

Composition of services with constraints

Philippe Balbiani¹, Fahima Cheikh¹
Pierre-Cyrille Héam², Olga Kouchnarenko²

Institut de recherche en informatique de Toulouse¹
Laboratoire d'informatique de l'université de Franche-Comté²

Abstract. Web services composition is the interleaving of action sequences executed by elementary components in accordance with a client specification. We consider Web services as automata able to execute actions and send and receive messages. For this model, we define the composition problem and study its complexity.

1 Introduction

Service oriented computing [10] is a programming paradigm which considers services as elementary components. From these components, distributed applications are realised in accordance with a client specification. To realise some distributed applications, elementary components have to be composed. The composition problem has been investigated since the 2000's with many solutions proposed [2, 1, 9]. What is this problem? To answer, we have first to know about which kind of services we talk. Often, services are seen as finite automata. In this case, client specification is given by a finite automata which represents all computations that a client wants to be executed by services. By executing their transitions, services modify their environment and that of the client. The problem of combining services becomes that of composing automata. This is the way followed by [2, 1]. In other cases, services are able to send and to receive messages. In this case, client specification is given by a logical formula which represents goals of a client that wants to be reached by services. By communicating together, services modify their knowledge and those of their client. It is the approach considered by [9]. This paper follows the line of reasoning suggested in [3–6] and which consists in describing the semantics of services by means of automata. It particularly focuses on the following problem: given services $\mathcal{A}_1, \dots, \mathcal{A}_n$ and the request \mathcal{A} of a client, can $\mathcal{A}_1, \dots, \mathcal{A}_n$ be organised as to answer \mathcal{A} ? The originality of our approach consists in modeling the services by Boolean automata, i.e. finite automata extended with parametric Boolean conditions. The main motivation for using this model is to manage conditional actions or communications of $\mathcal{A}_1, \dots, \mathcal{A}_n$. For instance, a conditional action may be, for some $i, j \in \{1, \dots, n\}$, that \mathcal{A}_i accepts to communicate with \mathcal{A}_j if and only if \mathcal{A}_j has a security certificate given by some authority \mathcal{A}_k with $k \in \{1, \dots, n\}$. A conditional action may also be, for some $i, j \in \{1, \dots, n\}$, that \mathcal{A}_i accepts to answer \mathcal{A}_j only if the IP address of \mathcal{A}_j is in a selected area. This kind of conditions frequently appears when specifying services. As far as we know,

the composition problems studied in the literature and discussed above do not handle this kind of conditions. This paper provides a theoretical study for the problem of computing a Boolean formula ϕ which exactly characterises the conditions required for $\mathcal{A}_1, \dots, \mathcal{A}_n$ to answer the client's request. It is organised as follows. Section 2 introduces the formal background. In section 3, we formally define the valuation decision problem, the Boolean formula decision problem and the Boolean formula synthesis problem for both simulation-based relations and trace-based relations. Sections 4 and 5 contains our complexity results. Missing proofs can be found in the annex.

2 Preliminaries

Let P be a countable set of Boolean variables (with typical members denoted p, p', \dots), Σ_a be a countable set of asynchronous actions (with typical members denoted α, β, \dots) and Σ_s be a countable set of synchronous actions (with typical members denoted σ, τ, \dots). We will assume that P, Σ_a and Σ_s are disjoint.

Finite automata A finite automaton is a tuple $\mathcal{A} = (Q, E, I, F)$ where

- Q is a finite set of states,
- E is a function from $Q \times Q$ into the set of all finite subsets of $\Sigma_a \cup (\{?, !\} \times \Sigma_s) \cup \{\epsilon\}$,
- $I \subseteq Q$ is the set of initial states and $F \subseteq Q$ is the set of final states.

For all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, (x, σ) will be denoted $x\sigma$. \mathcal{A} is said to be ϵ -free if and only if (iff) E is a function from $Q \times Q$ into the set of all finite subsets of $\Sigma_a \cup (\{?, !\} \times \Sigma_s)$. We shall say that \mathcal{A} is weakly asynchronous iff E is a function from $Q \times Q$ into the set of all finite subsets of $\Sigma_a \cup \{\epsilon\}$. \mathcal{A} is said to be strongly asynchronous iff E is a function from $Q \times Q$ into the set of all finite subsets of Σ_a . If \mathcal{A} is weakly asynchronous then let E^* be the function from $Q \times Q$ into the set of all finite subsets of Σ_a such that for all $q, r \in Q$,

- for all $\alpha \in \Sigma_a$, $\alpha \in E^*(q, r)$ iff there are sequences $(q_0, q_1, \dots, q_m), (r_0, r_1, \dots, r_n) \in Q^+$ such that
 - for all positive integers i , if $i \leq m$ then $\epsilon \in E(q_{i-1}, q_i)$,
 - for all positive integers j , if $j \leq n$ then $\epsilon \in E(r_{j-1}, r_j)$,
 - $\alpha \in E(q_m, r_0)$,
 - $q_0 = q$ and $r_n = r$.

In this case, a run of \mathcal{A} on a sequence $(\alpha_1, \dots, \alpha_k) \in \Sigma_a^*$ is a sequence $(q_0, q_1, \dots, q_k) \in Q^+$ such that for all positive integers i , if $i \leq k$ then $\alpha_i \in E^*(q_{i-1}, q_i)$, $q_0 \in I$ and $q_k \in F$. Moreover, the trace of \mathcal{A} , denoted $tr(\mathcal{A})$, is the set of all sequences $(\alpha_1, \dots, \alpha_k) \in \Sigma_a^*$ such that there is a run of \mathcal{A} on $(\alpha_1, \dots, \alpha_k)$.

Boolean automata The set $\mathcal{B}(P)$ of all Boolean formulas (with typical members denoted ϕ, ψ, \dots) is defined by: $\phi ::= p \mid \perp \mid \neg\phi \mid (\phi \vee \phi)$. The other constructs are defined as usual. A valuation is a function from P into $\{0, 1\}$. Every valuation V gives rise to a function \widehat{V} from $\mathcal{B}(P)$ into $\{0, 1\}$ in the usual way. A Boolean automaton is a tuple $\mathcal{A} = (Q, E, I, F)$ where

- Q is a finite set of states,
- E is a function from $Q \times Q$ into the set of all finite subsets of $\mathcal{B}(P) \times (\Sigma_a \cup \{?, !\} \times \Sigma_s) \cup \{\epsilon\}$,
- $I \subseteq Q$ is the set of initial states and $F \subseteq Q$ is the set of final states.

