A lower bound on web services composition

Anca Muscholl[1⋆] and Igor Walukiewicz[1∗]

LaBRI, Université Bordeaux 1 and CNRS
351, Cours de la Libération
F-33 405, Talence cedex, France

Abstract. A web service is modeled here as a finite state machine. A composition problem for web services is to decide if a given web service can be constructed from a given set of web services; where the construction is understood as a simulation of the specification by a fully asynchronous product of the given services. We show an EXPTIME-lower bound for this problem, thus matching the known upper bound. Our result also applies to richer models of web services, such as the Roman model.

Keywords: Automata simulation, complexity, web services composition.

1 Introduction

Inherently distributed applications such as web services [1] increasingly get into the focus of automated verification techniques. Often, some basic e-services are already implemented, but no such simple service can answer to a more complex query. For instance, a user interested in hiking Mt. Everest will ask a travel agency for information concerning weather forecast, group travels, guides etc. The travel agency will contact different e-services, asking for such information and making appropriate reservations, if places are available. In general, single services such as weather forecast or group reservations, are already available and it is important to be able to reuse them without any change. The task of the travel agency is to compose basic e-services in such a way that the user's requirements are met (and eventually some constraints wrt. the called services, such as avoiding unreliable ones). Thus, one main objective is to be able to check automatically that the composition of basic e-services satisfies certain desirable properties or realizes another complex e-service.

In this paper we study a problem that arises in the *composition* of e-services as considered in [2–4]. The setting is the following: we get as input a specification (goal) $\mathcal{B}$, together with n available services $\mathcal{A}_1, \ldots, \mathcal{A}_n$.

⋆ Work supported by the ACI Sécurité Informatique VERSYDIS.

Then we ask whether the composition of the services $\mathcal{A}_i$ can simulate the behavior of the goal $\mathcal{B}$. This problem is known as *composition synthesis*. It amounts to synthesize a so-called *delegator*, that tells at any moment which service must perform an action. In essence, a delegator corresponds to a simulation of the goal service $\mathcal{B}$ by the composition of the available services $\mathcal{A}_i$. In the most general setting, as considered for instance in [9, 8], services are modeled by communicating finite state machines [5], that have access to some local data. In this paper, we reconsider the simplified setting of the so-called Roman model [2] where services are finite state processes with no access to data. This restriction is severe, but it captures some quite natural cases. First, messages exchanged by services are often synchronous (hand-shaking), which means that we do not need the full power of communication channels. Second, even when data-driven web applications are considered, some restrictions on data are needed. For instance, [7] assumes that specific user information is considered as constants in the data base scheme.

The main result of this paper is the EXPTIME lower bound for the composition synthesis problem in the very simple setting where the composition of the finite state machines $\mathcal{A}_i$ is fully asynchronous (in particular there is no communication). We also show that the same question can be solved in polynomial time if we assume that the sets of actions of the available machines are pairwise disjoint, i.e., each request can be handled by precisely one service. Note that in the latter case, the set of actions depends of course on the number of processes, whereas for the first result we show that the case where the set of actions is fixed is already EXPTIME-hard. Thanks to the simplicity of the considered model the same lower bounds hold also for much richer frameworks, as for example Colombo [3]. For the Roman model a matching EXPTIME upper bound is known [2]. The complexity of the composition problem for Colombo depends on restrictions of the model and is undecidable in the most general case.

As related work, it is worth mentioning the approach of Pistore et al. [11] who use planning techniques. The other difference is that the final goal is specified by a formula, and not as a simulation condition as we have here. Moreover, the accent there is put on satisfying one demand, i.e., constructing a sequence of actions rather than a transition system, i.e. a new service. The other possibility is to consider bisimulation instead of simulation relation. This corresponds to the so-called *orchestration problem*, where the issue is to find a communication architecture of the available services, that is equivalent to the goal, modulo bisimulation. We think that this is less natural in our simple setting, mainly due the nature of the service composition which is modeled here as a fully asynchronous product. Bisimulation requirement would mean also that the client should be prepared to admit all the interleavings possible in the composition, which usually makes the specification of the client's goal too complex. A result that is closely related to ours is the EXPTIME completeness of the simulation and bisimulation problems between non-flat systems [10]. The main difference to our setting is that both system and services are given as composition of finite state machines using (binary) synchronization on actions, i.e., an action can synchronize two services. In a sense this paper shows that the lower bound for the simulation relation holds even without any synchronization.

2 Notations

An *asynchronous* product of n deterministic automata

$$\mathcal{A}_i = \langle Q_i, \Sigma_i, q_i^0, \delta_i : Q_i \times \Sigma_i \to Q_i \rangle$$

is a nondeterministic automaton:

$$\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n = \langle Q, \Sigma, \boldsymbol{q}, \delta : Q \times \Sigma \to \mathcal{P}(Q) \rangle$$

where: $Q = Q_1 \times \cdots \times Q_n$; $\Sigma = \bigcup_{i=1,\ldots,n} \Sigma_i$; $\boldsymbol{q} = (q_1^0, \ldots, q_n^0)$; and δ is defined by:

> $\boldsymbol{t} \in \delta(\boldsymbol{s}, a)$ iff for some i, $t_i = \delta_i(s_i, a)$ and for all $j \neq i$ we have $t_j = s_j$.

