Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 229 (2009) 3–18

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Algorithms and Complexity of Automata Synthesis by Asynhcronous Orchestration With Applications to Web Services Composition

Philippe Balbiani[1] Fahima Cheikh[2] Guillaume Feuillade[3]

Universit de Toulouse
CNRS, Institut de recherche en informatique de Toulouse
118 route de Narbonne, 31062 Toulouse CEDEX 9 (France)

Abstract

Composition of services is necessary for realizing complex tasks on the Web. It has been characterized either as a plan synthesis problem or as a software synthesis problem: given a goal and a set of Web services, generate a composition of the Web services that satisfies the goal. We propose algorithms for performing automated Web service composition. We also examine the composition of services from the perspective of computational complexity.

Keywords: Service composition, controller synthesis, computational complexity.

1 Introduction

The development of service oriented architectures for implementing distributed software systems demands that organizations make their abilities accessible via the Internet through Web service interfaces. The web services are published using Web service standards like WSDL [3] or the abstract WS-BPEL [2,15]. In most cases, Web services are nothing more than elementary components in a client-server architecture. Their importance lies in the fact that we can compose them to create complex business processes, using Web service standards like concrete WS-BPEL [2] and WS-CDL [14,23]. The WS-CDL is used to specify the choreography between

[1] Email: Philippe.Balbiani@irit.fr
[2] Email: Fahima.Cheikh@irit.fr
[3] Email: Guillaume.Feuillade@irit.fr

doi:10.1016/j.entcs.2009.06.036

services and concrete WS-BPEL is used for the orchestration of services. Composition of Web services involves multifarious difficulties and requires to formally define the semantics of the input services. If one tries to compose complex business processes from given input services then plan synthesis algorithms from artificial intelligence or software synthesis algorithms from computer science can be employed. There exist many approaches to the composition problem [17]. Characterizing Web service composition as a plan synthesis problem forces us to devise algorithms tackling incomplete information and uncertain effects. Different automated techniques have been proposed to solve the composition/plan problem [21,22,25]. Nevertheless, their computational complexity has not been investigated in details. Characterizing Web service composition as a software synthesis problem compels us to devise algorithms working with behavioural descriptions given in terms of automata. Different automated techniques have been proposed to solve the composition/software problem [5,6,7,9,10,24]. Nevertheless, their completeness rests on syntactical restrictions that prevent them from being fully applicable.

Although services might be considered as non autonomous agents which know only about themselves, service oriented architectures and multi-agent systems share many characteristics [12]. To illustrate the truth of this, one has only to mention the fact that several researchers have recently advocate the use of Web service technology to build multi-agent systems accessible through the Web [16] or the use of multi-agent-based coalition formation approaches for Web service composition [18].

In this paper, we propose a solution for the compositon/software problem. More precisely, we propose algorithms for performing automated Web service composition. We also examine the composition of services from the perspective of computational complexity. The differences between the work presented in this paper and the works done in [7,10,24] are the following. First, we do not consider the same relation between the goal and the available services. We consider the bisimulation relation whereas the papers mentioned above consider the simulation relation. Intuitively, the bisimulation relation does not allow the available services to perform sequences of actions not performed by the goal. In practice, this is important. For example, from a security point of view, if one wants to prohibit sequences of actions that allow services to guess secret information. The second difference is that we consider internal actions and communication actions as well. More precisely, the communication actions are performed through bounded channels. We impose this constraint since otherwise the composition problem will be undecidable. In other respect, in [7], the authors consider that the goal and the available services are deterministic. This restriction, that we do not consider in our paper, is also usually considered in the theory of controller and greatly simplifies the synthesis problem. Finally, in [7,10,24], there are guards/conditions on the transitions. Nevertheless, our result still hold if we add guards/conditions on transitions.

The section-by-section breakdown of the paper is as follows. Section 2 recalls the notion of finite automata and establishes the concept of Web service. In section 3, basic definitions are given and preliminary results are proved. These definitions and these results will be used in great depth in the remaining sections. Section 4 introduces the composition problem: given a goal and a set of Web services, generate a

composition of the Web services that satisfies the goal. In section 5, we examine the composition of services from the perspective of computational complexity. Two ways of solving the composition problem are presented in section 6. In section 7, we talk about some open problems.

2 Web services as finite automata

In this section, the notion of finite automata is recalled and the concept of Web service is established.

2.1 Finite automata

Let Σ be a finite set of actions. A finite automaton over Σ is a structure $\mathcal{A} = (S, \Delta, s^{in})$ where S is a finite set of states, Δ is a function

- Δ: $S \times \Sigma \to 2^S$,

$s^{in} \in S$ is an initial state. For all $\Sigma' \subseteq \Sigma$, the relation $\to_{\mathcal{A}}^{\Sigma'} \subseteq S \times S$ describes how the finite automaton can move from one state to another in 1 step under some action in Σ'. It is defined formally as follows: $s \to_{\mathcal{A}}^{\Sigma'} t$ iff there exists $a \in \Sigma'$ such that $t \in \Delta(s, a)$. Furthermore, let $\to_{\mathcal{A}}^{\Sigma'\star}$ be the reflexive transitive closure of $\to_{\mathcal{A}}^{\Sigma'}$. For all $\Sigma' \subseteq \Sigma$, we shall say that $\mathcal{A}$ loops over Σ' iff for all $a \in \Sigma'$, $\to_{\mathcal{A}}^{\{a\}} = Id_S$.

2.2 Products

Let $\mathcal{A}_1 = (S_1, \Delta_1, s_1^{in})$ and $\mathcal{A}_2 = (S_2, \Delta_2, s_2^{in})$ be finite automata over Σ. By $\mathcal{A}_1 \otimes \mathcal{A}_2$, we denote the asynchronous product of $\mathcal{A}_1$ and $\mathcal{A}_2$, i.e. the finite automaton $\mathcal{A} = (S, \Delta, s^{in})$ over Σ such that $S = S_1 \times S_2$, Δ is the function defined by

- $(t_1, t_2) \in \Delta((s_1, s_2), a)$ iff either $t_1 \in \Delta_1(s_1, a)$ and $t_2 = s_2$ or $t_1 = s_1$ and $t_2 \in \Delta_2(s_2, a)$,

$s^{in} = (s_1^{in}, s_2^{in})$. By $\mathcal{A}_1 \times \mathcal{A}_2$, we denote the synchronous product of $\mathcal{A}_1$ and $\mathcal{A}_2$, i.e. the finite automaton $\mathcal{A} = (S, \Delta, s^{in})$ over Σ such that $S = S_1 \times S_2$, Δ is the function defined by

- $(t_1, t_2) \in \Delta((s_1, s_2), a)$ iff $t_1 \in \Delta_1(s_1, a)$ and $t_2 \in \Delta_2(s_2, a)$,

$s^{in} = (s_1^{in}, s_2^{in})$.

2.3 Bisimulations

Let $\mathcal{A}_1 = (S_1, \Delta_1, s_1^{in})$ and $\mathcal{A}_2 = (S_2, \Delta_2, s_2^{in})$ be finite automata over Σ. For all $\Sigma' \subseteq \Sigma$, a relation $Z \subseteq S_1 \times S_2$ such that $(s_1^{in}, s_2^{in}) \in Z$ is called a bisimulation between $\mathcal{A}_1$ and $\mathcal{A}_2$ modulo Σ', notation Z: $\mathcal{A}_1 \longleftrightarrow \mathcal{A}_2$ (Σ'), iff the following conditions are satisfied for all $(s_1, s_2) \in Z$ and for all $a \in \Sigma \setminus \Sigma'$:

- for all $t_1 \in S_1$, if $s_1 \to_{\mathcal{A}_1}^{\Sigma'\star} \circ \to_{\mathcal{A}_1}^{\{a\}} \circ \to_{\mathcal{A}_1}^{\Sigma'\star} t_1$ then there exists $t_2 \in S_2$ such that $s_2 \to_{\mathcal{A}_2}^{\Sigma'\star} \circ \to_{\mathcal{A}_2}^{\{a\}} \circ \to_{\mathcal{A}_2}^{\Sigma'\star} t_2$ and $(t_1, t_2) \in Z$,

- for all $t_2 \in S_2$, if $s_2 \rightarrow_{\mathcal{A}_2}^{\Sigma'\star} \circ \rightarrow_{\mathcal{A}_2}^{\{a\}} \circ \rightarrow_{\mathcal{A}_2}^{\Sigma'\star} t_2$ then there exists $t_1 \in S_1$ such that $s_1 \rightarrow_{\mathcal{A}_1}^{\Sigma'\star} \circ \rightarrow_{\mathcal{A}_1}^{\{a\}} \circ \rightarrow_{\mathcal{A}_1}^{\Sigma'\star} t_1$ and $(t_1, t_2) \in Z$.

Furthermore, for all $\Sigma' \subseteq \Sigma$, if there is a bisimulation between $\mathcal{A}_1$ and $\mathcal{A}_2$ modulo Σ' then we write $\mathcal{A}_1 \longleftrightarrow \mathcal{A}_2$ (Σ').

