fr

doi:10.1016/j.entcs.2009.12.034

tution 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 or 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.

There is a lot of well-known component models and frameworks for developing service-oriented applications. Practical aspects of dynamic service selection and its implementation are presented e.g. in [4] reporting on the ANSO project bringing together major European industrial actors. Formal aspects of Web service compositions have been studied when using guarded automata in [11], or finite state machines in [2], or a finite state process algebra [12]. However, to our knowledge, for verifying Web services properties like substitutivity, none of these models allows taking service costs into account. To make up for this lack, this paper proposes and uses a formal model, called max/plus automata. This paper gives formal definitions of four – (partial) substitutivity and (partial) strong substitutivity – problems based on a simulation of automata taking path costs into account. New decision/complexity results for different classes of max/plus automata are then presented for these substitutivity problems.

Related works. Weighted automata – an extension of max/plus automata – is a formalism widely used in computer science for applications in images compression [19,22], speech-to-text processing [26,3] or discrete event systems [13]. These large application areas make them intensively studied from the theoretical point of view [23,33,14,21]. See [6] for more detail on max/plus automata.

There are numerous works dealing with component substitutivity or interoperability [28,9,8]. Our work is close to that in [9], where the authors addressed component substitutability using equivalences between component-interaction automata, which are defined 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 max/plus automata.

Different solutions have been proposed to allow taking QoS into account while specifying Web services and their compositions [24,31,10,7]. These approaches are promising, but they are currently W3C submissions, not W3C recommendations [3]. That is why following [15], we proposed in [16] to extend both BPEL and WSDL specifications with a notion of service costs for being closer to the Web services reality. In [15] the substitutivity problem has been investigated for the trace equivalence and an automatic approach to convert Web services description files into max/plus automata has been provided.

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 [25] and Park [27] is widely used in computer science because of its numerous advantages: It preserves branching behaviours and,

[3] A good introductory reference to the Web services standard is [17].

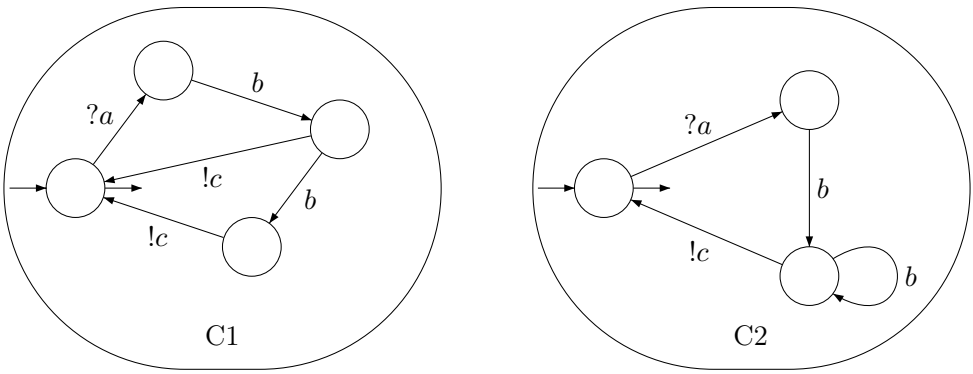
consequently, most of dynamic properties; there is a link between the strong bisimulation and modal logics [18]; this is a congruence for a number of composition operators, e.g. parallel composition, prefixing by an action, etc.

Bisimulation relations over max/plus automata were investigated in [5]. 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 [5], weights are related to successful paths of max/plus automata.

Layout of the paper. The remainder of the paper is organised as follows. A motivating example is given in Sect. 2. Section 3 recalls max/plus automata and defines four simulation-oriented substitutivity notions based on them. The verification issues on components substitutivity are presented in Sect. 4 and 5 before concluding in Sect. 6.

2 Motivating Example

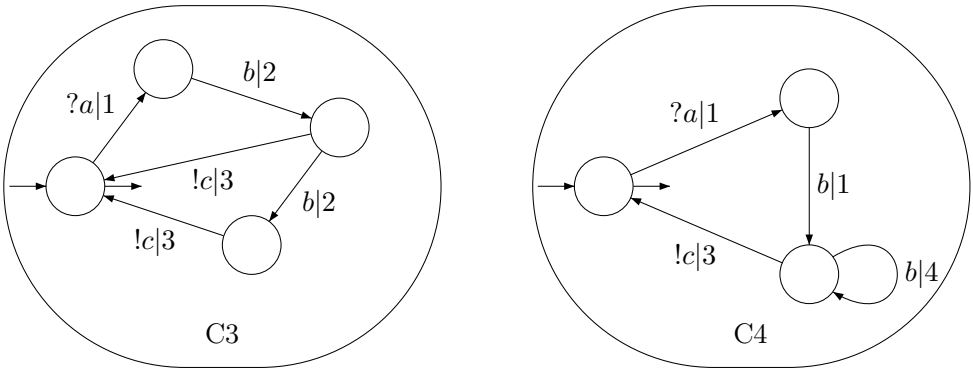
We illustrate the substitutivity problem by a characteristic toy example. Let's consider the two following components, represented by finite automata where, to be intuitive [4], communicating actions are labelled with ? or !.