Observe that the product automaton can be non deterministic because the alphabets Σ_i are not necessarily disjoint.

We define a *simulation relation* on nondeterministic automata in a standard way. Take two nondeterministic automata $\mathcal{A} = \langle Q_A, \Sigma, q_A^0, \delta_A : Q_A \times \Sigma \to \mathcal{P}(Q_A) \rangle$ and $\mathcal{B} = \langle Q_B, \Sigma, q_B^0, \delta_B : Q_B \times \Sigma \to \mathcal{P}(Q_B) \rangle$ over the same alphabet. The simulation relation $\preccurlyeq \subseteq Q_A \times Q_B$ is the biggest relation such that if $q_A \preccurlyeq q_B$ then for every $a \in \Sigma$ and every $q'_A \in \delta_A(q_A, a)$ there is $q'_B \in \delta_B(q_B, a)$ such that $q'_A \preccurlyeq q'_B$. We write $\mathcal{A} \preccurlyeq \mathcal{B}$ if $q_A^0 \preccurlyeq q_B^0$.

Problem: Given n deterministic automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and a deterministic automaton $\mathcal{B}$ decide if $\mathcal{B} \preccurlyeq \mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$.

We will show that this problem is EXPTIME-complete. It is clearly in EXPTIME as one can construct the product $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ explicitly and calculate the biggest simulation relation with $\mathcal{B}$. The rest of this paper will contain the proof of EXPTIME-hardness. We will start with the PSPACE-hardness, as this will allow us to introduce the method and some notation.

3 PSPACE-hardness

We will show PSPACE-hardness of the problem by reducing it to the existence of a looping computation of a linearly space bounded deterministic Turing machine. The presented proof of the PSPACE bound has the advantage to generalize to the encoding of alternating machines that we will present in the following section.

Fix a deterministic Turing machine M working in space bounded by the size of its input. We want to decide if on a given input the computation of the machine loops. Thus we do not need any accepting states in the machine and we can assume that there are no transitions from rejecting states. We denote by Q the states of M and by Γ the tape alphabet of M. A *configuration* of M is a word over $\Gamma \cup (Q \times \Gamma)$ with exactly one occurrence of a letter from $Q \times \Gamma$. A configuration is of size of n if it is a word of length n. Transitions of M will be denoted as $qa \longrightarrow q'bd$, where q, q' are the old/new state, a, b the old/new tape symbol and $d \in \{l, r\}$ the head move.

Suppose that the input is a word w of size n. We will construct automata $\mathcal{A}_1, \dots, \mathcal{A}_n$ and $\mathcal{B}$ such that $\mathcal{B} \preccurlyeq \mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ iff the computation of M on w is infinite.

We start with some auxiliary alphabets. For every $i = 1, \dots, n$ let

$$\Gamma_i = \Gamma \times \{i\} \quad \text{and} \quad \Delta_i = (Q \times \Gamma_i) \cup (Q \times \Gamma_i \times \{l, r\}) .$$

We will write a_i instead of (a, i) for elements of Γ_i. Let also $\Delta = \bigcup_{i=1,\dots,n} \Delta_i$.

The automaton $\mathcal{A}_i = \langle Q_i, \Sigma_i, q_i^0, \longrightarrow \rangle$ is defined as follows:

- The set of states is $Q_i = \Gamma \cup (Q \times \Gamma) \cup \{\top\}$, and the alphabet of the automaton is $\Sigma_i = \Delta$.
- We have transitions:
  - $a \xrightarrow{qa_i} qa$, for all $a \in \Gamma$ and $q \in Q$,
  - $qa \xrightarrow{q'b_id} b$, for $qa \to q'bd$ the transition of M on qa (there is at most one).
  - From a, transitions on letters in $\Delta_i \setminus \{qa_i : q \in Q\}$ go to $\top$. Similarly, from qa transitions on $\Delta_i \setminus \{qb_id\}$ go to $\top$ if there is a transition of M on qa; if not, then qa has no outgoing transitions. From $\top$ there are self-loops on all letters from Δ.

– For $i = 2, \ldots, n$ the initial state of $\mathcal{A}_i$ is w_i, the i-th letter of w; for $\mathcal{A}_1$ the initial state is $q^0 w_1$, i.e., the initial state of M and the first letter of w.

Figure 1 shows a part of $\mathcal{A}_i$:

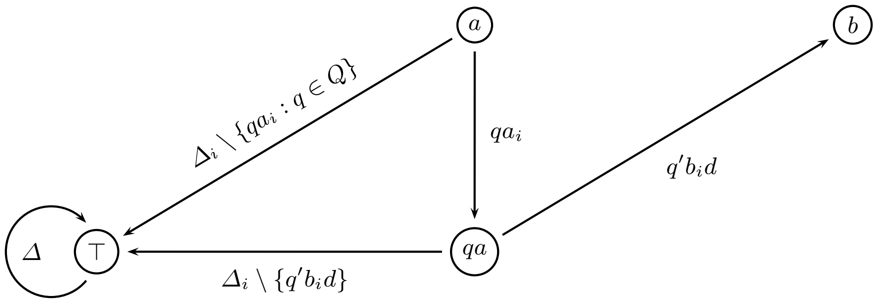

Fig. 1. Part of $\mathcal{A}_i$

The idea is classical: automaton $\mathcal{A}_i$ controls the i-th tape symbol, whereas automaton $\mathcal{B}$ defined below is in charge of the control part of M. The challenge is to do this without using any synchronization between adjacent automata $\mathcal{A}_i, \mathcal{A}_{i+1}$. Next, we introduce an automaton K that will be then used to define $\mathcal{B}$. The set of states of K is $Q_K = \{s, e\} \cup (Q \times \bigcup \Gamma_i \times \{l, r\})$; the initial state is s and the final one e; the alphabet is Δ; the transitions are defined by:

– $s \xrightarrow{q'b_ir} q'b_ir$ for $i = 1, \ldots, n-1$, whenever we have a transition $qa \to q'br$ in M for some state q and some letter a;
– $s \xrightarrow{q'b_{i+1}l} q'b_{i+1}l$ for $i = 2, \ldots, n$, whenever we have a transition $qa \to qbl$ in M for some state q and some letter a;
– $q'b_ir \xrightarrow{q'c_{i+1}} e$ and $q'b_{i+1}l \xrightarrow{q'c_i} e$ for all $c \in \Gamma$.

We define $\mathcal{B}$ as the minimal deterministic automaton recognizing $(L(K))^*$. In other words, $\mathcal{B}$ is obtained by gluing together states s and e. Figure 2 is a schema of the automaton K.

Remark 1. All $\mathcal{A}_i$ and $\mathcal{B}$ are deterministic automata of size polynomial in n. The input alphabets of the $\mathcal{A}_i$ are almost pairwise disjoint: the only states with common labels on outgoing transitions are the $\top$ states.

Definition 1. *We say that a configuration C of size n of M* corresponds *to a global state $\boldsymbol{s}$ of $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ iff $s_i = C(i)$ for $i = 1, \ldots, n$; in other words, if the state of $\mathcal{A}_i$ is the same as the i-th letter of C.*

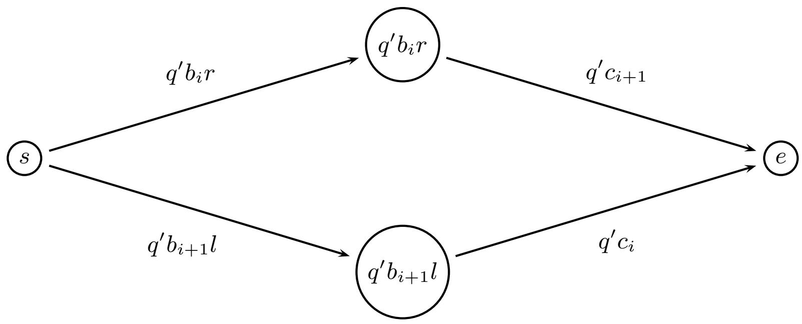

Fig. 2. Automaton K

Definition 2. *We say that a global state $\boldsymbol{s}$ of $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ is* proper *when there is no $\top$-state in $\boldsymbol{s}$.*

Lemma 1. *If $\boldsymbol{s}$ is a proper state then for every letter $a \in \Delta$ there is at most one transition of $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ from $\boldsymbol{s}$ on a. Once the automaton enters a state that is not proper it stays in non proper states.*

It is easy to see that from a non proper state, $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ can simulate any state of $\mathcal{B}$. The reason is that from $\top$, any move on letters from Δ is possible.

Lemma 2. *Suppose that $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ is in a state $\boldsymbol{s}$ that corresponds to a configuration C of M.*

- *If C is a configuration with no successor, then there is a word $v \in L(K)$ that cannot be simulated by $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ from $\boldsymbol{s}$.*
- *Otherwise the successor configuration $C \vdash C'$ exists, and there is a unique word $v \in L(K)$ such that $\boldsymbol{s} \xrightarrow{v} \boldsymbol{t}$ and $\boldsymbol{t}$ is proper. Moreover $\boldsymbol{t}$ corresponds to C'. All other words from $L(K)$ lead to non proper configurations of $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$.*

Proof. For the first claim, assume that $\boldsymbol{s}$ corresponds to a configuration, thus there is exactly one i such that $\mathcal{A}_i$ is in a state from $Q \times \Gamma$. The other automata are in states from Γ.

If C is terminal then $\mathcal{A}_i$ is in a state qa which has no outgoing transition. This means that this state can simulate no move on letters $q'b_ir$, for $q' \in Q$ and $b_i \in \Gamma_i$ (and such a move exists in K, as the machine M must have a move to the right if it is nontrivial). All other automata are also not capable to simulate $q'b_ir$ as they can do only moves on letters Δ_j for $j \neq i$.

Now suppose that $C \vdash C'$. To avoid special, but simple, cases suppose that the position i of the state is neither the first nor the last. Let $s_i = qa$

and suppose also that $qa \to q'br$ is the move of M on qa. The case when the move is to the left is similar.

The only possible move of K from s which will put $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ into a proper state is $q'b_ir$. This makes $\mathcal{A}_i$ to change the state to b and it makes K to change the state to $q'b_ir$. From this latter state the only possible move of K is on letters $q'c'_{i+1}$ for arbitrary $c' \in \Gamma$. Suppose that $\mathcal{A}_{i+1}$ is in the state $c = s_{i+1} \in \Gamma$, then all moves of K on $q'c'_{i+1}$ with $c' \neq c$ can be matched with a move to $\top$ of $\mathcal{A}_{i+1}$. On $q'c_{i+1}$ the automaton $\mathcal{A}_{i+1}$ goes to $q'c$ and automaton K goes to e. This way the state in the configuration is changed and transmitted to the right. We have that the new state of $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ corresponds to the configuration C'.