The notions of ϵ -freeness, weak asynchronicity and strong asynchronicity are defined for Boolean automata in the same way as they are defined for finite automata. As example, take the case of the Boolean automata \mathcal{A}_1 and \mathcal{A}_2 from figure 1.

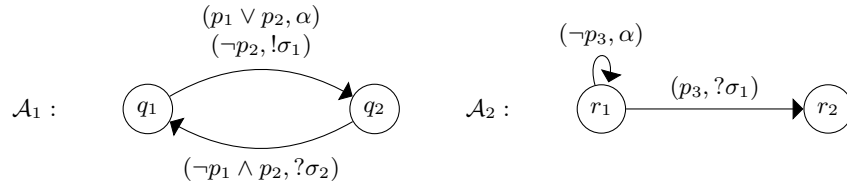


Fig. 1. Boolean automata \mathcal{A}_1 and \mathcal{A}_2 .

Synchronization Let $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ be ϵ -free Boolean automata. Their synchronization, denoted $\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n$, is the weakly asynchronous Boolean automaton $\mathcal{A} = (Q, E, I, F)$ defined by:

- $Q = Q_1 \times \dots \times Q_n$,
- E is the function from $Q \times Q$ into the set of all finite subsets of $\mathcal{B}(P) \times (\Sigma_a \cup \{\epsilon\})$ such that for all $\mathbf{q}, \mathbf{r} \in Q$, for all $\phi \in \mathcal{B}(P)$,
 - for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \in E(\mathbf{q}, \mathbf{r})$ iff there is $i \in \{1, \dots, n\}$ such that $\mathbf{q} \equiv_i \mathbf{r}$ and $(\phi, \alpha) \in E_i(q_i, r_i)$,
 - $(\phi, \epsilon) \in E(\mathbf{q}, \mathbf{r})$ iff there are $i_1, i_2 \in \{1, \dots, n\}$, there are $\phi_{i_1}, \phi_{i_2} \in \mathcal{B}(P)$, there is $\sigma \in \Sigma_s$ such that $i_1 \neq i_2$, $\mathbf{q} \equiv_{i_1, i_2} \mathbf{r}$, either $(\phi_{i_1}, (?), \sigma) \in E_{i_1}(q_{i_1}, r_{i_1})$ and $(\phi_{i_2}, (!), \sigma) \in E_{i_2}(q_{i_2}, r_{i_2})$ or $(\phi_{i_1}, (!), \sigma) \in E_{i_1}(q_{i_1}, r_{i_1})$ and $(\phi_{i_2}, (?), \sigma) \in E_{i_2}(q_{i_2}, r_{i_2})$ and $\phi = \phi_{i_1} \wedge \phi_{i_2}$.
- $I = I_1 \times \dots \times I_n$ and $F = F_1 \times \dots \times F_n$,

the binary relations $\equiv_i, \equiv_{i_1, i_2} \subseteq Q \times Q$ being such that for all $\mathbf{q}, \mathbf{r} \in Q$, $\mathbf{q} \equiv_i \mathbf{r}$ iff for all $j \in \{1, \dots, n\}$, if $i \neq j$ then $q_j = r_j$ and $\mathbf{q} \equiv_{i_1, i_2} \mathbf{r}$ iff for all $j \in \{1, \dots, n\}$, if $i_1 \neq j$ and $i_2 \neq j$ then $q_j = r_j$. Consider, as example, the Boolean automata \mathcal{A}_1 and \mathcal{A}_2 from figure 1 and $\mathcal{A}_1 \otimes \mathcal{A}_2$ from figure 2.

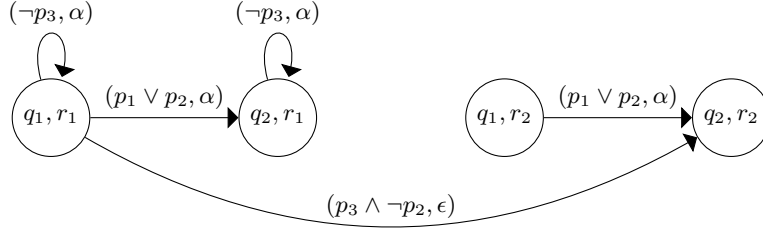


Fig. 2. Boolean automata $\mathcal{A}_1 \otimes \mathcal{A}_2$.

From Boolean automata to finite automata Let $\mathcal{A} = (Q, E, I, F)$ be a Boolean automaton and V be a valuation. The interpretation of \mathcal{A} through V , denoted \mathcal{A}^V , is the finite automaton $\mathcal{A}' = (Q', E', I', F')$ defined by:

- $Q' = Q$,
- E' is the function from $Q' \times Q'$ into the set of all finite subsets of $\Sigma_a \cup (\{?, !\} \times \Sigma_s) \cup \{\epsilon\}$ such that for all $q, r \in Q'$,
 - for all $\alpha \in \Sigma_a$, $\alpha \in E'(q, r)$ iff there is $\phi \in \mathcal{B}(P)$ such that $(\phi, \alpha) \in E(q, r)$ and $\widehat{V}(\phi) = 1$,
 - for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $x\sigma \in E'(q, r)$ iff there is $\phi \in \mathcal{B}(P)$ such that $(\phi, x\sigma) \in E(q, r)$ and $\widehat{V}(\phi) = 1$,
 - $\epsilon \in E'(q, r)$ iff there is $\phi \in \mathcal{B}(P)$ such that $(\phi, \epsilon) \in E(q, r)$ and $\widehat{V}(\phi) = 1$,
- $I' = I$ and $F' = F$.

As example, take the case of \mathcal{A} and \mathcal{A}^V from figure 3.