- Component C1 works as follows. Action $?a$ encodes that C1 receives a request; at this stage, C1 performs action b once or twice (depending on the abstracted value passed through the $?a$ request). Then C1 sends a message $!c$ to acknowledge that its task is successfully performed.
- Component C2 works similarly but can perform action b as many times as it is required.

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

[4] In the rest of the paper, the actions are not partitioned into input/output/internal actions.



For both C3 and C4, receiving a request $?a$ costs 1 energy unit and sending the confirmation $!c$ costs 3 energy units. However, for C3 each action b costs 2 energy units. For C4, performing the first b action costs only 1 energy unit but other b actions cost 4 energy units. The intuition behind this modelling is as follows. C3 has a low-cache memory allowing it to locally compute action b twice. C4 has a high performance low-cache memory that allows it to locally compute action b with a cost of 1 energy unit but only once. C4 also has a local hard drive that makes more b 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 $?abb!c$ on C3 has the cost of 8 energy units whereas the same sequence of actions costs 9 energy units on C4.

3 Theoretical Background

In this paper, Σ denotes a finite set of actions. We first introduce the notion of max/plus automata.

Definition 3.1 A *finite max/plus 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.

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 max/plus automaton and related notions.

A *partial execution* or a *path* of a finite max/plus 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* tr 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: $\text{cost}_{\mathcal{A}}(\pi) = \sum_{i=0}^{n} c_i$. A state p of a max/plus 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). 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. 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.

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 $\mathbb{Q}$,

- 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
- 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.

In the rest of the paper, the actions are not partitioned into input/output/internal actions (see [15,16] for a conversion of Web services description files into max/plus automata). Consequently, the communications are not covered by τ-transitions. The only τ-transitions appear when translating the assign and empty BPEL activities into max/plus automata. Those transitions can be abstracted without loosing behaviours since they do not take part in services exchanges.

In this setting, i.e. without silent τ-transitions, the $\preceq$-simulation relation is compatible with a sequential composition operator modeling e.g. the sequence BPEL structured activities, and with an asynchronous parallel composition operator implementing e.g. the flow BPEL structured activities. Both operators are important in practice since they allow building complex services by a composition of services.

Definition 3.2 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 automata. The sequential composition of $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted $\mathcal{A}_1.\mathcal{A}_2$, is an automaton $\mathcal{A}_{12} = (Q_{12}, A_{12}, E_{12}, I_{12}, F_{12})$ where

- $Q_{12} = \{p_1; p_2 \mid \forall p_1 \in Q_1, p_2 \in Q_2\} \cup Q_2$,
- $A_{12} = A_1 \cup A_2$,

- $I_{12} = \{p_1; p_2 \mid \forall p_1 \in I_1, p_2 \in I_2\}$,
- $F_{12} = F_2$,

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

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

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. It is easy to establish the following compatibility result whose proof is rather basic for strong simulation relations handling deadlocks.

Proposition 3.3 *Let $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ be three automata, let $\mathcal{A}_1 \preceq \mathcal{A}_3$. One has:*

(i) $\mathcal{A}_1.\mathcal{A}_2 \preceq \mathcal{A}_3.\mathcal{A}_2$

(ii) $\mathcal{A}_2.\mathcal{A}_1 \preceq \mathcal{A}_2.\mathcal{A}_3$

Definition 3.4 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 automata. The parallel composition of $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted $\mathcal{A}_1 \| \mathcal{A}_2$, is an automaton $\mathcal{A}_{12} = (Q_{12}, A_{12}, E_{12}, I_{12}, F_{12})$ where

- $Q_{12} = \{p_1 \| p_2 \mid \forall p_1 \in Q_1, p_2 \in Q_2\} \cup \{p_{12}^i \mid p_{12}^i \notin (Q_1 \cup Q_2)\} \cup \{p_{12}^f \mid p_{12}^f \notin (Q_1 \cup Q_2)\}$,
- $A_{12} = A_1 \cup A_2 \cup \{\epsilon\}$,
- $I_{12} = \{p_{12}^i \mid \exists p_{12}^i \in Q_{12} \setminus (Q_1 \cup Q_2)\}$,
- $F_{12} = \{p_{12}^f \mid \exists p_{12}^f \in Q_{12} \setminus (Q_1 \cup Q_2)\}$,