Lemma 3. *We have $\mathcal{B} \preccurlyeq \mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ iff M has an infinite computation.*

Proof. Recall that $\mathcal{B}$ is the minimal deterministic automaton recognizing $(L(K))^*$, and has initial state s. The initial state of $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ corresponds to the initial configuration C_0 of M. We show that $s \preccurlyeq \boldsymbol{t}$ with $\boldsymbol{t}$ corresponding to a configuration C of M, iff the computation of M starting in C is infinite.

From a configuration C, machine M has only one computation: either infinite, or finite that is blocking. Suppose that the computation from C has at least one step and let C_1 be the successor configuration. By Lemma 2 from state s there is exactly one word $v_1 \in L(K)$ such that $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ in order to simulate it is forced to go to a proper state $\boldsymbol{t}_1$. Morover $\boldsymbol{t}_1$ corresponds to C_1. On all other words from $L(K)$, the product $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ can go to a non proper state and from there it can simulate any future behaviour of $\mathcal{B}$. If C_1 has no successor configuration then, again by Lemma 2, there is a word in $L(K)$ that cannot be simulated by $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ from $\boldsymbol{t}_1$. If C_1 has a successor then we repeat the whole argument. Thus the behaviour of $\mathcal{B}$ from s can be simulated by $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ from the state corresponding to C iff the machine M has an infinite computation starting from C.

One can note that the construction presented in this section uses actions that are common to several processes in a quite limited way: the only states that have common outgoing labels are the $\top$ states from which all behaviours are possible. This observation motivates the question of the complexity when the automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ have pairwise disjoint alphabets. With this restriction, the simulation problem can be solved efficiently:

Theorem 1. *The following question can be solved in polynomial time:*

Input: *n deterministic automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ over pairwise disjoint input alphabets, and a deterministic automaton $\mathcal{B}$.*

Output: *decide if $\mathcal{B} \preccurlyeq \mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$.*

Proof. Let $\mathcal{C}_i$ be a automaton with a single state $\top$, and with self-loops on every letter from the alphabet Σ_i of $\mathcal{A}_i$. We write $\mathcal{A}^{(i)}$ for the asynchronous product of all $\mathcal{C}_j$, $j \neq i$, and of $\mathcal{A}_i$. Similarly, $\boldsymbol{t}^{(i)}$ will denote $\boldsymbol{t}$ with all components but i replaced by $\top$. Suppose now that p is a state of $\mathcal{B}$, and $\boldsymbol{t}$ a state of $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$. We write $p \preccurlyeq_i \boldsymbol{t}$ if p is simulated by $\boldsymbol{t}^{(i)}$ in $\mathcal{A}^{(i)}$. Notice that since $\mathcal{B}$ and $\mathcal{A}_i$ are both deterministic, we can decide if $p \not\preccurlyeq_i \boldsymbol{t}$ in logarithmic space (hence in polynomial time), by guessing simultaneously a path in $\mathcal{B}$ and one in $\mathcal{A}_i$.

We show now that $p \preccurlyeq \boldsymbol{t}$ in $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ iff $p \preccurlyeq_i \boldsymbol{t}$ for all i.

If $p \preccurlyeq \boldsymbol{t}$, then all the more $p \preccurlyeq \boldsymbol{t}^{(i)}$, since $\mathcal{C}_j$ can simulate $\mathcal{A}_j$ for all $j = 1, \ldots, n$. Conversely, assume that $p \preccurlyeq_i \boldsymbol{t}$ for all i, but $p \not\preccurlyeq \boldsymbol{t}$. This means that there exist computations $p \stackrel{a_1 \ldots a_k}{\longrightarrow} p'$ in $\mathcal{B}$, $\boldsymbol{t} \stackrel{a_1 \ldots a_k}{\longrightarrow} \boldsymbol{u}$ in $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ and a letter $a \in \Sigma_i$ for some i, such that p' has an outgoing a-transition, but $\boldsymbol{u}_i$ does not (in $\mathcal{A}_i$). Clearly, we also have a computation $\boldsymbol{t}^{(i)} \stackrel{a_1 \ldots a_k}{\longrightarrow} \boldsymbol{u}^{(i)}$ in $\mathcal{A}^{(i)}$. Since $\boldsymbol{u}_i$ has no outgoing a-transition, so neither does $\boldsymbol{u}^{(i)}$, which contradicts $p \preccurlyeq_i \boldsymbol{t}$.

4 Exptime-hardness

This time we take an alternating Turing machine M working in space bounded by the size of the input. We want to decide if M has an infinite computation. This means that the machine can make choices of existential transitions in such a way that no matter what are the choices of universal transitions the machine can always continue. Clearly, one can reduce the word problem to this problem, hence it is Exptime-hard (see [6]; for more details on complexity see any standard textbook on complexity).