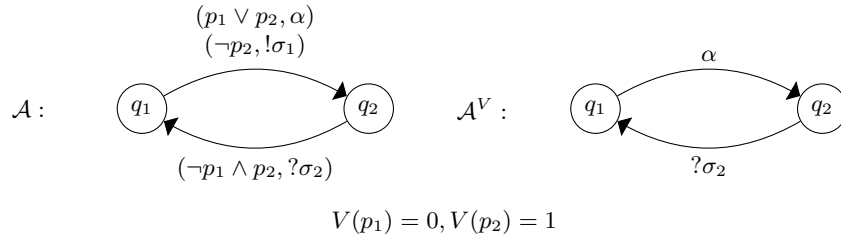


Fig. 3. Boolean automaton \mathcal{A} and the associated automaton \mathcal{A}^V .

Trace inclusion and trace equivalence Let $\mathcal{A} = (Q, E, I, F)$, $\mathcal{A}' = (Q', E', I', F')$ be weakly asynchronous finite automata. We shall say that \mathcal{A} is trace-included in \mathcal{A}' , denoted $\mathcal{A} \sqsubseteq \mathcal{A}'$, iff $tr(\mathcal{A}) \subseteq tr(\mathcal{A}')$. \mathcal{A} is said to be trace-equivalent to \mathcal{A}' , denoted $\mathcal{A} \equiv \mathcal{A}'$, iff $\mathcal{A} \sqsubseteq \mathcal{A}'$ and $\mathcal{A}' \sqsubseteq \mathcal{A}$.

Simulation and bisimulation Let $\mathcal{A} = (Q, E, I, F)$, $\mathcal{A}' = (Q', E', I', F')$ be weakly asynchronous finite automata. We define a binary relation $Z \subseteq Q \times Q'$ such that $\text{dom}(Z) \cap I \neq \emptyset$ and $\text{ran}(Z) \cap I' \neq \emptyset$ to be a simulation of \mathcal{A} by \mathcal{A}' , denoted $Z: \mathcal{A} \longleftarrow \mathcal{A}'$, iff for all $q \in Q$, for all $q' \in Q'$, if $q Z q'$ then

- for all $\alpha \in \Sigma_a$, for all $r \in Q$, if $\alpha \in E^*(q, r)$ then there is $r' \in Q'$ such that $r Z r'$ and $\alpha \in E'^*(q', r')$,
- if $q \in I$ then $q' \in I'$ and if $q \in F$ then $q' \in F'$.

Note: $\text{dom}(Z)$ and $\text{ran}(Z)$ respectively denote the domain of Z and the range of Z . If there is a simulation Z of \mathcal{A} by \mathcal{A}' then we write $\mathcal{A} \longleftarrow \mathcal{A}'$. We define a binary relation $Z \subseteq Q \times Q'$ to be a bisimulation between \mathcal{A} and \mathcal{A}' , denoted $Z: \mathcal{A} \longleftrightarrow \mathcal{A}'$, iff $Z: \mathcal{A} \longleftarrow \mathcal{A}'$ and $Z^{-1}: \mathcal{A}' \longleftarrow \mathcal{A}$. If there is a bisimulation between \mathcal{A} and \mathcal{A}' then we write $\mathcal{A} \longleftrightarrow \mathcal{A}'$.

3 Composition of services

Let $R \in \{\sqsubseteq, \equiv, \longleftarrow, \longleftrightarrow\}$. The valuation decision (VD) problem for R is defined by:

- input: a strongly asynchronous finite automaton $\mathcal{A} = (Q, E, I, F)$ and ϵ -free Boolean automata $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$,
- output: check if there is a valuation V such that $\mathcal{A} R (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$.

The Boolean formula decision (BFD) problem for R is defined by:

- input: a strongly asynchronous finite automaton $\mathcal{A} = (Q, E, I, F)$, ϵ -free Boolean automata $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ and a Boolean formula ϕ ,
- output: check if for all valuations V , $\mathcal{A} R (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$.

The Boolean formula synthesis (BFS) problem for R is defined by:

- input: a strongly asynchronous finite automaton $\mathcal{A} = (Q, E, I, F)$ and ϵ -free Boolean automata $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$,
- output: find out a Boolean formula ϕ such that for all valuations V , $\mathcal{A} R (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$.

4 Trace inclusion and trace equivalence

Upper bound Let $\mathcal{A} = (Q, E, I, F)$ be a strongly asynchronous finite automaton and $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ be ϵ -free Boolean automata. We now define a deterministic algorithm which returns the value “accept” iff there is a valuation V such that $\mathcal{A} \sqsubseteq (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$:

1. for all valuations V
 - (a) compute $(\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$;
 - (b) check if $\mathcal{A} \sqsubseteq (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$;

2. if one of these calls returns the value “accept” then return the value “accept”
else return the value “reject”;

Obviously, the deterministic algorithm above is exponential-space-bounded. Similarly, an exponential-space-bounded deterministic algorithm which returns the value “accept” iff there is a valuation V such that $\mathcal{A} \equiv (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ can be defined. As a result,

Proposition 1. *Let $R \in \{\sqsubseteq, \equiv\}$. The VD problem for R is in $EXPSPACE$.*

Similarly,

Proposition 2. *Let $R \in \{\sqsubseteq, \equiv\}$. The BFD problem for R is in $EXPSPACE$.*

Let $\mathcal{A} = (Q, E, I, F)$ be a strongly asynchronous finite automaton and $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ be ϵ -free Boolean automata. We now define a deterministic algorithm which returns a Boolean formula ϕ such that for all valuations V , $\mathcal{A} \sqsubseteq (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$:

1. $\phi := \perp$;
2. for all maximal consistent conjunctions ψ of P -literals
 - (a) compute the unique valuation V such that $\widehat{V}(\psi) = 1$;
 - (b) compute $(\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$;
 - (c) if $\mathcal{A} \sqsubseteq (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ then $\phi := \phi \vee \psi$;

Obviously, the deterministic algorithm above is exponential-space-bounded. Similarly, an exponential-space-bounded deterministic algorithm which returns a Boolean formula ϕ such that for all valuations V , $\mathcal{A} \equiv (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$ can be defined. As a result,