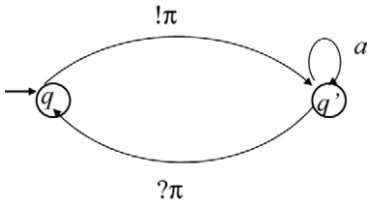

Fig. 1.

2.4 Web services

Let Π be a finite set of channels. Following the line of reasoning suggested by [5,6,9], we model Web services on finite automata with input and output. Web services communicate by sending asynchronous messages through channels. Communication through channels can be assumed to be reliable so that messages, once they are sent, do not get lost during their transmission. In this paper, for simplicity, we abstract from message contents and we consider that channels cannot contain, at all times, more than 1 message. Formally, a Web service over Π and Σ is a finite automaton over $(\{!, ?\} \times \Pi) \cup \Sigma$. For all $\pi \in \Pi$, the send action $!\pi$ consists of adding a message at channel π whereas the receive action $?\pi$ consists of taking away a message at channel π. The action $!\pi$ can be executed provided the channel is not full (i.e. π must contain exactly 0 message) whereas the action $?\pi$ can be executed provided the channel is not empty (i.e. π must contain exactly 1 message). This motivates the following definition. Let $\mathcal{A} = (S, \Delta, s^{in})$ be a finite automaton over $(\{!, ?\} \times \Pi) \cup \Sigma$. By $FA(\mathcal{A})$, we denote the finite automaton $\mathcal{A}' = (S', \Delta', s^{in'})$ over $(\{!, ?\} \times \Pi) \cup \Sigma$ of exponential size such that $S' = S \times 2^{\Pi}$, Δ' is the function defined by

- $(t, Q) \in \Delta'((s, P), !\pi)$ iff $t \in \Delta(s, !\pi)$, $Q = P \cup \{\pi\}$, $\pi \notin P$,
- $(t, Q) \in \Delta'((s, P), ?\pi)$ iff $t \in \Delta(s, ?\pi)$, $Q = P \setminus \{\pi\}$, $\pi \in P$,
- $(t, Q) \in \Delta'((s, P), a)$ iff $t \in \Delta(s, a)$, $Q = P$,

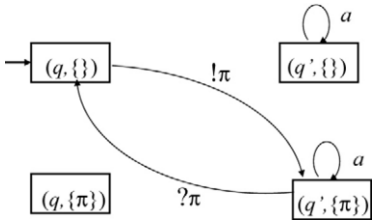

Fig. 2.

$s^{in'} = (s^{in}, \emptyset)$. Intuitively, $FA(\mathcal{A})$ is the finite automaton obtained from $\mathcal{A}$ when the status of channels is included into states. Remark that one can construct $FA(\mathcal{A})$

in exponential time. Take the case of $\mathcal{A}$, the finite automaton from figure 1. Then $FA(\mathcal{A})$ is the finite automaton from figure 2.

3 Basic definitions and preliminary results

In this section, basic definitions are given and preliminary results are proved. These definitions and these results will be used in great depth in the remaining sections.

3.1 Basic definitions

It is convenient to take a finite set Π° of channels such that $(\Sigma \cup \Pi) \cap \Pi^\circ = \emptyset$ and $Card(\Pi) = Card(\Pi^\circ)$ and to use a bijection $\pi \mapsto \pi^\circ$ from Π to Π°. By L°, we mean the finite automaton $\mathcal{A}' = (S', \Delta', s^{in'})$ over $\{!, ?\} \times \Pi^\circ$ such that $S' = \{0\}$, Δ' is the function defined by

- $\Delta'(0, !\pi^\circ) = \{0\}$ and $\Delta'(0, ?\pi^\circ) = \{0\}$,

$s^{in'} = 0$. Let $\mathcal{A} = (S, \Delta, s^{in})$ be a finite automaton over $(\{!, ?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$. By $Del^\circ(\mathcal{A})$, we denote the finite automaton $\mathcal{A}' = (S', \Delta', s^{in'})$ over $(\{!, ?\} \times \Pi) \cup \Sigma$ such that $S' = S$, Δ' is the function defined by

- $\Delta'(s, !\pi) = \Delta(s, !\pi) \cup \Delta(s, !\pi^\circ)$ and $\Delta'(s, ?\pi) = \Delta(s, ?\pi) \cup \Delta(s, ?\pi^\circ)$,
- $\Delta'(s, a) = \Delta(s, a)$,

$s^{in'} = s^{in}$. By $FA^\circ(\mathcal{A})$, we denote the finite automaton $\mathcal{A}' = (S', \Delta', s^{in'})$ over $(\{!, ?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ of exponential size such that $S' = S \times 2^\Pi$, Δ' is the function defined by

- $(t, Q) \in \Delta'((s, P), !\pi)$ iff $t \in \Delta(s, !\pi)$, $Q = P \cup \{\pi\}$, $\pi \notin P$ and $(t, Q) \in \Delta'((s, P), ?\pi)$ iff $t \in \Delta(s, ?\pi)$, $Q = P \setminus \{\pi\}$, $\pi \in P$,
- $(t, Q) \in \Delta'((s, P), !\pi^\circ)$ iff $t \in \Delta(s, !\pi^\circ)$, $Q = P \cup \{\pi\}$, $\pi \notin P$ and $(t, Q) \in \Delta'((s, P), ?\pi^\circ)$ iff $t \in \Delta(s, ?\pi^\circ)$, $Q = P \setminus \{\pi\}$, $\pi \in P$,
- $(t, Q) \in \Delta'((s, P), a)$ iff $t \in \Delta(s, a)$, $Q = P$,

$s^{in'} = (s^{in}, \emptyset)$. Remark that one can construct $FA^\circ(\mathcal{A})$ in exponential time. Let $\mathcal{A} = (S, \Delta, s^{in})$ be a finite automaton over $\{!, ?\} \times \Pi$. By $Ren^\circ(\mathcal{A})$, we denote the finite automaton $\mathcal{A}' = (S', \Delta', s^{in'})$ over $(\{!, ?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ such that $S' = S$, Δ' is the function defined by

- $\Delta'(s, !\pi) = \{s\}$ and $\Delta'(s, ?\pi) = \{s\}$,
- $\Delta'(s, !\pi^\circ) = \Delta(s, !\pi)$ and $\Delta'(s, ?\pi^\circ) = \Delta(s, ?\pi)$,
- $\Delta'(s, a) = \{s\}$,

$s^{in'} = s^{in}$. Obviously, $Ren^\circ(\mathcal{A})$ loops over $(\{!, ?\} \times \Pi) \cup \Sigma$.

3.2 Preliminary results

Now, we present some useful lemmas.

Lemma 3.1 *Let $\mathcal{A}_1 = (S_1, \Delta_1, s_1^{in})$ be a finite automaton over $(\{!,?\} \times \Pi) \cup \Sigma$ and $\mathcal{A}_2 = (S_2, \Delta_2, s_2^{in})$ be a finite automaton over $\{!,?\} \times \Pi$. Then, $FA(\mathcal{A}_1 \otimes \mathcal{A}_2)$ is isomorphic to $Del^\circ(FA^\circ(\mathcal{A}_1 \otimes L^\circ) \times Ren^\circ(\mathcal{A}_2))$.*

Proof. States in $FA(\mathcal{A}_1 \otimes \mathcal{A}_2)$ are of the form $((s_1, s_2), P)$ with $s_1 \in S_1$, $s_2 \in S_2$ and $P \subseteq \Pi$ whereas states in $Del^\circ(FA^\circ(\mathcal{A}_1 \otimes L^\circ) \times Ren^\circ(\mathcal{A}_2))$ are of the form $(((s_1, 0), P), s_2)$ with $s_1 \in S_1$, $P \subseteq \Pi$ and $s_2 \in S_2$. Obviously, the bijection $((s_1, s_2), P) \mapsto (((s_1, 0), P), s_2)$ is an isomorphism from $FA(\mathcal{A}_1 \otimes \mathcal{A}_2)$ to $Del^\circ(FA^\circ(\mathcal{A}_1 \otimes L^\circ) \times Ren^\circ(\mathcal{A}_2))$. □

Lemma 3.2 *Let $\mathcal{A}_1 = (S_1, \Delta_1, s_1^{in})$ be a finite automaton over $(\{!,?\} \times \Pi) \cup \Sigma$ and $\mathcal{A}_2 = (S_2, \Delta_2, s_2^{in})$ be a finite automaton over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!,?\} \times \Pi) \cup \Sigma$. Then, $Del^\circ(FA^\circ(\mathcal{A}_1 \otimes L^\circ) \times \mathcal{A}_2)$ is isomorphic to $FA(\mathcal{A}_1 \otimes Del^\circ(L^\circ \times \mathcal{A}_2))$.*