We will assume that M has always a choice between two transitions, i.e., for each non blocking state/symbol pair qa there will be precisely two distinct tuples $q'b'd'$, $q''b''d''$ such that $qa \to q'b'd'$ and $qa \to q''b''d''$. If q is existential then it is up to the machine to choose a move; if q is universal then the choice is made from outside. To simplify the presentation we will assume that $d' = d''$, i.e., both moves go in the same direction. Every machine can be transformed to an equivalent one with this property. We will also assume that the transitions are ordered in some way so we will be able to say that $qa \to q'b'd$ is the first transition and $qa \to q''b''d$ is the second one.

Suppose that the input word is w of size n. We will construct automata $\mathcal{A}'_1, \mathcal{A}''_1, \ldots, \mathcal{A}'_n, \mathcal{A}''_n$ and $\mathcal{B}$ such that $\mathcal{B}$ is simulated by $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ iff there is an infinite alternating computation of M on w. The main idea is that automata $\mathcal{A}'_i$ and $\mathcal{A}''_i$ control the i-th tape symbol, as in the previous section, and each one is in charge of one of the two possible transitions (if any) when the input head is at position i in an existential state (universal moves are simpler).

We will modify a little the alphabets that we use. Let

$$\begin{aligned} \Delta'_i =& (Q \times \Gamma_i) \cup (Q \times \Gamma_i \times \{l, r\} \times \{1\}) \\ \Delta''_i =& (Q \times \Gamma_i) \cup (Q \times \Gamma_i \times \{l, r\} \times \{2\}) \end{aligned}$$

We then put $\Delta_i = \Delta'_i \cup \Delta''_i$, $\Delta = \bigcup_i \Delta_i$, $\Delta' = \bigcup_i \Delta'_i$ and $\Delta'' = \bigcup_i \Delta''_i$.

The automaton $\mathcal{A}'_i$ is defined as follows:

- The set of states is $Q'_i = \{\top\} \cup \Gamma \cup (Q \times \Gamma) \cup (Q \times \Gamma \times \{l, r\})$, the alphabet of the automaton is $\Sigma'_i = \Delta \cup \{\zeta\}$; where ζ is a new letter common to all automata.
- We have the following transitions:
  - $a \xrightarrow{qa_i} qa$ for all $a \in \Gamma$ and $q \in Q$,
  - $qa \xrightarrow{q'b'_id1} b'$ and $qa \xrightarrow{q''b''_id1} b''$ if q is an universal state and $qa \to q'b'd$, $qa \to q''b''d$ are the two transitions from qa. We have also transitions to $\top$ on all the letters from $\Delta'_i \setminus \{q'b'_id1, q''b''_id1\}$.
  - $qa \xrightarrow{\zeta} q'b'd \xrightarrow{q'b'_id1} b'$ and $qa \xrightarrow{q''b''_id1} b''$ if q is an existential state and $qa \to q'b'd$, $qa \to q''b''d$ are the first and the second transitions from qa, respectively. We have also transitions to $\top$ on all the letters from $\Delta'_i \setminus \{q'b'_id1, q''b''_id1\}$. From $q'b'd$ all transitions on $\Delta'_i \setminus \{q'b'_id1\}$ go to $\top$.
  - From a, transitions on letters in $\Delta'_i \setminus \{qa_i : q \in Q\}$ go to $\top$. If qa is terminal then there are no outgoing transitions from qa. From $\top$ there are self-loops on all letters from $\Delta^c := \Delta \cup \{\zeta\}$.
- The initial state of $\mathcal{A}'_i$ is w_i, the i-th letter of w except for $\mathcal{A}_1$ whose initial state is q^0w_1, the initial state of M and the first letter of w.

Figure 3 below presents parts of $\mathcal{A}'_i$ corresponding to universal and existential states.

The automaton $\mathcal{A}''_i$ is the same as $\mathcal{A}'_i$ with the difference that we have $q'b'd2$ instead of $q''b''d1$, $q''b''d2$ instead of $q'b'd1$ (notice the change of primes and double primes), and Δ'' instead of Δ'.

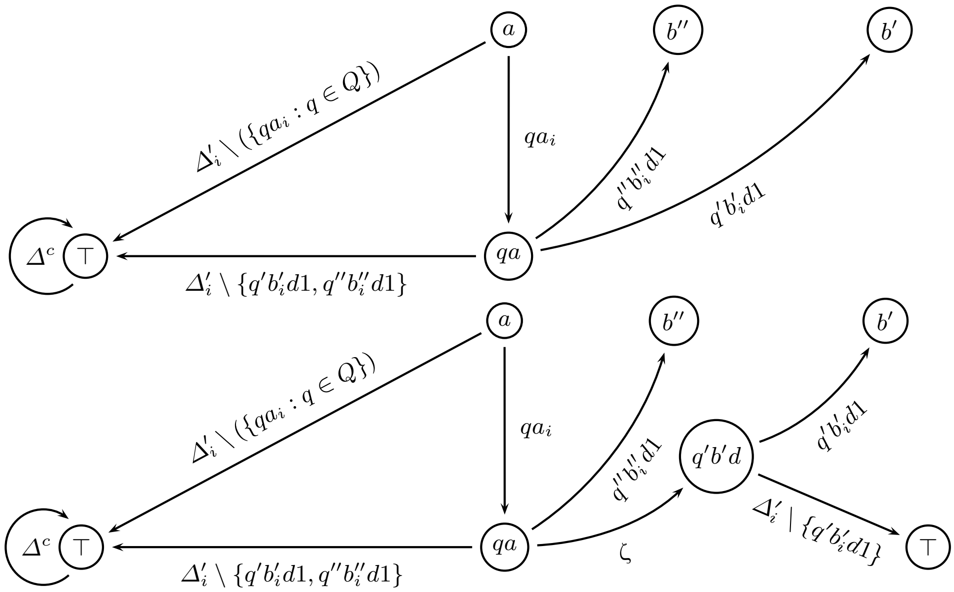

Fig. 3. Parts of the automaton $\mathcal{A}'_i$ corresponding to universal and existential states q, respectively. The alphabet Δ^c is $\Delta \cup \{\zeta\}$.