Proposition 3. *Let $R \in \{\sqsubseteq, \equiv\}$. The BFS problem for R is solvable by means of an exponential-space-bounded deterministic algorithm.*

Lower bound By giving a polynomial time reduction of the equivalence problem for regular expressions with squaring, which is known to be $EXPSPACE$ -hard [7], to the VD problem for \sqsubseteq , we prove that the VD problem for \sqsubseteq is $EXPSPACE$ -hard. The set of all regular expressions with squaring (with typical members denoted exp, exp', \dots) is defined by:

$$- exp ::= \alpha \mid \epsilon \mid (exp \circ exp) \mid (exp \cup exp) \mid exp^+ \mid exp^2.$$

The number of occurrences of operators $\epsilon, \circ, \cup, ^+$ and 2 in regular expression exp with squaring is denoted $op(exp)$. Every regular expression exp with squaring gives rise to a language denoted $lang(exp)$ in the usual way. Let $\sigma_1, \sigma_2, \dots$ be an enumeration of Σ_s . Given a regular expression exp with squaring, let $n = 2 \times op(exp)$. Let $\mathcal{A} = (Q, E, I, F)$ be the strongly asynchronous finite automaton defined as follows: $Q = \{q\}$, E is the function from $Q \times Q$ into the set of all finite subsets of $\mathcal{B}(P) \times \Sigma_a$ such that for all $q, r \in Q$, for all $\phi \in \mathcal{B}(P)$, for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \in E(q, r)$ iff $\phi = \top$, $I = \{q\}$ and $F = \{q\}$. For all positive integers i , if $i \leq n$ then let $\mathcal{A}_i = (Q_i, E_i, I_i, F_i)$ be the ϵ -free Boolean automaton defined as follows: $Q_i = \{q_{1_i}, q_{2_i}\}$, E_i is the function from $Q_i \times Q_i$ into the set of all finite subsets of $\mathcal{B}(P) \times (\Sigma_a \cup (\{?, !\} \times \Sigma_s))$ such that for all $q, r \in Q_i$, for all $\phi \in \mathcal{B}(P)$,

- for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \notin E_i(q, r)$,
- for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \in E_i(q, r)$ iff $\phi = \top$, $x = ?$, $\sigma = \sigma_i$, $q = q_{1_i}$ and $r = q_{2_i}$ or $\phi = \top$, $x = !$, $\sigma = \sigma_i$, $q = q_{2_i}$ and $r = q_{1_i}$,

$I_i = \{q_{1_i}\}$ and $F_i = \{q_{1_i}\}$. Let $\mathcal{A}_0 = (Q_0, E_0, I_0, F_0)$ be the ϵ -free Boolean automaton defined by induction on exp as follows:

Basis: Case $exp = \alpha$. In this case, $\mathcal{A}_0 = (Q_0, E_0, I_0, F_0)$ is defined as follows:

$Q_0 = \{q_I, q_F\}$, E_0 is the function from $Q_0 \times Q_0$ into the set of all finite subsets of $\mathcal{B}(P) \times (\Sigma_a \cup (\{?, !\} \times \Sigma_s))$ such that for all $q, r \in Q_0$, for all $\phi \in \mathcal{B}(P)$,

- for all $\beta \in \Sigma_a$, $(\phi, \beta) \in E_0(q, r)$ iff $q = q_I$, $r = q_F$, $\phi = \top$ and $\beta = \alpha$,
- for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \notin E_0(q, r)$,

$I_0 = \{q_I\}$ and $F_0 = \{q_F\}$. The Boolean automaton \mathcal{A}_0 is represented in figure 4. The reader may easily verify that for all valuations V , $lang(exp) = tr(\mathcal{A}_0^V)$. Remark that $0 = n$. Note also that I_0 and F_0 are singletons.

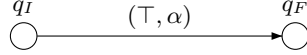


Fig. 4. Finite automaton \mathcal{A}_0 in the case $exp = \alpha$.

Hypothesis: exp' and exp'' are regular expressions with squaring such that there is an ϵ -free Boolean automaton $\mathcal{A}'_0 = (Q'_0, E'_0, I'_0, F'_0)$ such that for all valuations V , $lang(exp') = tr((\mathcal{A}'_0 \otimes \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_{n'})^V)$ where $n' = 2 \times op(exp')$ and there is an ϵ -free Boolean automaton $\mathcal{A}''_0 = (Q''_0, E''_0, I''_0, F''_0)$ such that for all valuations V , $lang(exp'') = tr((\mathcal{A}''_0 \otimes \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_{n''})^V)$ where $n'' = 2 \times op(exp'')$. We also assume that I'_0, F'_0, I''_0 and F''_0 are singletons.

Step: The cases $exp = \epsilon$, $exp = exp' \circ exp''$, $exp = exp' \cup exp''$ and $exp = exp'^+$ are considered in the annex.

Case $exp = exp'^2$. In this case, $\mathcal{A}_0 = (Q_0, E_0, I_0, F_0)$ is defined as follows: $Q_0 = Q'_0 \cup \{q_I, q, r, q_F\}$, E_0 is the function from $Q_0 \times Q_0$ into the set of all finite subsets of $\mathcal{B}(P) \times (\Sigma_a \cup (\{?, !\} \times \Sigma_s))$ such that for all $s, t \in Q_0$, for all $\phi \in \mathcal{B}(P)$,

- for all $\beta \in \Sigma_a$, $(\phi, \beta) \in E_0(s, t)$ iff $s, t \in Q'_0$ and $(\phi, \beta) \in E'_0(s, t)$,
- for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \in E_0(s, t)$ iff $s, t \in Q'_0$ and $(\phi, x\sigma) \in E'_0(s, t)$ or $s = q_I$, $t = q'_I$, $\phi = \top$, $x = !$ and $\sigma = \sigma_{n'+1}$ or $s = q'_F$, $t = q$, $\phi = \top$, $x = !$ and $\sigma = \sigma_{n'+2}$ or $s = q$, $t = q_I$, $\phi = \top$, $x = ?$ and $\sigma = \sigma_{n'+1}$ or $s = q'_F$, $t = r$, $\phi = \top$, $x = ?$ and $\sigma = \sigma_{n'+2}$ or $s = r$, $t = q_F$, $\phi = \top$, $x = ?$ and $\sigma = \sigma_{n'+1}$,

$I_0 = \{q_I\}$ and $F_0 = \{q_F\}$. The Boolean automaton \mathcal{A}_0 is represented in figure 5. The reader may easily verify that for all valuations V , $\text{lang}(exp) = \text{tr}((\mathcal{A}_0 \otimes \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_{n'} \otimes \mathcal{A}_{n'+1} \otimes \mathcal{A}_{n'+2})^V)$. Remark that $n' + 2 = n$. Note also that I_0 and F_0 are singletons.