Proof. States in $Del^\circ(FA^\circ(\mathcal{A}_1 \otimes L^\circ) \times \mathcal{A}_2)$ are of the form $(((s_1, 0), P), s_2)$ with $s_1 \in S_1$, $P \subseteq \Pi$ and $s_2 \in S_2$ whereas states in $FA(\mathcal{A}_1 \otimes Del^\circ(L^\circ \times \mathcal{A}_2))$ are of the form $((s_1, (0, s_2)), P)$ with $s_1 \in S_1$, $s_2 \in S_2$ and $P \subseteq \Pi$. Obviously, the bijection $(((s_1, 0), P), s_2) \mapsto ((s_1, (0, s_2)), P)$ is an isomorphism from $Del^\circ(FA^\circ(\mathcal{A}_1 \otimes L^\circ) \times \mathcal{A}_2)$ to $FA(\mathcal{A}_1 \otimes Del^\circ(L^\circ \times \mathcal{A}_2))$. □

Lemma 3.3 *Let $\mathcal{A}_1 = (S_1, \Delta_1, s_1^{in})$ be a finite automaton over $(\{!,?\} \times \Pi) \cup \Sigma$ and $\Pi' \subseteq \Pi$. Then, one can construct in polynomial time a modal μ-calculus formula $f(\mathcal{A}_1, \Pi')$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ of polynomial size such that for all finite automata $\mathcal{A}_2 = (S_2, \Delta_2, s_2^{in})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$, $\mathcal{A}_1 \longleftrightarrow Del^\circ(\mathcal{A}_2)$ $(\{!,?\} \times \Pi')$ iff $\mathcal{A}_2 \models f(\mathcal{A}_1, \Pi')$.*

Proof. See [8] for details. □

By lemmas 3.1, 3.2 and 3.3, we infer immediately the following theorem.

Theorem 3.4 *Let $\mathcal{A} = (S_\mathcal{A}, \Delta_\mathcal{A}, s_\mathcal{A}^{in})$ and $\mathcal{B} = (S_\mathcal{B}, \Delta_\mathcal{B}, s_\mathcal{B}^{in})$ be finite automata over $(\{!,?\} \times \Pi) \cup \Sigma$ and $\Pi' \subseteq \Pi$. Then, the following conditions are equivalent:*

(i) *There exists a finite automaton $\mathcal{C}$ over $\{!,?\} \times \Pi$ such that $FA(\mathcal{A}) \longleftrightarrow FA(\mathcal{B} \otimes \mathcal{C})$ $(\{!,?\} \times \Pi')$.*

(ii) *There exists a finite automaton $\mathcal{C}$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!,?\} \times \Pi) \cup \Sigma$ and such that $FA(\mathcal{A}) \longleftrightarrow Del^\circ(FA^\circ(\mathcal{B} \otimes L^\circ) \times \mathcal{C})$ $(\{!,?\} \times \Pi')$.*

(iii) *There exists a finite automaton $\mathcal{C}$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!,?\} \times \Pi) \cup \Sigma$ and such that $FA^\circ(\mathcal{B} \otimes L^\circ) \times \mathcal{C} \models f(FA(\mathcal{A}), \Pi')$.*

This theorem will be used in section 6 to define decision procedures for the composition problem of Web services.

4 Composition of Web services

This section considers issues that arise when addressing the task of combining and coordinating a set of Web services. We assume the process of Web service composition to be goal oriented: given a goal and a set of Web services, generate a

composition of the Web services that satisfies the goal. According to [21,22,25], goals are conditions on the behaviour of the composition that can be expressed in the *EaGLe* language. In this approach, service composition boils down to the task of combining and coordinating the available Web services into a complex business process satisfying the given condition. According to [5,6,9], goals are finite automata with input and output, i.e. Web services as defined in section 2.4. In this approach, service composition boils down to the task of combining and coordinating the available Web services into a complex business process that can simulate the given finite automaton with input and output. In this paper, we automate composition as defined in the second approach. This brings us to the following decision problem:

- CP: given a finite set Σ of actions, a finite set Π of channels, finite automata $\mathcal{A} = (S_{\mathcal{A}}, \Delta_{\mathcal{A}}, s^{in}_{\mathcal{A}})$ and $\mathcal{B}_1 = (S_{\mathcal{B}_1}, \Delta_{\mathcal{B}_1}, s^{in}_{\mathcal{B}_1}), \ldots, \mathcal{B}_n = (S_{\mathcal{B}_n}, \Delta_{\mathcal{B}_n}, s^{in}_{\mathcal{B}_n})$ over $(\{!,?\}\times\Pi)\cup\Sigma$ and $\Pi' \subseteq \Pi$, determine whether there exists a finite automaton $\mathcal{C} = (S_{\mathcal{C}}, \Delta_{\mathcal{C}}, s^{in}_{\mathcal{C}})$ over $\{!,?\} \times \Pi$ such that $FA(\mathcal{A}) \longleftrightarrow FA(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes \mathcal{C})$ $(\{!,?\} \times \Pi')$.

In CP, $\mathcal{A}$ plays the role of the given finite automaton with input and output and $\mathcal{B}_1$, …, $\mathcal{B}_n$ play the role of the available Web services. As for the finite automaton $\mathcal{C}$, it plays the role of the Web service that will combine and coordinate the available Web services into a complex business process that can simulate the given finite automaton with input and output. Take the case of $\mathcal{A}$, $\mathcal{B}_1$, $\mathcal{B}_2$, the finite automata from figure 3. Then the finite automaton $\mathcal{C}$ from figure 4 is such that $FA(\mathcal{A}) \longleftrightarrow FA(\mathcal{B}_1 \otimes \mathcal{B}_2 \otimes \mathcal{C})$ $(\{!,?\} \times \{\pi_1, \pi'_1, \pi_2, \pi'_2\})$.

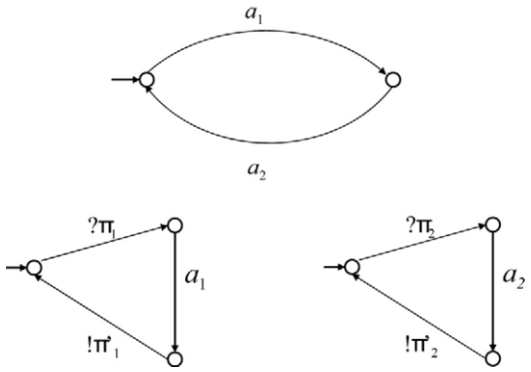

Fig. 3.

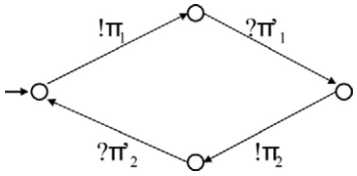

Fig. 4.

5 Lower bound

Now, we are ready to announce the first result of this paper:

$$CP \text{ is } EXPTIME\text{-hard.}$$

Let Σ be a finite set of actions. A Petri net over Σ is a structure of the form $\mathcal{N} = (P, T, F, l)$ where P is a finite set of places, T is a finite set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ is a relation, l is a function

- $l\colon T \to \Sigma$.

The Petri nets considered in this paper are 1-safe, that is to say: their places cannot contain more than one token (see [19] for details). For all $t \in T$, let the preset denoted ${}^\bullet t$ be the set of all $p \in P$ such that $p\ F\ t$ and the postset denoted $t^\bullet$ be the set of all $p \in P$ such that $t\ F\ p$. By $FA(\mathcal{N})$, we denote the finite automaton $\mathcal{A}' = (S', \Delta', u^{in'})$ over Σ such that $S' = 2^P$, Δ' is the function defined by

- $v' \in \Delta'(u', a)$ iff there exists $t \in T$ such that $l(t) = a$, ${}^\bullet t \subseteq u'$ and $v' = (u' \setminus {}^\bullet t) \cup t^\bullet$,

$u^{in'} = \emptyset$. Let us consider the following decision problem:

- PN: given a finite set Σ of actions and Petri nets $\mathcal{N} = (P_\mathcal{N}, T_\mathcal{N}, F_\mathcal{N}, l_\mathcal{N})$, $\mathcal{O} = (P_\mathcal{O}, T_\mathcal{O}, F_\mathcal{O}, l_\mathcal{O})$ over Σ, determine whether $FA(\mathcal{N}) \longleftrightarrow FA(\mathcal{O})$ $(\emptyset)$.

Seeing that PN is $EXPTIME$-hard [13], it suffices to reduce PN to the restriction of CP where $n = 1$, in order to demonstrate that CP is $EXPTIME$-hard.

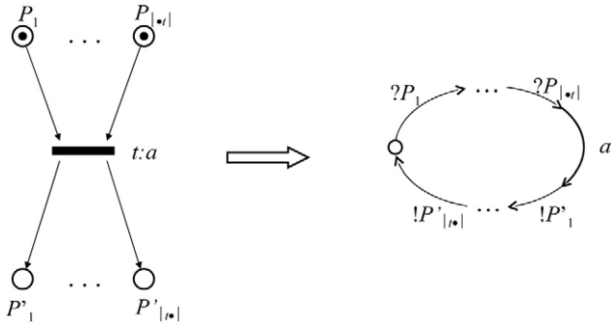

Fig. 5.