Next, we define a new automaton K that will be used to define new automaton $\mathcal{B}$. The states of K are

$$Q_K = \{s, e, \mathit{choice}\} \cup (Q \times \bigcup_i \Gamma_i \times \{l, r\})$$

plus some auxiliary states to implement transitions on two letters at a time. We will write transitions with two letters on them for readability. The initial state is s and the final one is e. The alphabet is $\Sigma_K = \bigcup \Sigma_i$. The transitions are defined by (cf. Figure 4):

- $s \xrightarrow{\zeta} \mathit{choice}$;
- $s \xrightarrow{(q'b_ir1)(q'b_ir2)} q'b_ir$ whenever we have a transition $qa \to q'br$ in M for some universal state q and some letter a, and similarly from *choice* instead of s when q is existential;
- $s \xrightarrow{(q'b_{i+1}l1)(q'b_{i+1}l2)} q'b_{i+1}l$ whenever we have a transition $qa \to q'bl$ in M for some universal state q and some letter a, and similarly from *choice* instead of s when q is existential;
- $q'b_ir \xrightarrow{(q'c_{i+1})^2} e$ and $q'b_{i+1}l \xrightarrow{(q'c_i)^2} e$ for all $c \in \Gamma$.

We define $\mathcal{B}$ as the minimal deterministic automaton recognizing $(L(K))^*$. It is obtained by gluing together states s and e.

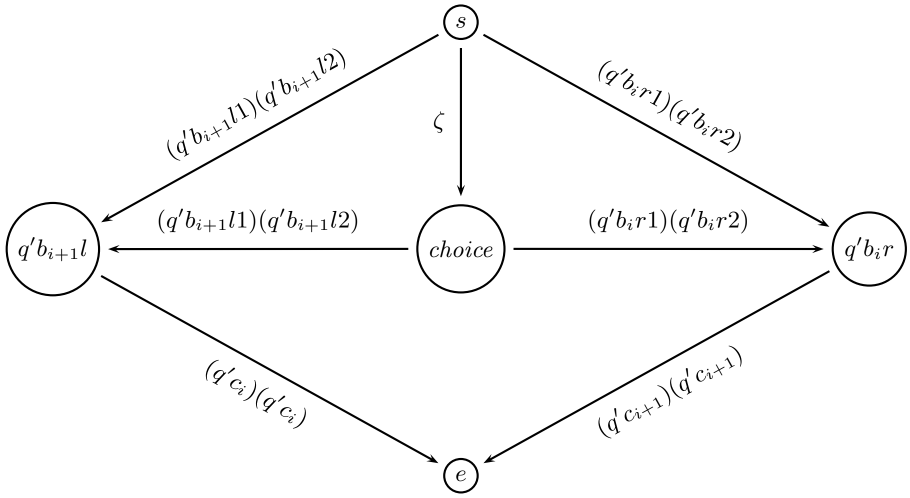

Fig. 4. Automaton K

Remark 2. All $\mathcal{A}'_i$, $\mathcal{A}''_i$ and $\mathcal{B}$ are deterministic and of size polynomial in n.

Definition 3. *A configuration C of size n* corresponds *to a global state $\boldsymbol{s}$ of $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ if $s_{2i} = s_{2i-1} = C(i)$ for $i = 1, \ldots, n$; in other words, if the states of $\mathcal{A}'_i$ and $\mathcal{A}''_i$ are the same as the i-th letter of C.*

Definition 4. *We say that a global state $\boldsymbol{s}$ of $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ is* proper *when $\top$ does not appear in $\boldsymbol{s}$.*

It is easy to see that from a non proper state, $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ can simulate any state of $\mathcal{B}$. The reason is that from $\top$, any move on letters from Δ^c is possible.

Lemma 4. *Suppose that $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ is in a state $\boldsymbol{s}$ corresponding to a configuration C of M. If C has no successor configuration then there is a word $v \in L(K)$ that cannot be simulated by $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ from $\boldsymbol{s}$. Otherwise, C has two successor configurations $C \vdash C'$ and $C \vdash C''$. We have two cases:*

- *If C is universal then there are two words v' and v'' in $L(K)$ that lead from $\boldsymbol{s}$ to proper states only, one state for v' and one for v''. These states correspond to C' and C'', respectively. On all other words from $L(K)$, non proper states can be reached from $\boldsymbol{s}$.*
- *If C is existential, then on the letter ζ the automaton $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ can reach only one of the two states $\boldsymbol{s'}$ or $\boldsymbol{s''}$. From $\boldsymbol{s'}$ there*

is a word v' such that $\zeta v' \in L(K)$ and on v' from $\boldsymbol{s'}$ the automaton $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ can reach a unique state, which moreover corresponds to C'. Similarly for $\boldsymbol{s''}$ and C''. On all words from $L(K)$ that are different from $\zeta v'$ and $\zeta v''$, non proper states can be reached from $\boldsymbol{s}$.