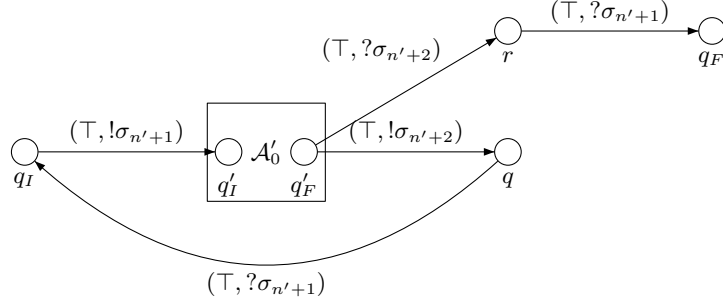


Fig. 5. Finite automaton \mathcal{A}_0 in the case $exp = exp'^2$.

The reader may easily verify that $\text{lang}(exp) = \Sigma_a^*$ iff there is a valuation V such that $\mathcal{A} \sqsubseteq (\mathcal{A}_0 \otimes \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$. Similarly, the reader may easily verify that $\text{lang}(exp) = \Sigma_a^*$ iff there is a valuation V such that $\mathcal{A} \equiv (\mathcal{A}_0 \otimes \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$. As a result,

Proposition 4. *Let $R \in \{\sqsubseteq, \equiv\}$. The VD problem for R is EXPSPACE-hard.*

Similarly,

Proposition 5. *Let $R \in \{\sqsubseteq, \equiv\}$. The BFD problem for R is EXPSPACE-hard.*

According to Meyer and Stockmeyer [7], there is a constant $c_0 > 1$ such that for all deterministic Turing machines M with space bound $n \mapsto c_0^n$, M cannot solve the equivalence problem for regular expressions with squaring. Suppose that for all constants $c > 1$, there is a deterministic algorithm f_c with space bound $n \mapsto c^n$ such that f_c can solve the BFS problem for R and let M_c be the deterministic Turing machine that behaves as follows given a regular expression exp with squaring:

1. M_c computes $n = 2 \times \text{op}(exp)$;
2. M_c computes the strongly asynchronous finite automaton $\mathcal{A} = (Q, E, I, F)$ defined as above;

3. for all positive integers i , if $i \leq n$ then M_c computes the ϵ -free Boolean automaton $\mathcal{A}_i = (Q_i, E_i, I_i, F_i)$ defined as above;
4. M_c computes the ϵ -free Boolean automaton $\mathcal{A}_0 = (Q_0, E_0, I_0, F_0)$ defined by induction on exp as above;
5. M_c simulates f_c on input \mathcal{A} and \mathcal{A}_0 and $\mathcal{A}_1, \dots, \mathcal{A}_n$ until it is about to return a value ϕ_c ;
6. if $\models \phi_c$ then return the value “accept” else return the value “reject”;

Obviously, M_{c_0} is a deterministic Turing machine with space bound $n \mapsto c_0^n$ and solving the equivalence problem for regular expressions with squaring: a contradiction. As a result,

Proposition 6. *Let $R \in \{\sqsubseteq, \equiv\}$. There is a constant $c_0 > 1$ such that for all deterministic algorithms f with space bound $n \mapsto c_0^n$, f cannot solve the BFS problem for R .*

5 Simulation and bisimulation

Upper bound Let $\mathcal{A} = (Q, E, I, F)$ be a strongly asynchronous finite automaton and $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ be ϵ -free Boolean automata. We now define a deterministic algorithm which returns the value “accept” iff there is a valuation V such that $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$:

1. for all valuations V
 - (a) compute $(\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$;
 - (b) check if $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$;
2. if one of these calls returns the value “accept” then return the value “accept” else return the value “reject”;

Obviously, the deterministic algorithm above is exponential-time-bounded. Similarly, an exponential-time-bounded deterministic algorithm which returns the value “accept” iff there is a valuation V such that $\mathcal{A} \longleftrightarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ can be defined. As a result,

Proposition 7. *Let $R \in \{\longleftarrow, \longleftrightarrow\}$. The VD problem for \longleftrightarrow is in EXP-TIME.*

Similarly,

Proposition 8. *Let $R \in \{\longleftarrow, \longleftrightarrow\}$. The BFD problem for R is in EXP-TIME.*

Let $\mathcal{A} = (Q, E, I, F)$ be a strongly asynchronous finite automaton and $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ be ϵ -free Boolean automata. We now define a deterministic algorithm which returns a Boolean formula ϕ such that for all valuations V , $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$:

1. $\phi := \perp$;
2. for all maximal consistent conjunctions ψ of P -literals

- (a) compute the unique valuation V such that $\widehat{V}(\psi) = 1$;
- (b) compute $(\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$;
- (c) if $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ then $\phi := \phi \vee \psi$;

Obviously, the deterministic algorithm above is exponential-time-bounded. Similarly, an exponential-time-bounded deterministic algorithm which returns a Boolean formula ϕ such that for all valuations V , $\mathcal{A} \longleftrightarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$ can be defined. As a result,

Proposition 9. *Let $R \in \{\longleftarrow, \longleftrightarrow\}$. The BFS problem for R is solvable by means of an exponential-time-bounded deterministic algorithm.*