Given a finite set Σ of actions and Petri nets $\mathcal{N} = (P_\mathcal{N}, T_\mathcal{N}, F_\mathcal{N}, l_\mathcal{N})$, $\mathcal{O} = (P_\mathcal{O}, T_\mathcal{O}, F_\mathcal{O}, l_\mathcal{O})$ over Σ, we are asked whether $FA(\mathcal{N}) \longleftrightarrow FA(\mathcal{O})$ $(\emptyset)$. The instance $\rho(\Sigma, \mathcal{N}, \mathcal{O})$ of CP that we construct is given by the finite set Σ^e of actions, the finite set Π of channels, the finite automata $\mathcal{A} = (S_\mathcal{A}, \Delta_\mathcal{A}, s^{in}_\mathcal{A})$ and $\mathcal{B} = (S_\mathcal{B}, \Delta_\mathcal{B}, s^{in}_\mathcal{B})$ over $(\{!, ?\} \times \Pi) \cup \Sigma^e$ and $\Pi' \subseteq \Pi$ defined by

- $\Sigma^e = \Sigma \cup \{a^e_1, a^e_2, a^e_3, a^e_4\}$ where a^e_1, a^e_2, a^e_3 and a^e_4 are new actions,
- $\Pi = P_\mathcal{N} \cup P_\mathcal{O}$,
- $\Pi' = P_\mathcal{N} \cup P_\mathcal{O}$,
- $\mathcal{A}$ is the finite automaton from figure 6,
- $\mathcal{B}$ is the finite automaton from figure 7.

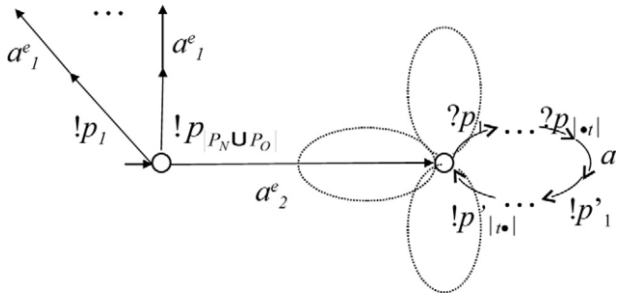

Fig. 6.

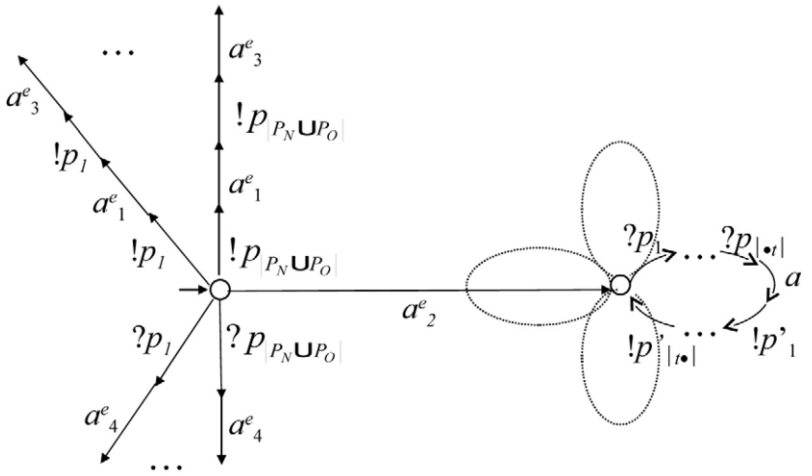

Fig. 7.

In order to understand how the flower-form parts of $\mathcal{A}$ and $\mathcal{B}$ are defined, the reader is invited to consult figure 5. This figure shows that the firing of the transition a empties the places $P_1, \ldots, P_{|\bullet t|}$ that are in the preset of the transition and fills up the places $P'_1, \ldots, P'_{|t\bullet|}$ that are in the postset of this transition. This transition is represented in the automata $\mathcal{A}$ and $\mathcal{B}$ by the following sequence of actions: receive messages on the channels $P_1, \ldots, P_{|\bullet t|}$ corresponding to the places in the preset of the transition, which has the effect to empty the channels, then the transition a followed by the emission of messages in the channels $P'_1, \ldots, P'_{|t\bullet|}$ corresponding to the places in the postset of the transition, which has the effect to fill up these channels. The actions sequences of the automata $\mathcal{A}$ and $\mathcal{B}$ containing the actions a^e_1, a^e_2, a^e_3 or a^e_4 ensure that no mediator can interfere with the simulation of the 1-safe Petri nets $\mathcal{N}$ and $\mathcal{O}$. This completes the construction. The flower part of $\mathcal{A}$ (resp. of $\mathcal{B}$) will be denoted $\mathcal{A}'$ (resp. $\mathcal{B}'$). The construction of $\mathcal{A}$ and $\mathcal{B}$ is done such that $FA(\mathcal{A}')$ and $FA(\mathcal{N})$ are isomorphic modulo $\{!, ?\} \times \Pi'$. In the same way, $FA(\mathcal{B}')$ and $FA(\mathcal{O})$ are isomorphic modulo $\{!, ?\} \times \Pi'$. Obviously, ρ can be computed in logarithmic space. Moreover, $FA(\mathcal{N}) \longleftrightarrow FA(\mathcal{O})$ $(\emptyset)$ iff there exists a finite automaton $\mathcal{C} = (S_\mathcal{C}, \Delta_\mathcal{C}, s^{in}_\mathcal{C})$ over $\{!, ?\} \times \Pi$ such that $FA(\mathcal{A}) \longleftrightarrow FA(\mathcal{B} \otimes \mathcal{C})$ $(\{!, ?\} \times \Pi')$.

Concerning the left to right implication, suppose that the Petri nets $\mathcal{N}$ and $\mathcal{O}$ are bisimilar. Hence, $FA(\mathcal{A}')$ and $FA(\mathcal{B}')$ are bisimilar modulo $\{!,?\} \times \Pi'$. Let $\mathcal{C}$ be the finite automaton that does nothing, i.e. $\mathcal{C}$ contains only one state (its initial state) and has no transition. With such a $\mathcal{C}$, the automata $FA(\mathcal{B})$ and $FA(\mathcal{B} \otimes \mathcal{C})$ are isomorphic. Thus, it is enough to prove that $FA(\mathcal{A})$ and $FA(\mathcal{B})$ are bisimilar modulo $\{!,?\} \times \Pi'$. Indeed, $FA(\mathcal{A}')$ and $FA(\mathcal{B}')$ are bisimilar modulo $\{!,?\} \times \Pi'$. Moreover, in $FA(\mathcal{A})$ and $FA(\mathcal{B})$, the transitions labelled $!p$ and a_1^e are executable for all places $p \in P_{\mathcal{N}} \cup P_{\mathcal{O}}$ whereas, in $FA(\mathcal{B})$, the transitions labelled $?p$, a_4^e and a_3^e are executable for no place $p \in P_{\mathcal{N}} \cup P_{\mathcal{O}}$. Hence, with such a $\mathcal{C}$, $FA(\mathcal{A}) \longleftrightarrow FA(\mathcal{B} \otimes \mathcal{C})$ $(\{!,?\} \times \Pi')$.

Concerning the right to left implication, suppose that the Petri nets $\mathcal{N}$ and $\mathcal{O}$ are not bisimilar and that there exists a finite automaton $\mathcal{C}$ such that $FA(\mathcal{A})$ and $FA(\mathcal{B})$ are bisimilar modulo $\{!,?\} \times \Pi'$. In the case $\mathcal{C}$ is the automaton that does nothing, the fact that $FA(\mathcal{A})$ and $FA(\mathcal{B} \otimes \mathcal{C})$ are bisimilar modulo $\{!,?\} \times \Pi'$ implies that $FA(\mathcal{A}')$ and $FA(\mathcal{B}')$ are bisimilar modulo $\{!,?\} \times \Pi'$. This contradicts the fact that $\mathcal{N}$ and $\mathcal{O}$ are not bisimilar. Hence, $\mathcal{C}$ has, at least, a $!p$ transition or a $?p$ transition starting from its initial state for some place $p \in P_{\mathcal{N}} \cup P_{\mathcal{O}}$. If $\mathcal{C}$ has a $!p$ transition starting from its initial state, then, in $FA(\mathcal{B} \otimes \mathcal{C})$, the transitions $!p$ (executed by $\mathcal{C}$), $?p$ (executed by $\mathcal{B}$) and a_4^e (executed by $\mathcal{B}$) can be executed from the initial state, whereas no sequence in $FA(\mathcal{A})$ ends with a transition labelled a_4^e. This contradicts the fact that $FA(\mathcal{A})$ and $FA(\mathcal{B} \otimes \mathcal{C})$ are bisimilar modulo $\{!,?\} \times \Pi'$. If $\mathcal{C}$ has a $?p$ transition starting from its initial state, then, in $FA(\mathcal{B} \otimes \mathcal{C})$, the transitions $!p$ (executed by $\mathcal{B}$), $?p$ (executed by $\mathcal{C}$), a_1^e (executed by $\mathcal{B}$), $!p$ (executed by $\mathcal{B}$) and a_3^e (executed by $\mathcal{C}$) can be executed from the initial state, whereas no sequence in $FA(\mathcal{A})$ ends with a transition labelled a_3^e. This contradicts the fact that $FA(\mathcal{A})$ and $FA(\mathcal{B} \otimes \mathcal{C})$ are bisimilar modulo $\{!,?\} \times \Pi'$.