Proof. As $\boldsymbol{s}$ corresponds to the configuration C, there is some i such that both automata $\mathcal{A}'_i$ and $\mathcal{A}''_i$ are in state qa, for some $q \in Q$ and $a \in \Gamma$, and all other automata are in states from Γ.

If C is a configuration without successor, then the state qa in $\mathcal{A}'_i$ and $\mathcal{A}''_i$ does not have any outgoing transition. Thus these automata cannot simulate the ζ transition of K from s. No other automaton $\mathcal{A}'_j$, or $\mathcal{A}''_j$ can simulate the ζ transition either, as they are all in states from Γ.

Suppose that C is an universal configuration with two possible transitions to the right, $qa \to q'b'r$ and $qa \to q''b''r$. The case when the moves are to the left is similar. In $\mathcal{A}'_i$ from the state qa we have a transition on $q'b'_ir1$ leading to b' and on $q''b''_ir1$ leading to b''. Similarly for $\mathcal{A}''_i$, but on $q'b'_ir2$ and $q''b''_ir2$. These transitions can simulate both transitions $(q'b'_ir1)(q'b'_ir2)$ and $(q''b''_ir1)(q''b''_ir2)$ that are possible from s in K. (All other transitions from s in K lead from $\boldsymbol{s}$ to a non proper state of $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$.) Let us focus only on the first case, when $(q'b'_ir1)(q'b'_ir2)$ is executed in K and the state $q'b'_ir$ is reached. From this state only transitions $(q'c'_{i+1})^2$ are possible, for all $c' \in \Gamma$. Suppose that $\mathcal{A}'_{i+1}$ and $\mathcal{A}''_{i+1}$ are in state $c \in \Gamma$. Transition $(q'c_{i+1})^2$ of K is simulated by moves to $q'c$ in both $\mathcal{A}'_{i+1}$ and $\mathcal{A}''_{i+1}$. This way the new state is transferred to the right. Transitions $(q'c'_{i+1})^2$ where $c \neq c'$ are simulated in $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ by moves of $\mathcal{A}'_{i+1}$ and $\mathcal{A}''_{i+1}$ to $\top$.

Suppose that C is an existential configuration, with possible transitions $qa \to q'b'r$ and $qa \to q''b''r$. The case when moves are to the left is similar. Consider first the transition of K from s that corresponds to the letter ζ. Both $\mathcal{A}'_i$ and $\mathcal{A}''_i$ can simulate this transition: the first goes to state $q'b'r$, and the second goes to $q''b''r$. Assume that it is the transition of $\mathcal{A}'_i$ that is taken; the other case is symmetric. We get to the position when K is in the state *choice*, $\mathcal{A}'_i$ is in the state $q'b'r$ and $\mathcal{A}''_i$ in the state qa. From *choice*, automaton K can do $(q'b'_ir1)(q'b'_ir2)$ that can be simulated by the transitions of $\mathcal{A}'_i$ and $\mathcal{A}''_i$ (every other transition of K can be simulated by a move of $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ to a non proper state). Both automata reach the state b'. Automaton K is now in state $q'b_ir$ from where it can do $(q'c_{i+1})^2$ for any $c \in \Gamma$. The result of simulating these transitions while reaching a proper state is the transfer of the state to the right, in the same way as in the case of the universal move.

Finally, it remains to see what happens if K makes a move from s that is different from ζ. In this case, at least one of the automata $\mathcal{A}'_i$, $\mathcal{A}''_i$ can simulate the corresponding transition on $(pe_i d1)$, $(pe_i d2)$ respectively, by going to state $\top$, since we suppose that in any configuration of M, the two outgoing transitions are distinct. Hence, a non proper state can be reached.

Theorem 2. *The following problem is* EXPTIME*-complete:*
Input: *deterministic automata* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *and a deterministic automaton* $\mathcal{B}$*.*
Output: *decide if* $\mathcal{B} \preccurlyeq \mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$*.*

Proof. We use the construction presented above. By Lemma 4 we can show similarly to the previous section, that the initial state s of $\mathcal{B}$ can be simulated from a state $\boldsymbol{t}$ of $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ that corresponds to a configuration C of the alternating Turing machine M, iff M has an infinite alternating computation from C. The problem is clearly in EXPTIME as the state space of $\mathcal{A}'_1 \otimes \mathcal{A}''_1 \cdots \otimes \mathcal{A}'_n \otimes \mathcal{A}''_n$ can be constructed in EXPTIME.

We conclude the section by showing that Theorem 2 still holds under the assumption that the alphabet of the automata is of constant size.

Theorem 3. *Let Σ be a fixed alphabet of at least* 2 *letters. The following problem is* EXPTIME*-complete:*
Input: *deterministic automata* $\mathcal{A}_1, \ldots, \mathcal{A}_n$ *and a deterministic automaton* $\mathcal{B}$ *over the input alphabet* Σ*.*
Output: *decide if* $\mathcal{B} \preccurlyeq \mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$*.*

Proof. We reduce directly from Theorem 2. Suppose that the input alphabet of all automata $\mathcal{A}_i, \mathcal{B}$ is $\Sigma \times \{1, \ldots, m\}$, for some m. Moreover, let S be the set of states of $\mathcal{B}$ and let $Q = Q_1 \times \cdots \times Q_n$ be the set of global states of $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$.