Lower bound By giving a polynomial time reduction of the simulation problem of a strongly asynchronous finite automaton by means of a product of strongly asynchronous finite automata, which is known to be *EXPTIME*-hard [8], to the VD problem for \longleftarrow , we prove that the VD problem for \longleftarrow is *EXPTIME*-hard. Given a strongly asynchronous finite automaton $\mathcal{A} = (Q, E, I, F)$ and strongly asynchronous finite automata $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$, let $\mathcal{A}'_1 = (Q'_1, E'_1, I'_1, F'_1), \dots, \mathcal{A}'_n = (Q'_n, E'_n, I'_n, F'_n)$ be the ϵ -free Boolean automaton defined as follows: $Q'_i = Q_i$, E'_i is the function from $Q'_i \times Q'_i$ into the set of all finite subsets of $\mathcal{B}(P) \times (\Sigma_a \cup (\{?, !\} \times \Sigma_s))$ such that for all $q, r \in Q'_i$, for all $\phi \in \mathcal{B}(P)$,

- for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \in E'_i(q, r)$ iff $\phi = \top$ and $\alpha \in E_i(q, r)$,
- for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \notin E'_i(q, r)$,

$I'_i = I_i$ and $F'_i = F_i$. The reader may easily verify that $\mathcal{A} \longleftarrow \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n$ iff there is a valuation V such that $\mathcal{A} \longleftarrow (\mathcal{A}'_1 \otimes \dots \otimes \mathcal{A}'_n)^V$. As a result,

Proposition 10. *The VD problem for \longleftarrow is *EXPTIME*-hard.*

Similarly,

Proposition 11. *The BFD problem for \longleftarrow is *EXPTIME*-hard.*

By giving a polynomial time reduction of the acceptance problem of a linear-space-bounded deterministic Turing machine, which is known to be *PSPACE*-hard, to the VD problem for \longleftrightarrow , we prove that the VD problem for \longleftrightarrow is *PSPACE*-hard. Let $acc, \alpha_1, \alpha_2, \dots$ be an enumeration of Σ_a . Given a linear-space-bounded deterministic Turing machine $M = (Q_M, q_M^0, q_M^1, \Sigma_M, \delta_M)$ and a word $\mathbf{w} \in \Sigma_M^*$, let n be the length of \mathbf{w} . Let $\mathcal{A} = (Q, E, I, F)$ be the strongly asynchronous finite automaton defined as follows: $Q = (Q_M \times \{1, \dots, n\}) \cup \{\perp\}$, E is the function from $Q \times Q$ into the set of all finite subsets of $\mathcal{B}(P) \times \Sigma_a$ such that for all $(q, i), (r, j) \in Q$, for all $\phi \in \mathcal{B}(P)$,

- for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \in E((q, i), (r, j))$ iff $\phi = \top$, $\alpha = \alpha_i$ and there are $u, v \in \Sigma_M$, there is $d \in \{-1, +1\}$ such that $\delta_M(q, u, r, v, d)$ is defined and $j = i + d$ or $\phi = \top$, $\alpha = acc$, $q = q_M^1$, $r = q_M^1$ and $j = i$,

- for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \in E((q, i), \perp)$ iff $\phi = \top$, $\alpha \neq acc$ and $\alpha \neq \alpha_i$ or for all $r \in Q_M$, for all $u, v \in \Sigma_M$, for all $d \in \{-1, +1\}$, $\delta_M(q, u, r, v, d)$ is not defined,
- for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \notin E(\perp, (r, j))$,
- for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \in E(\perp, \perp)$ iff $\phi = \top$ and $\alpha \neq acc$,

$I = \{(q_M^0, 1)\}$ and $F = \emptyset$. Let $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ be the ϵ -free Boolean automata defined as follows: $Q_i = (Q_M \times \Sigma_M) \cup \{\perp_i\}$, E_i is the function from $Q_i \times Q_i$ into the set of all finite subsets of $\mathcal{B}(P) \times (\Sigma_a \cup (\{?, !\} \times \Sigma_s))$ such that for all $(q, u), (r, v) \in Q_i$, for all $\phi \in \mathcal{B}(P)$,

- for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \in E_i((q, u), (r, v))$ iff $\phi = \top$, $\alpha = \alpha_i$ and there is $d \in \{-1, +1\}$ such that $\delta_M(q, u, r, v, d)$ is defined,
- for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \in E_i((q, u), \perp_i)$ iff $\phi = \top$, $\alpha \neq acc$ and $\alpha \neq \alpha_i$ or for all $r \in Q_M$, for all $v \in \Sigma_M$, for all $d \in \{-1, +1\}$, $\delta_M(q, u, r, v, d)$ is not defined,
- for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \notin E_i(\perp_i, (r, v))$,
- for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \in E_i(\perp_i, \perp_i)$ iff $\phi = \top$ and $\alpha \neq acc$,
- for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \notin E_i((q, u), (r, v))$,
- for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \notin E_i((q, u), \perp_i)$,
- for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \notin E_i(\perp_i, (r, v))$,
- for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \notin E_i(\perp_i, \perp_i)$,

$I_i = \{(q_M^0, w_i)\}$ and $F_i = \emptyset$. Suppose that M does not accept \mathbf{w} . Let $Z \subseteq Q \times (Q_1 \times \dots \times Q_n)$ be the binary relation such that

- $(q, i) Z ((q_1, u_1), \dots, (q_n, u_n))$ iff $q = q_i$ and $(q_M^0, 1, \mathbf{w}) \Longrightarrow_M^* (q, i, u_1 \dots u_n)$,
- $\cdot Z (\cdot, \dots, \cdot, \perp_i, \cdot, \dots, \cdot)$,
- $\perp Z (\cdot, \dots, \cdot)$.

The reader may easily verify that there is a valuation V such that Z is a bisimulation between \mathcal{A} and $(\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$. Hence, there is a valuation V such that $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$. Reciprocally, suppose that there is a valuation V such that $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$. Therefore, there is a bisimulation Z between \mathcal{A} and $(\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$. Obviously, M does not accept \mathbf{w} . As a result,

Proposition 12. *The VD problem for \longleftarrow is PSPACE-hard.*

Similarly,

Proposition 13. *The BFD problem for \longleftarrow is PSPACE-hard.*

6 Conclusion

This paper presented different new complexity results for the VD problem and the BFD problem within the framework of service composition with constraints. To sum up, the results are given in snapshot of our work below.