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

$$[\text{START}] \quad p_{12}^i \xrightarrow{\epsilon,0}_{\mathcal{A}_1\|\mathcal{A}_2} p_1 \| p_2 \; p_k \in I_k \qquad [\text{PAR1}] \quad \frac{p_1 \xrightarrow{a_1,c_1}_{\mathcal{A}_1} q_1}{p_1 \| p_2 \xrightarrow{a_1,c_1}_{\mathcal{A}_1\|\mathcal{A}_2} q_1 \| p_2}$$

$$[\text{PAR2}] \quad \frac{p_2 \xrightarrow{a_2,c_2}_{\mathcal{A}_2} q_2}{p_1 \| p_2 \xrightarrow{a_2,c_2}_{\mathcal{A}_1\|\mathcal{A}_2} p_1 \| q_2} \qquad [\text{END}] \quad q_1 \| q_2 \xrightarrow{\epsilon,0}_{\mathcal{A}_1\|\mathcal{A}_2} p_{12}^f \; q_k \in F_k$$

This definition means that all moves of parallel composition are moves of either $\mathcal{A}_1$ or of $\mathcal{A}_2$, or an initial/ending move of both $\mathcal{A}_1$ and $\mathcal{A}_2$. The following proposition claims that the $\preceq$-simulation is compatible with the parallel composition above.

Proposition 3.5 *Let $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ be four automata, such that $\mathcal{A}_1 \preceq \mathcal{A}_3$ and $\mathcal{A}_2 \preceq \mathcal{A}_4$. We have:*

(i) $\mathcal{A}_1 \| \mathcal{A}_2 \preceq \mathcal{A}_3 \| \mathcal{A}_4$

(ii) $\mathcal{A}_2 \| \mathcal{A}_1 \preceq \mathcal{A}_4 \| \mathcal{A}_3$

3.1 Modelling Substitutivity

A problem occurring while managing Web services is to determine that a Web service performs the same tasks as another possibly failing service, with comparable or higher quality. More formally, for two Web services modelled by their max/plus 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. 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.

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 $\text{cost}_{\mathcal{A}_2}(\pi_2) \leq \text{cost}_{\mathcal{A}_1}(\pi_1)$, false otherwise.

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 $\pi_1 \preceq \pi_2'$, $\text{cost}_{\mathcal{A}_2}(\pi_2') \leq \text{cost}_{\mathcal{A}_1}(\pi_1)$, false otherwise.

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 $\text{cost}_{\mathcal{A}_2}(\pi_2) \leq \text{cost}_{\mathcal{A}_1}(\pi_1)$, false otherwise.

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.

Notice that in the above definitions we choose that $\text{cost}(\pi_2) \leq \text{cost}(\pi_1)$ modeling 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 max/plus automata.

Theorem 3.6 *Given two max/plus automata $\mathcal{A}_1$ and $\mathcal{A}_2$, it is*

- *undecidable to test whether for every $u \in L(\mathcal{A}_1)$, $\text{cost}_{\mathcal{A}_1}(u) \leq \text{cost}_{\mathcal{A}_2}(u)$ [23]; the same problem is decidable if $\mathcal{A}_1$ and $\mathcal{A}_2$ are both finitely ambiguous [14,33],*
- *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. $\text{cost}_{\mathcal{A}_1}(\pi) \leq 0$) [23],*
- *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 [14,33],*
- *decidable in polynomial time to test whether $\mathcal{A}_1$ is finitely ambiguous [34].*
- *PSPACE-complete to decide whether $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$ [1].*

4 Strong Substitutivity Problems

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

Lemma 4.1 *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 4.2 *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 $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the strong substitutivity problem if and only if $\mathcal{A}_1 \preceq A_2$ and for every successful path π of $\mathcal{B}$, $\mathrm{cost}_{\mathcal{B}}(\pi) \geq 0$.