Hence, CP is $EXPTIME$-hard.

6 Upper bound

Whether CP is in $EXPTIME$ or not is not known to us. Now, we are ready to announce the second result of this paper:

$$CP \text{ is in } 2EXPTIME.$$

6.1 A 2EXPTIME decision procedure based on controller synthesis

Let us consider the following decision problem:

- CS: given a finite set Σ of actions, a finite set Π of channels, a finite automaton $\mathcal{B} = (S_{\mathcal{B}}, \Delta_{\mathcal{B}}, s_{\mathcal{B}}^{in})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ and a modal μ-calculus formula ϕ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$, determine whether there exists a finite automaton $\mathcal{C} = (S_{\mathcal{C}}, \Delta_{\mathcal{C}}, s_{\mathcal{C}}^{in})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!,?\} \times \Pi) \cup \Sigma$ and such that $\mathcal{B} \times \mathcal{C} \models \phi$.

Arnold *et al.* [4] have proposed a decision procedure to resolve this problem. This procedure is based on modal μ-calculus. The language of modal μ-calculus cannot

express the fact that a finite automaton over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ loops over $(\{!,?\} \times \Pi) \cup \Sigma$. That is why Arnold *et al.* [4] extend it in such a way that looping becomes expressible. This extension is called modal-loop μ-calculus. It consists in associating with each $\theta \in (\{!,?\} \times \Pi) \cup \Sigma$ a proposition λ_θ whose interpretation is that a state s of a finite automaton $\mathcal{A} = (S, \Delta, s^{in})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ satisfies λ_θ iff $\Delta(s,\theta) = \{s\}$. Thus, one can construct in polynomial time a modal-loop μ-calculus formula $g(\Pi,\Sigma)$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ of polynomial size such that for all finite automata $\mathcal{A} = (S, \Delta, s^{in})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$, $\mathcal{A} \models g(\Pi,\Sigma)$ iff $\mathcal{A}$ loops over $(\{!,?\} \times \Pi) \cup \Sigma$. Moreover, given a finite automaton $\mathcal{B} = (S_\mathcal{B}, \Delta_\mathcal{B}, s^{in}_\mathcal{B})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ and a modal μ-calculus formula ϕ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$, Arnold *et al.* [4] show how to construct in polynomial time a modal μ-calculus formula $\phi/\mathcal{B}$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ of polynomial size such that for all finite automata $\mathcal{C} = (S_\mathcal{C}, \Delta_\mathcal{C}, s^{in}_\mathcal{C})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$, $\mathcal{B} \times \mathcal{C} \models \phi$ iff $\mathcal{C} \models \phi/\mathcal{B}$. Hence, CS is equivalent to the following decision problem:

- given a finite set Σ of actions, a finite set Π of channels, a finite automaton $\mathcal{B} = (S_\mathcal{B}, \Delta_\mathcal{B}, s^{in}_\mathcal{B})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ and a modal μ-calculus formula ϕ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$, determine whether $\phi/\mathcal{B} \wedge g(\Pi,\Sigma)$ is satisfiable.

Now, let us go back to CP and take a finite set Σ of actions, a finite set Π of channels, finite automata $\mathcal{A} = (S_\mathcal{A}, \Delta_\mathcal{A}, s^{in}_\mathcal{A})$ and $\mathcal{B}_1 = (S_{\mathcal{B}_1}, \Delta_{\mathcal{B}_1}, s^{in}_{\mathcal{B}_1})$, ..., $\mathcal{B}_n = (S_{\mathcal{B}_n}, \Delta_{\mathcal{B}_n}, s^{in}_{\mathcal{B}_n})$ over $(\{!,?\} \times \Pi) \cup \Sigma$ and $\Pi' \subseteq \Pi$. To determine whether there exists a finite automaton $\mathcal{C} = (S_\mathcal{C}, \Delta_\mathcal{C}, s^{in}_\mathcal{C})$ over $\{!,?\} \times \Pi$ such that $FA(\mathcal{A}) \longleftrightarrow FA(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes \mathcal{C})(\{!,?\} \times \Pi')$, we consider the following algorithm:

(i) Compute $\phi = f(FA(\mathcal{A}), \Pi')$.

(ii) Compute $\mathcal{B} = FA^\circ(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes L^\circ)$.

(iii) Compute $\phi' = \phi/\mathcal{B} \wedge g(\Pi,\Sigma)$.

(iv) If ϕ' is satisfiable then return the value $true$ else return the value $false$.

By theorem 3.4, the above algorithm returns the value $true$ iff there exists a finite automaton $\mathcal{C}$ over $\{!,?\} \times \Pi$ such that $FA(\mathcal{A}) \longleftrightarrow FA(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes \mathcal{C})$ $(\{!,?\} \times \Pi')$. It can be implemented in double exponential time. More precisely:

- In step (i), the computation of $FA(\mathcal{A})$ takes exponential time in the size of Π whereas the computation of ϕ (the existence of which is implied by lemma 3.3) takes polynomial time in the size of $FA(\mathcal{A})$. Moreover, the size of $FA(\mathcal{A})$ is exponential in the size of Π whereas the size of ϕ is polynomial in the size of $FA(\mathcal{A})$.
- In step (ii), the computation of $\mathcal{B}_1 \otimes, \ldots, \otimes \mathcal{B}_n \otimes L^\circ$ takes exponential time in the size of $\mathcal{B}_1, \ldots, \mathcal{B}_n$ whereas the computation of $\mathcal{B}$ takes exponential time in the size of Π. Hence, the computation of $\mathcal{B}$ takes exponential time in the size of $\mathcal{B}_1, \ldots, \mathcal{B}_n$ and in the size of Π. Moreover, the size of $\mathcal{B}$ is exponential in the size of $\mathcal{B}_1, \ldots, \mathcal{B}_n$ and in the size of Π.
- In step (iii), the computation of ϕ' can be done in polynomial time in the size of ϕ, $\mathcal{B}$, Π and Σ. Moreover, the size of ϕ' is polynomial in the size of ϕ, $\mathcal{B}$, Π and

Σ. Hence, the size of ϕ' is exponential in the size of $\mathcal{A}$, $\mathcal{B}_1, \ldots, \mathcal{B}_n$ and Π.

- Step (iv) can be executed in deterministic exponential (with respect to the size of ϕ') time, seeing that the satisfiability problem for the modal-loop μ-calculus is in $EXPTIME$.

6.2 A 2EXPTIME decision procedure based on filtration

Let us consider the following decision problem:

- FIL: given a finite set Σ of actions, a finite set Π of channels, a finite automaton $\mathcal{A} = (S_{\mathcal{A}}, \Delta_{\mathcal{A}}, s^{in}_{\mathcal{A}})$ over $(\{!,?\} \times \Pi) \cup \Sigma$, a finite automaton $\mathcal{B} = (S_{\mathcal{B}}, \Delta_{\mathcal{B}}, s^{in}_{\mathcal{B}})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ and $\Pi' \subseteq \Pi$, determine whether there exists a finite automaton $\mathcal{C} = (S_{\mathcal{C}}, \Delta_{\mathcal{C}}, s^{in}_{\mathcal{C}})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!,?\} \times \Pi) \cup \Sigma$ and such that $\mathcal{A} \longleftrightarrow Del^\circ(\mathcal{B} \times \mathcal{C})$ $(\{!,?\} \times \Pi')$.

Suppose that we are given a finite set Σ of actions, a finite set Π of channels, a finite automaton $\mathcal{A} = (S_{\mathcal{A}}, \Delta_{\mathcal{A}}, s^{in}_{\mathcal{A}})$ over $(\{!,?\} \times \Pi) \cup \Sigma$, a finite automaton $\mathcal{B} = (S_{\mathcal{B}}, \Delta_{\mathcal{B}}, s^{in}_{\mathcal{B}})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ and $\Pi' \subseteq \Pi$. Let $\mathcal{C} = (S_{\mathcal{C}}, \Delta_{\mathcal{C}}, s^{in}_{\mathcal{C}})$ be a finite automaton over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!,?\} \times \Pi) \cup \Sigma$ and such that $\mathcal{A} \longleftrightarrow Del^\circ(\mathcal{B} \times \mathcal{C})$ $(\{!,?\} \times \Pi')$. Hence, there exists a bisimulation Z between $\mathcal{A}$ and $Del^\circ(\mathcal{B} \times \mathcal{C})$ modulo $(\{!,?\} \times \Pi')$ such that $s^{in}_{\mathcal{A}}\ Z\ (s^{in}_{\mathcal{B}}, s^{in}_{\mathcal{C}})$. Let $\equiv\ \subseteq S_{\mathcal{C}} \times S_{\mathcal{C}}$ be the binary relation such that for all $s^1_{\mathcal{C}}, s^2_{\mathcal{C}} \in S_{\mathcal{C}}$,

- $s^1_{\mathcal{C}} \equiv s^2_{\mathcal{C}}$ iff for all $s_{\mathcal{A}} \in S_{\mathcal{A}}$ and for all $s_{\mathcal{B}} \in S_{\mathcal{B}}$, $s_{\mathcal{A}}\ Z\ (s_{\mathcal{B}}, s^1_{\mathcal{C}})$ iff $s_{\mathcal{A}}\ Z\ (s_{\mathcal{B}}, s^2_{\mathcal{C}})$.