R	valuation decision problem	Boolean formula decision problem
\equiv	$EXPSPACE$ -complete	$EXPSPACE$ -complete
\subseteq	$EXPSPACE$ -complete	$EXPSPACE$ -complete
\leftarrow	$EXPTIME$ -complete	$EXPTIME$ -complete
\leftrightarrow	$PSPACE$ -hard in $EXPTIME$	$PSPACE$ -hard in $EXPTIME$

As pointed out by the above table, a still open question is to evaluate the exact complexity of the valuation decision problem for \leftrightarrow and the Boolean formula decision problem for \leftarrow : are they in $PSPACE$ or are they $EXPTIME$ -hard? Moreover, we focused on the identification of a relevant abstraction to manage conditional actions in the service composition problem. In the future, we intend to explore practical algorithmic approaches to handle the Boolean formula synthesis problem.

References

- Berardi, D., D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella: *Automatic services composition based on behavioral descriptions*. International Journal of Co-operative Information Systems **14** (2005) 333–376.
- Berardi, D., F. Cheikh, G. De Giacomo, F. Patrizi: *Automatic service composition via simulation*. International Journal of Foundations of Computer Science **19** (2008) 429–451.
- Foster, H.: *A Rigorous Approach to Engineering Web Service Compositions*. Thesis of the University of London (2006).
- Foster, H., S. Uchitel, J. Magee, J. Kramer: *WS-Engineer: a model-based approach to engineering Web service compositions and choreography*. In: Test and Analysis of Web Services. Springer (2007) 87–119.
- Fu, X., T. Bultan, J. Su: *Analysis of interacting BPEL Web services*. In: International World Wide Web Conference. ACM (2004) 621–630.
- Héam, P.-C., O. Kouchnarenko, J. Voinot: *How to handle QoS aspects in Web services substitutivity verification*. In: Enabling Technologies: Infrastructure for Collaborative Enterprises. IEEE (2007) 333–338.
- Meyer, A., L. Stockmeyer: *The equivalence problem for regular expressions with squaring requires exponential space*. In: Switching and Automata Theory. IEEE (1972) 125–129.
- Muscholl, A., I. Walukiewicz: *A lower bound on Web services composition*. In: Foundations of Software Science and Computational Structures. Springer (2007) 274–286.
- 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.
- Singh, M., M. Huhns: *Service-Oriented Computing. Semantics, Process, Agents*. Wiley (2005).

Annex

Proof of proposition 2. Let $\mathcal{A} = (Q, E, I, F)$ be a strongly asynchronous finite automaton, $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ be ϵ -free Boolean automata and ϕ be a Boolean formula. We now define a deterministic algorithm which returns the value “accept” iff for all valuations $V, \mathcal{A} \sqsubseteq (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$:

1. for all valuations V
 - (a) compute $(\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$;
 - (b) compute $\widehat{V}(\phi)$;
 - (c) check if $\mathcal{A} \sqsubseteq (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$;
2. if all these calls returns the value “accept” then return the value “accept” else return the value “reject”;

Obviously, the deterministic algorithm above is exponential-space-bounded. Similarly, an exponential-space-bounded deterministic algorithm which returns the value “accept” iff for all valuations $V, \mathcal{A} \equiv (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$ can be defined.

Proof of proposition 4. We consider the cases $exp = \epsilon, exp = exp' \circ exp'', exp = exp' \cup exp''$ and $exp = exp'^+$ of the definition of the ϵ -free Boolean automaton $\mathcal{A}_0 = (Q_0, E_0, I_0, F_0)$ by induction on exp .

Case $exp = \epsilon$. In this case, $\mathcal{A}_0 = (Q_0, E_0, I_0, F_0)$ is the ϵ -free Boolean automaton defined as follows:

- $Q_0 = \{q_I, q, r, s, q_F\}$,
- E_0 is the function from $Q_0 \times Q_0$ into the set of all finite subsets of $\mathcal{B}(P) \times (\Sigma_a \cup (\{?, !\} \times \Sigma_s))$ such that for all $t, u \in Q_0$, for all $\phi \in \mathcal{B}(P)$,
 - for all $\beta \in \Sigma_a, (\phi, \beta) \notin E_0(t, u)$,
 - for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s, (\phi, x\sigma) \in E_0(t, u)$ iff $t = q_I, u = q, \phi = \top, x = !$ and $\sigma = \sigma_1$ or $t = q, u = r, \phi = \top, x = ?$ and $\sigma = \sigma_1$ or $t = r, u = s, \phi = \top, x = !$ and $\sigma = \sigma_2$ or $t = s, u = q_F, \phi = \top, x = ?$ and $\sigma = \sigma_2$,
- $I_0 = \{q_I\}$,
- $F_0 = \{q_F\}$.

The Boolean automaton \mathcal{A}_0 is represented in figure 6. The reader may easily verify that for all valuations $V, lang(exp) = tr((\mathcal{A}_0 \otimes \mathcal{A}_1 \otimes \mathcal{A}_2)^V)$. Remark that $2 = n$. Note that I_0 and F_0 are singletons.

Case $exp = exp' \circ exp''$. In this case, $\mathcal{A}'_0 = (Q'_0, E'_0, I'_0, F'_0)$ being the ϵ -free Boolean automaton associated to exp' and $\mathcal{A}''_0 = (Q''_0, E''_0, I''_0, F''_0)$ being the ϵ -free Boolean automaton associated to exp'' where every $\sigma_i, i \in \{1, \dots, n''\}$, has been replaced by $\sigma_{n'+i}$, $\mathcal{A}_0 = (Q_0, E_0, I_0, F_0)$ is the ϵ -free Boolean automaton defined as follows:

- $Q_0 = Q'_0 \cup Q''_0 \cup \{q_I, q, q_F\}$,

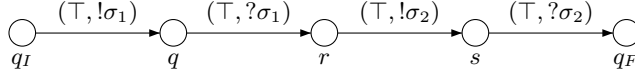


Fig. 6. Finite automaton \mathcal{A}_0 in the case $exp = \epsilon$.