- Assume that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the strong substitutivity problem. By Lemma 4.1, 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 $\mathcal{B}$,

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

  By definition of $\mathcal{B}$, there exist $p_0, p_1, \dots, p_n$ states of $\mathcal{A}_1$, $q_0, q_1, \dots, q_n$ states of $\mathcal{A}_2$, integers $c_1, c_2, \dots, c_n, d_1, d_2, \dots, d_n$ such that
  · $\pi_1 = (p_0, a_1, c_1, p_1), (p_1, a_2, c_2, p_2), \dots, (p_{n-1}, a_n, c_n, p_n)$ is a successful path in $\mathcal{A}_1$,
  · $\pi_2 = (q_0, a_1, d_1, q_1), (q_1, a_2, d_2, q_2), \dots, (q_{n-1}, a_n, d_n, q_n)$ 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:

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

$$\text{Consequently,} \quad \sum_{i=1}^{n} \alpha_i \geq 0.$$

- 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 4.1, for every successful path in $\mathcal{A}_1$ there exists a $\preceq$-related successful path in $\mathcal{A}_2$.
  Finally, consider two successful paths

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

  in $\mathcal{A}_1$ and

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

  in $\mathcal{A}_2$ such that $\pi_1 \preceq \pi_2$.
  By definition there exists an successful path π in $\mathcal{B}$,

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

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 $\mathrm{cost}(\pi) \geq 0$:

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

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

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 [29,30]. 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 4.3 *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 4.2. We claim that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the partial strong substitutivity problem if and only if $\mathcal{A}_1 \preceq A_2$ and if every transition of $\mathcal{B}$ has a positive weight.

The proof is quite similar to the one of Theorem 4.2: 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 (partial) substitutivity problem.

Theorem 5.1 *The substitutivity problem is polynomial time decidable if $\mathcal{A}_2$ is finitely ambiguous.*

Proof Let $\mathcal{A}_1 = (Q_1, \Sigma, E_1, I_1, F_1)$ be a max/plus automaton and $\mathcal{A}_2 = (Q_2, \Sigma, E_2, I_2, F_2)$ a finitely ambiguous max/plus 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 $a_1a_2\dots a_n$, 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 $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the substitutivity problem 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 $\mathrm{cost}_{\mathcal{B}}(\pi) \geq 0$.

($\Rightarrow$) Assume first that $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the substitutivity problem. 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 $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the substitutivity problem, there exists an execution π_2 in $\mathcal{A}_2$ of label $a_1a_2\dots a_n$ such that

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

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 $\mathrm{cost}_{\mathcal{A}_2}(\pi_2) = \mathrm{cost}_{\mathcal{A}_4}(\pi_4)$ and $\mathrm{cost}_{\mathcal{A}_1}(\pi_1) = \mathrm{cost}_{\mathcal{A}_3}(\pi_3)$ 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 $\mathrm{cost}_{\mathcal{B}}(\pi) \geq 0$.

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 $\mathcal{A}_1 \preceq A_2$, 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:

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

Set

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

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

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

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 3.6. □

Theorem 5.2 *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 $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the partial substitutivity problem 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 $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the partial substitutivity problem 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 $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the partial substitutivity problem.

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 Conclusion

In this paper we proposed to manage both functional and non functional aspects of components. To sum up, this paper exposes how max/plus 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 were provided.

On the implementation side, following our work in [15,16], an automatic translation of Web services description files into max/plus automata has been implemented, as well as an algorithm for the trace-based substitutivity problem. The prototype has been tested on a Dell Latitude D600 with an Intel Pentium M 1.4Ghz and 512Mo of RAM. The tests have been performed on different versions of a movie store example, a book store example provided by Oracle [20], and the classical loan approval example. For the book store example specified in BPEL/WSDL, the automaton is built in less than 3 seconds, and it has 67 states and 101 transitions due to the flow activities putting in parallel three activities. We intend to continue the implementation and extend that work to simulation-based substitutivity problems presented in this paper. It would necessitate considering, e.g., correlation sets which are currently not supported in the translator.

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. In other respects, the problem 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 include 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 that 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. In addition, the proposed framework seems to be well-adapted to handle energy dispersion associated with actions, that is particularly relevant for embedded systems or sensor networks (see for instance [32]).

Acknowledgement

We would like to thank the anonymous referees for their interesting and helpful comments and suggestions to improve this work.

References

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

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

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

[4] A. Bottaro and R. S. Hall. Dynamic contextual service ranking. In *6th International Symposium on Software Composition (SC 2007), Braga, Portugal*, pages 129–143, 2007.

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

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

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

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

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

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

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

[12] 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.

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

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

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

[16] P.-C. Héam, O. Kouchnarenko, and J. Voinot. Towards formalizing QoS of web services with weighted automata. Technical Report 6218, INRIA, 2007.

[17] T. Hauser and U. Löwer. *Web Services. Die Standards*. Galileo Computing, November 2003.

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

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

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

[21] 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.

[22] 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.

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

[24] 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.

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

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

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

[28] 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 *Lecture Notes in Computer Science*. Springer, 2007.

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

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

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

[32] 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.

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

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