Note that $\equiv$ is an equivalence relation. Let $s_{\mathcal{C}} \in S_{\mathcal{C}}$. The set of all states in $S_{\mathcal{C}}$ equivalent to $s_{\mathcal{C}}$ modulo $\equiv$, in symbols $\mid s_{\mathcal{C}} \mid$, is called the equivalence class of $s_{\mathcal{C}}$ in $S_{\mathcal{C}}$ modulo $\equiv$ with $s_{\mathcal{C}}$ as its representative. The set of all equivalence classes of $S_{\mathcal{C}}$ modulo $\equiv$, in symbols $S_{\mathcal{C}}/\equiv$, is called the quotient set of $S_{\mathcal{C}}$ modulo $\equiv$. Suppose that $\mathcal{C}^f = (S_{\mathcal{C}^f}, \Delta_{\mathcal{C}^f}, s^{in}_{\mathcal{C}^f})$ is the finite automaton over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!,?\} \times \Pi) \cup \Sigma$ and such that $S_{\mathcal{C}^f} = S_{\mathcal{C}}/\equiv$, $\Delta_{\mathcal{C}^f}$ is the function such that

- $\mid t_{\mathcal{C}} \mid \in \Delta_{\mathcal{C}^f}(\mid s_{\mathcal{C}} \mid, \vartheta\pi^\circ)$, $\vartheta \in \{!,?\}$ and $\pi^\circ \in \Pi^\circ$, iff for all $s_{\mathcal{A}} \in S_{\mathcal{A}}$ and for all $s_{\mathcal{B}}, t_{\mathcal{B}} \in S_{\mathcal{B}}$, if $t_{\mathcal{B}} \in \Delta_{\mathcal{B}}(s_{\mathcal{B}}, \vartheta\pi^\circ)$ and $s_{\mathcal{A}}\ Z\ (s_{\mathcal{B}}, s_{\mathcal{C}})$ then there exists $t_{\mathcal{A}} \in S_{\mathcal{A}}$ such that $t_{\mathcal{A}} \in \Delta_{\mathcal{A}}(s_{\mathcal{A}}, \vartheta\pi)$ and $t_{\mathcal{A}}\ Z\ (t_{\mathcal{B}}, t_{\mathcal{C}})$,

$s^{in}_{\mathcal{C}^f} = \mid s^{in}_{\mathcal{C}} \mid$. Then $\mathcal{C}^f$ is called the greatest filtration of $\mathcal{C}$ through $\mathcal{A}$ and $\mathcal{B}$. Let $Z^f \subseteq S_{\mathcal{A}} \times (S_{\mathcal{B}} \times S_{\mathcal{C}^f})$ be the binary relation such that for all $s_{\mathcal{A}} \in S_{\mathcal{A}}$ and for all $(s_{\mathcal{B}}, \mid s_{\mathcal{C}} \mid) \in S_{\mathcal{B}} \times S_{\mathcal{C}^f}$, $s_{\mathcal{A}}\ Z^f\ (s_{\mathcal{B}}, \mid s_{\mathcal{C}} \mid)$ iff $s_{\mathcal{A}}\ Z\ (s_{\mathcal{B}}, s_{\mathcal{C}})$. It is a simple matter to check that Z^f: $\mathcal{A} \longleftrightarrow Del^\circ(\mathcal{B} \times \mathcal{C}^f)$ $(\{!,?\} \times \Pi')$. For our purpose, the crucial property of the greatest filtration is that the following conditions are equivalent:

- there exists a finite automaton $\mathcal{C} = (S_{\mathcal{C}}, \Delta_{\mathcal{C}}, s^{in}_{\mathcal{C}})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!,?\} \times \Pi) \cup \Sigma$ and there exists a relation $Z \subseteq S_{\mathcal{A}} \times (S_{\mathcal{B}} \times S_{\mathcal{C}})$ such that Z: $\mathcal{A} \longleftrightarrow Del^\circ(\mathcal{B} \times \mathcal{C})$ $(\{!,?\} \times \Pi')$,
- there exists a finite automaton $\mathcal{C} = (S_{\mathcal{C}}, \Delta_{\mathcal{C}}, s^{in}_{\mathcal{C}})$ over $(\{!,?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!,?\} \times \Pi) \cup \Sigma$ and there exists a relation $Z \subseteq S_{\mathcal{A}} \times (S_{\mathcal{B}} \times S_{\mathcal{C}})$ such that Z: $\mathcal{A} \longleftrightarrow Del^\circ(\mathcal{B} \times \mathcal{C})$ $(\{!,?\} \times \Pi')$ and
  (C_1) $S_{\mathcal{C}} \subseteq 2^{S_{\mathcal{A}} \times S_{\mathcal{B}}}$,

(C_2) $t_{\mathcal{C}} \in \Delta_{\mathcal{C}}(s_{\mathcal{C}}, \vartheta\pi^\circ)$, $\vartheta \in \{!, ?\}$ and $\pi^\circ \in \Pi^\circ$, iff for all $s_{\mathcal{A}} \in S_{\mathcal{A}}$ and for all $s_{\mathcal{B}}, t_{\mathcal{B}} \in S_{\mathcal{B}}$, if $t_{\mathcal{B}} \in \Delta_{\mathcal{B}}(s_{\mathcal{B}}, \vartheta\pi^\circ)$ and $(s_{\mathcal{A}}, s_{\mathcal{B}}) \in s_{\mathcal{C}}$ then there exists $t_{\mathcal{A}} \in S_{\mathcal{A}}$ such that $t_{\mathcal{A}} \in \Delta_{\mathcal{A}}(s_{\mathcal{A}}, \vartheta\pi)$ and $(t_{\mathcal{A}}, t_{\mathcal{B}}) \in t_{\mathcal{C}}$,

(C_3) $s_{\mathcal{A}}\ Z\ (s_{\mathcal{B}}, s_{\mathcal{C}})$ iff $(s_{\mathcal{A}}, s_{\mathcal{B}}) \in s_{\mathcal{C}}$.

Hence, we can give a simple algorithm for solving FIL:

(i) Compute the finite automaton $\mathcal{C}^0 = (S_{\mathcal{C}}^0, \Delta_{\mathcal{C}}^0, {s_{\mathcal{C}}^{in}}^0)$ over $(\{!, ?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!, ?\} \times \Pi) \cup \Sigma$ and the relation $Z^0 \subseteq S_{\mathcal{A}} \times (S_{\mathcal{B}} \times S_{\mathcal{C}}^0)$ such that
- $S_{\mathcal{C}}^0 = 2^{S_{\mathcal{A}} \times S_{\mathcal{B}}}$,
- $t_{\mathcal{C}} \in \Delta_{\mathcal{C}}^0(s_{\mathcal{C}}, \vartheta\pi^\circ)$, $\vartheta \in \{!, ?\}$ and $\pi^\circ \in \Pi^\circ$, iff for all $s_{\mathcal{A}} \in S_{\mathcal{A}}$ and for all $s_{\mathcal{B}}, t_{\mathcal{B}} \in S_{\mathcal{B}}$, if $t_{\mathcal{B}} \in \Delta_{\mathcal{B}}(s_{\mathcal{B}}, \vartheta\pi^\circ)$ and $(s_{\mathcal{A}}, s_{\mathcal{B}}) \in s_{\mathcal{C}}$ then there exists $t_{\mathcal{A}} \in S_{\mathcal{A}}$ such that $t_{\mathcal{A}} \in \Delta_{\mathcal{A}}(s_{\mathcal{A}}, \vartheta\pi)$ and $(t_{\mathcal{A}}, t_{\mathcal{B}}) \in t_{\mathcal{C}}$,
- $s_{\mathcal{A}}\ Z^0\ (s_{\mathcal{B}}, s_{\mathcal{C}})$ iff $(s_{\mathcal{A}}, s_{\mathcal{B}}) \in s_{\mathcal{C}}$.