In each automaton $\mathcal{A}_i$, $\mathcal{B}$ we replace every transition $s \xrightarrow{a_l} t$ by a sequence of transitions with labels from $\Sigma \cup \{\#, \$\}$ as follows:

$$s \xrightarrow{a} (stl0) \xrightarrow{\#} (stl1) \xrightarrow{\#} (stl2) \cdots \xrightarrow{\#} (stll) \xrightarrow{\$} t$$

The $(l+1)$ states $(stl0), \ldots, (stll)$ are new. Let $\mathcal{A}'_i, \mathcal{B}'$ be the automata obtained from $\mathcal{A}_i$, $\mathcal{B}$, with state space Q' and S', respectively.

Take $\preccurlyeq$, the largest simulation relation from $\mathcal{B}$ to $\mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$. We show how to extend $\preccurlyeq$ to $\preccurlyeq'$ such that $\preccurlyeq'$ is a simulation relation from $\mathcal{B}'$ to $\mathcal{A}'_1 \otimes \cdots \otimes \mathcal{A}'_n$ (not necessarily the largest one). Let $\preccurlyeq'$ be the union of $\preccurlyeq$ with the set of all pairs $((stlk), \boldsymbol{u}')$, where $s, t \in S$, $\boldsymbol{u}' = (u'_1, \ldots, u'_n) \in Q'$, and such that:

– $s \xrightarrow{a_l} t$ and $\boldsymbol{v} \xrightarrow{a_l} \boldsymbol{w}$ for some $a \in \Sigma$, $\boldsymbol{v} = (v_1, \ldots, v_n)$ and $\boldsymbol{w} = (w_1, \ldots, w_n)$ such that $s \preccurlyeq \boldsymbol{v}$, $t \preccurlyeq \boldsymbol{w}$,
– there is some i with $u'_i = (v_i w_i l k)$, and $u'_j = v_j = w_j$ for $j \neq i$.

It is immediate to check that $\preccurlyeq'$ is a simulation relation. First, (old) states from S can only be simulated by (old) states from Q. Second, a new state $(stlj)$ of $\mathcal{B}$ can be simulated only by states $\boldsymbol{u}' \in Q' \setminus Q$. It can be shown easily that the largest simulation relation from $\mathcal{B}'$ to $\mathcal{A}'_1 \otimes \cdots \otimes \mathcal{A}'_n$ coincides with $\preccurlyeq'$ (hence with $\preccurlyeq$) on the set $S \times Q$ of pairs of old states.

5 Conclusions

We have shown an Exptime lower bound for the composition of e-services that are described as a fully asynchronous product of finite state machines. Thus, we answer the question left open in [2]. Since our lower bound holds for the simplest model one can think of (no synchronization at all), it also applies to richer models, such as products with synchronization on actions as in [10] or communicating finite-state machines (CFSM) as in [9, 8]. It is easy to see that the simulation of a finite-state machine by a CFSM $\mathcal{A}$ with bounded message queues is in Exptime, since the state space of $\mathcal{A}$ is exponential in this case. Hence, this problem, as well as any of its variants with some restricted form of communication, is Exptime-complete as well.

It remains open whether the bisimulation problem for a finite automaton and a fully asynchronous product of finite automata is also Exptime-hard. Another interesting question is how far one can relax the restrictions on e-services given by communicating finite-state machines, in order to preserve decidability.

Acknowledgement: We thank the anonymous referees for interesting comments and suggestions for improvement.

References

1. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services. Concepts, Architectures and Applications.* Springer, 2004.
2. D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In *Proc. of the 1st Int. Conf. on Service Oriented Computing (ICSOC 2003)*, volume 2910 of *Lecture Notes in Computer Science*, pages 43–58. Springer, 2003.
3. D. Berardi, D. Calvanese, G. D. Giacomo, R. Hull, and M. Mecella. Automatic composition of web services in colombo. In *SEBD 2005*, pages 8–15, 2005.

4. D. Berardi, D. Calvanese, G. D. Giacomo, R. Hull, and M. Mecella. Automatic composition of web services with messaging. In *VLDB 2005*, pages 613–624, 2005.
5. D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.
6. A. K. Chandra, D. Kozen and L. J. Stockmeyer. Alternation. *J. ACM* 28(1):114–133, 1981.
7. A. Deutsch, L. Sui, and V. Vianu. Specification and verification of data-driven web services. In *Symposium on Principles of Database Systems (PODS)*, 2004.
8. X. Fu, T. Bultan, and J. Su. Conversation protocols: a formalism for specification and verification of reactive electronic services. In *Theor. Comput. Sci.* 328(1-2):19–37, 2004.
9. R. Hull, M. Benedikt, V. Christophides, J. Su. E-services: a look behind the curtain. In *Symposium on Principles of Database Systems (PODS)*, pp. 1-14, 2003.
10. F. Laroussinie and Ph. Schnoebelen. The state explosion problem from trace to bisimulation equivalence. In *FoSSaCS 2000*, pages 192–207, 2000.
11. M. Pistore, A. Marconi, P. Bertoli, and P. Traverso. Automated composition of web services by planning at the knowledge level. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 2005.