(ii) For all non negative integers n, compute the set $\lhd^n$ of all $s_{\mathcal{C}} \in {S_{\mathcal{C}}}^n$ such that for some $s_{\mathcal{A}} \in S_{\mathcal{A}}$ and for some $s_{\mathcal{B}} \in S_{\mathcal{B}}$ such that $s_{\mathcal{A}}\ Z^n\ (s_{\mathcal{B}}, s_{\mathcal{C}})$, there exists $\vartheta \in \{!, ?\}$ and there exists $\pi^\circ \in \Pi^\circ$ such that one of the following cases holds:
- there exists $t_{\mathcal{A}} \in \Delta_{\mathcal{A}}(s_{\mathcal{A}}, \vartheta\pi)$ such that for all $t_{\mathcal{B}} \in \Delta_{\mathcal{B}}(s_{\mathcal{B}}, \vartheta\pi^\circ)$ and for all $t_{\mathcal{C}} \in \Delta_{\mathcal{C}}^n(s_{\mathcal{C}}, \vartheta\pi^\circ)$, not $t_{\mathcal{A}}\ Z^n\ (t_{\mathcal{B}}, t_{\mathcal{C}})$,
- there exists $t_{\mathcal{B}} \in \Delta_{\mathcal{B}}(s_{\mathcal{B}}, \vartheta\pi^\circ)$ and there exists $t_{\mathcal{C}} \in \Delta_{\mathcal{C}}^n(s_{\mathcal{C}}, \vartheta\pi^\circ)$ such that for all $t_{\mathcal{A}} \in \Delta_{\mathcal{A}}(s_{\mathcal{A}}, \vartheta\pi)$, not $t_{\mathcal{A}}\ Z^n\ (t_{\mathcal{B}}, t_{\mathcal{C}})$.

(iii) For all non negative integers n, compute the finite automaton $\mathcal{C}^{n+1} = (S_{\mathcal{C}}^{n+1}, \Delta_{\mathcal{C}}^{n+1}, {s_{\mathcal{C}}^{in}}^{n+1})$ over $(\{!, ?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!, ?\} \times \Pi) \cup \Sigma$ and the relation $Z^{n+1} \subseteq S_{\mathcal{A}} \times (S_{\mathcal{B}} \times S_{\mathcal{C}}^{n+1})$ such that
- $S_{\mathcal{C}}^{n+1} = S_{\mathcal{C}}^n \setminus \lhd^n$,
- $t_{\mathcal{C}} \in \Delta_{\mathcal{C}}^{n+1}(s_{\mathcal{C}}, \vartheta\pi^\circ)$, $\vartheta \in \{!, ?\}$ and $\pi^\circ \in \Pi^\circ$, iff for all $s_{\mathcal{A}} \in S_{\mathcal{A}}$ and for all $s_{\mathcal{B}}, t_{\mathcal{B}} \in S_{\mathcal{B}}$, if $t_{\mathcal{B}} \in \Delta_{\mathcal{B}}(s_{\mathcal{B}}, \vartheta\pi^\circ)$ and $(s_{\mathcal{A}}, s_{\mathcal{B}}) \in s_{\mathcal{C}}$ then there exists $t_{\mathcal{A}} \in S_{\mathcal{A}}$ such that $t_{\mathcal{A}} \in \Delta_{\mathcal{A}}(s_{\mathcal{A}}, \vartheta\pi)$ and $(t_{\mathcal{A}}, t_{\mathcal{B}}) \in t_{\mathcal{C}}$,
- $s_{\mathcal{A}}\ Z^{n+1}\ (s_{\mathcal{B}}, s_{\mathcal{C}})$ iff $(s_{\mathcal{A}}, s_{\mathcal{B}}) \in s_{\mathcal{C}}$.

The following lemma shows that the above algorithm returns the value *true* iff there exists a finite automaton $\mathcal{C} = (S_{\mathcal{C}}, \Delta_{\mathcal{C}}, s_{\mathcal{C}}^{in})$ over $(\{!, ?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!, ?\} \times \Pi) \cup \Sigma$ and such that $\mathcal{A} \longleftrightarrow Del^\circ(\mathcal{B} \times \mathcal{C})\ (\{!, ?\} \times \Pi')$.

Lemma 6.1 *Let $\mathcal{C} = (S_{\mathcal{C}}, \Delta_{\mathcal{C}}, s_{\mathcal{C}}^{in})$ be a finite automaton over $(\{!, ?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!, ?\} \times \Pi) \cup \Sigma$ and $Z \subseteq S_{\mathcal{A}} \times (S_{\mathcal{B}} \times S_{\mathcal{C}})$ be a relation such that Z: $\mathcal{A} \longleftrightarrow Del^\circ(\mathcal{B} \times \mathcal{C})\ (\{!, ?\} \times \Pi')$ and*

(C_1) $S_{\mathcal{C}} \subseteq 2^{S_{\mathcal{A}} \times S_{\mathcal{B}}}$,

(C_2) $t_{\mathcal{C}} \in \Delta_{\mathcal{C}}(s_{\mathcal{C}}, \vartheta\pi^\circ)$, $\vartheta \in \{!, ?\}$ *and* $\pi^\circ \in \Pi^\circ$, *iff for all* $s_{\mathcal{A}} \in S_{\mathcal{A}}$ *and for all* $s_{\mathcal{B}}, t_{\mathcal{B}} \in S_{\mathcal{B}}$, *if* $t_{\mathcal{B}} \in \Delta_{\mathcal{B}}(s_{\mathcal{B}}, \vartheta\pi^\circ)$ *and* $(s_{\mathcal{A}}, s_{\mathcal{B}}) \in s_{\mathcal{C}}$ *then there exists* $t_{\mathcal{A}} \in S_{\mathcal{A}}$ *such that* $t_{\mathcal{A}} \in \Delta_{\mathcal{A}}(s_{\mathcal{A}}, \vartheta\pi)$ *and* $(t_{\mathcal{A}}, t_{\mathcal{B}}) \in t_{\mathcal{C}}$,

(C_3) $s_{\mathcal{A}}\ Z\ (s_{\mathcal{B}}, s_{\mathcal{C}})$ *iff* $(s_{\mathcal{A}}, s_{\mathcal{B}}) \in s_{\mathcal{C}}$.

Then, for all $s_{\mathcal{C}} \in S_{\mathcal{C}}$ and for all non negative integers n, $s_{\mathcal{C}} \in {S_{\mathcal{C}}}^n$.

Proof. Let $s_{\mathcal{C}} \in S_{\mathcal{C}}$. If there exists a non negative integer n such that $s_{\mathcal{C}} \in {S_{\mathcal{C}}}^n$ and $s_{\mathcal{C}} \notin {S_{\mathcal{C}}}^{n+1}$ then for some $s_{\mathcal{A}} \in S_{\mathcal{A}}$ and for some $s_{\mathcal{B}} \in S_{\mathcal{B}}$ such that $s_{\mathcal{A}}\ Z^n\ (s_{\mathcal{B}}, s_{\mathcal{C}})$,

there exists $\vartheta \in \{!, ?\}$ and there exists $\pi^\circ \in \Pi^\circ$ such that one of the following cases holds:

- there exists $t_{\mathcal{A}} \in \Delta_{\mathcal{A}}(s_{\mathcal{A}}, \vartheta\pi)$ such that for all $t_{\mathcal{B}} \in \Delta_{\mathcal{B}}(s_{\mathcal{B}}, \vartheta\pi^\circ)$ and for all $t_{\mathcal{C}} \in \Delta^n_{\mathcal{C}}(s_{\mathcal{C}}, \vartheta\pi^\circ)$, not $t_{\mathcal{A}}\ Z^n\ (t_{\mathcal{B}}, t_{\mathcal{C}})$,
- there exists $t_{\mathcal{B}} \in \Delta_{\mathcal{B}}(s_{\mathcal{B}}, \vartheta\pi^\circ)$ and there exists $t_{\mathcal{C}} \in \Delta^n_{\mathcal{C}}(s_{\mathcal{C}}, \vartheta\pi^\circ)$ such that for all $t_{\mathcal{A}} \in \Delta_{\mathcal{A}}(s_{\mathcal{A}}, \vartheta\pi)$, not $t_{\mathcal{A}}\ Z^n\ (t_{\mathcal{B}}, t_{\mathcal{C}})$.

The reader may easily verify that both cases lead to a contradiction. □

An obvious analysis of the complexity of the above algorithm yields the following facts:

- there exists a non negative integer n such that $n \leq 2^{Card(S_{\mathcal{A}}) \times Card(S_{\mathcal{B}})}$ and $\mathcal{C}^{n+1} = \mathcal{C}^n$,
- for all non negative integers n, $\mathcal{C}^{n+1}$ can be obtained from $\mathcal{C}^n$ in time polynomial in the size of $\mathcal{C}^n$.

Hence, the above algorithm can be implemented in exponential time. Now, let us go back to CP and take a finite set Σ of actions, a finite set Π of channels, finite automata $\mathcal{A} = (S_{\mathcal{A}}, \Delta_{\mathcal{A}}, s^{in}_{\mathcal{A}})$ and $\mathcal{B}_1 = (S_{\mathcal{B}_1}, \Delta_{\mathcal{B}_1}, s^{in}_{\mathcal{B}_1}), \ldots, \mathcal{B}_n = (S_{\mathcal{B}_n}, \Delta_{\mathcal{B}_n}, s^{in}_{\mathcal{B}_n})$ over $(\{!, ?\} \times \Pi) \cup \Sigma$ and $\Pi' \subseteq \Pi$. To determine whether there exists a finite automaton $\mathcal{C} = (S_{\mathcal{C}}, \Delta_{\mathcal{C}}, s^{in}_{\mathcal{C}})$ over $\{!, ?\} \times \Pi$ such that $FA(\mathcal{A}) \longleftrightarrow FA(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes \mathcal{C})$, we consider the following algorithm:

(i) Compute $\mathcal{A}' = FA(\mathcal{A})$.

(ii) Compute $\mathcal{B}' = FA^\circ(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes L^\circ)$.

(iii) If there exists a finite automaton $\mathcal{C} = (S_{\mathcal{C}}, \Delta_{\mathcal{C}}, s^{in}_{\mathcal{C}})$ over $(\{!, ?\} \times (\Pi \cup \Pi^\circ)) \cup \Sigma$ looping over $(\{!, ?\} \times \Pi) \cup \Sigma$ and such that $\mathcal{A}' \longleftrightarrow Del^\circ(\mathcal{B}' \times \mathcal{C})\ (\{!, ?\} \times \Pi')$ then return the value $true$ else return the value $false$.

By theorem 3.4, the above algorithm returns the value $true$ iff there exists a finite automaton $\mathcal{C}$ over $\{!, ?\} \times \Pi$ such that $FA(\mathcal{A}) \longleftrightarrow FA(\mathcal{B}_1 \otimes \ldots \otimes \mathcal{B}_n \otimes \mathcal{C})\ (\{!, ?\} \times \Pi')$. It can be implemented in double exponential time.

7 Conclusion and open problems

We have presented a framework in which Web services are described as message passing automata. Deterministic algorithms that check a composition's existence and return one if it exists have been proposed. In order to ensure their termination in a finite number of steps, we have characterized the computational complexity ($EXPTIME$-hardness and membership in $2EXPTIME$) of the composition problem. Our main results are that CP is $EXPTIME$-hard and CP is in $2EXPTIME$. An interesting (and still open) question is to evaluate the exact complexity of Web service composition: is CP in $EXPTIME$ or is CP $2EXPTIME$-hard? Variants of CP can be considered as well. For instance, one may consider that the given automata are deterministic or that the channels they use can contain more than 1 message at a time. Concerning the second variant, the results obtained in this pa-

per remain true. The only difference is in the construction of $FA(\mathcal{A})$, for which the construction will also be done in exponential time. More precisely, if for some positive integer k, the channels used by automaton $\mathcal{A}$ can contain at most k messages at a time, then states in $FA(\mathcal{A})$ will be pairs of the form $(q, (k_1, \ldots, k_m))$, where q is a state of the automaton $\mathcal{A}$, m is the cardinality of the set Π of all channels and $(k_1, \ldots, k_m)$ is a sequence of m integers in $\{0, \ldots, k\}$. Take another example: one may replace "bisimulation" by "trace equivalence". What is the complexity of Web service composition in this case? Let us remark that the construction used in section 5 for the case of bisimulation can also be used for the case of trace equivalence. Seeing that trace equivalence between 1-safe Petri nets is $EXPSPACE$-hard [13], it follows that Web services composition is $EXPSPACE$-hard in the case of trace equivalence. In other respects, we have not considered which message is actually sent/received when performing a messaging action. To enrich our formalism that way, we may augment each send/receive action with an additional first-order term indicating what kind of message is exchanged. Henceforth, a message exchange action consists of a channel π and a first-order term t which indicate that a message of the form t is sent or received through channel π. For which classes of messages is Web service composition decidable? When this problem is decidable, how complex is it?

Acknowledgement

The work presented in this paper was partially supported by the FP7-ICT-2007-1 Project no. 216471, "AVANTSSAR: Automated Validation of Trust and Security of Service-Oriented Architectures" (www.avantssar.eu). We are also grateful for the support of the project ANR-05-SSIA-0007-01 *Cops* financed by the "Agence nationale de la recherche" (www.irit.fr/COPS/Accueil.htm).

References

[1] Alvarez, C., Balcázar, J., Gabarró, J., Sántha, M.: *Parallel complexity in the design and analysis of concurrent systems.* In: *PARLE '91, Parallel Architectures and Languages Europe.* Springer-Verlag (1991) 288–303.

[2] Andrews, T., F. Curbera , H. Dholakia , Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weerawarana. 'Business Process Execution Language for Web services (BPEL4WS)'. www-106.ibm.com/developerworks/library/ws-bpel/, (2004).

[3] Ariba, Microsoft, IBM. 'Web Services Description Language (WSDL)'. www.w3.org/TR/2001/NOTE-wsdl-20010315, (2001).

[4] Arnold, A., A. Vincent, I. Walukiewicz, *Games for synthesis of controllers with partial observation.* Theoretical computer science **303** (2003) 7–34.

[5] Berardi, D., D. Calvanese, G. De Giacomo, R. Hull, M. Mecella, 'Automated composition of transition-based semantic Web services with messaging'. In: 'Proceedings of the 31st International Conference on Very Large Data Bases'. VLDB Endowment (2005) 613–624.

[6] Berardi, D., D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella, 'Automatic composition of e-services that export their behavior'. In: 'Service-Oriented Computing — ICSOC 2003'. Springer (2003) 43–58.

[7] Berardi, D., F. Cheikh, G. De Giacomo, F. Patrizi, 'Automatic service composition via simulation'. In: 'International Journal of Foundations of Computer Science'. World Scientific Publishing **19** (2008) 429–451.

[8] Berwanger, D., E. Grädel, G. Lenzi, *The variable hierarchy of the μ-calculus is strict*. Theory of Computing Systems **40** (2007) 437–466.

[9] Cheikh, F., G. De Giacomo, M. Mecella, 'Automatic Web services composition in trust-aware communities'. In: 'Proceedings of the 3rd ACM Workshop on Secure Web Services'. Association for Computing Machinery (2006) 43–52.

[10] De Giacomo, G., M. De Leoni, M. Mecella, F. Patrizi, 'Automatic workflows composition of mobile services'. In:'Proceedings of the IEEE International Conference on Web Services'. International Conference on Web Services (2007) 823–830.

[11] Fitting, M., 'Proof Methods for Modal and Intuitionistic Logic'. Reidel (1983).

[12] Huhns, M., *Agents as Web services*. IEEE Internet Computing **6** (2002) 93–95.

[13] Jategaonkar, L., A. Meyer, *Deciding true concurrency equivalences on safe, finite nets*. Theoretical computer science **154** (1996) 107–143.

[14] Kavantzas, N., D. Burdett, et al. 'Web Services Choreography Description Language (WS-CDL)'. www.w3.org/TR/2004/WD-ws-cdl-10-20041217/, (2004).

[15] König, D., N. Lohmann, S. Moser, C. Stahl, K. Wolf, 'Extending the compatibility notion for abstract WS-BPEL processes'. In: 'Proceedings of the seventeenth International Conference on World Wide Web'. World Wide Web (2008) 785–794.

[16] Melliti, T., S. Haddad, A. Suna, A. El Fallah Seghrouchni, '$Web - MASI$: multi-agent systems interoperability using a Web services based approach'. In: 'Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology 2005'. IEEE (2005) 739–742.

[17] Milanovic, N., M. Malek, *Current solutions for Web service composition*. IEEE Internet Computing **8** (2004) 51–59.

[18] Müller, I., R. Kowalczyk, P. Braun, 'Towards agent-based coalition formation for service composition'. In: 'Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology 2006'. IEEE (2006) 73–80.

[19] Murata , T., *Petri nets: Properties, analyses and applications*. Proceedings of the IEEE **17** (1989) 541–574.

[20] Papadimitriou, C., 'Computational Complexity'. Addison-Wesley (1994).

[21] Pistore, M., F. Barbon, P. Bertoli, D. Shaparau, P. Traverso, 'Planning and monitoring Web service composition'. In: 'Artificial Intelligence: Methodology, Systems, and Applications'. Springer (2004) 106–115.

[22] Pistore, M., A. Marconi, P. Bertoli, P. Traverso, 'Automated composition of Web services by planning at the knowledge level'. In: 'Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence'. International Joint Conferences on Artificial Intelligence (2005) 1252–1259.

[23] Ross-Talbot S., 'Web services choreography and Process Algebra'. www.daml.org/services/swsl/materials/WS-CDL.pdf, (2004).

[24] Sardiña, S., F. Patrizi, G. De Giacomo, 'Automatic synthesis of global behavior from multiple distributed behaviors'. In:'Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence'. Conference on Artificial Intelligence (2007) 1063–1069.

[25] Traverso, P., M. Pistore, 'Automated composition of semantic Web services into executable processes'. In: 'The Semantic Web — ISWC 2004'. Springer (2004) 380–394.