- E_0 is the function from $Q_0 \times Q_0$ into the set of all finite subsets of $\mathcal{B}(P) \times (\Sigma_a \cup (\{?, !\} \times \Sigma_s))$ such that for all $r, s \in Q_0$, for all $\phi \in \mathcal{B}(P)$,
 - for all $\beta \in \Sigma_a$, $(\phi, \beta) \in E_0(r, s)$ iff $r, s \in Q'_0$ and $(\phi, \beta) \in E'_0(r, s)$ or $r, s \in Q''_0$ and $(\phi, \beta) \in E''_0(r, s)$,
 - for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \in E_0(r, s)$ iff $r, s \in Q'_0$ and $(\phi, x\sigma) \in E'_0(r, s)$ or $r, s \in Q''_0$ and $(\phi, x\sigma) \in E''_0(r, s)$ or $r = q_I, s = q'_I, \phi = \top, x = !$ and $\sigma = \sigma_{n'+n''+1}$ or $r = q'_F, s = q, \phi = \top, x = ?$ and $\sigma = \sigma_{n'+n''+1}$ or $r = q, s = q''_I, \phi = \top, x = !$ and $\sigma = \sigma_{n'+n''+2}$ or $r = q'_F, s = q_F, \phi = \top, x = ?$ and $\sigma = \sigma_{n'+n''+2}$,
- $I_0 = \{q_I\}$,
- $F_0 = \{q_F\}$.

The Boolean automaton \mathcal{A}_0 is represented in figure 7. The reader may easily verify that for all valuations V , $lang(exp) = tr((\mathcal{A}_0 \otimes \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_{n'} \otimes \mathcal{A}_{n'+1} \otimes \dots \otimes \mathcal{A}_{n'+n''} \otimes \mathcal{A}_{n'+n''+1} \otimes \mathcal{A}_{n'+n''+2})^V)$. Remark that $n' + n'' + 2 = n$. Note that I_0 and F_0 are singletons.

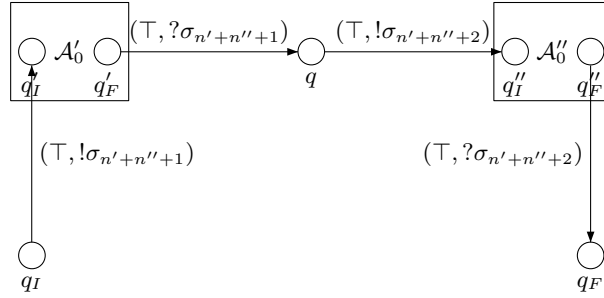


Fig. 7. Finite automaton \mathcal{A}_0 in the case $exp = exp' \circ exp''$.

Case $exp = exp' \cup exp''$. In this case, $\mathcal{A}'_0 = (Q'_0, E'_0, I'_0, F'_0)$ being the ϵ -free Boolean automaton associated to exp' and $\mathcal{A}''_0 = (Q''_0, E''_0, I''_0, F''_0)$ being the ϵ -free Boolean automaton associated to exp'' where every $\sigma_i, i \in \{1, \dots, n''\}$, has been replaced by $\sigma_{n'+i}$, $\mathcal{A}_0 = (Q_0, E_0, I_0, F_0)$ is the ϵ -free Boolean automaton defined as follows:

- $Q_0 = Q'_0 \cup Q''_0 \cup \{q_I, q_F\}$,
- E_0 is the function from $Q_0 \times Q_0$ into the set of all finite subsets of $\mathcal{B}(P) \times (\Sigma_a \cup (\{?, !\} \times \Sigma_s))$ such that for all $q, r \in Q_0$, for all $\phi \in \mathcal{B}(P)$,
 - for all $\beta \in \Sigma_a$, $(\phi, \beta) \in E_0(q, r)$ iff $q, r \in Q'_0$ and $(\phi, \beta) \in E'_0(q, r)$ or $q, r \in Q''_0$ and $(\phi, \beta) \in E''_0(q, r)$,
 - for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \in E_0(q, r)$ iff $q, r \in Q'_0$ and $(\phi, x\sigma) \in E'_0(q, r)$ or $q, r \in Q''_0$ and $(\phi, x\sigma) \in E''_0(q, r)$ or $q = q_I, r = q'_I, \phi = \top, x = !$ and $\sigma = \sigma_{n'+n''+1}$ or $q = q'_F, r = q_F, \phi = \top, x = ?$ and $\sigma = \sigma_{n'+n''+1}$ or $q = q_I, r = q''_I, \phi = \top, x = !$ and $\sigma = \sigma_{n'+n''+2}$ or $q = q''_F, r = q_F, \phi = \top, x = ?$ and $\sigma = \sigma_{n'+n''+2}$,
- $I_0 = \{q_I\}$,
- $F_0 = \{q_F\}$.

The Boolean automaton \mathcal{A}_0 is represented in figure 8. The reader may easily verify that for all valuations V , $lang(exp) = tr((\mathcal{A}_0 \otimes \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_{n'} \otimes \mathcal{A}_{n'+1} \otimes \dots \otimes \mathcal{A}_{n'+n''} \otimes \mathcal{A}_{n'+n''+1} \otimes \mathcal{A}_{n'+n''+2})^V)$. Remark that $n' + n'' + 2 = n$. Note that I_0 and F_0 are singletons.

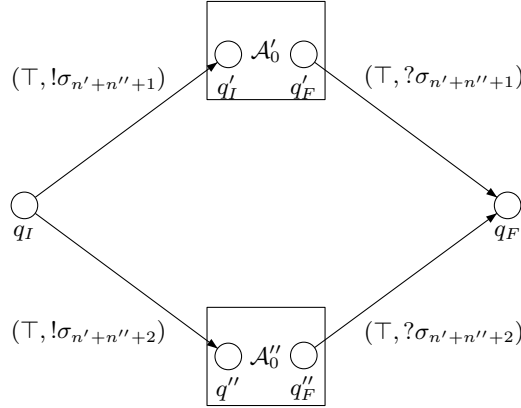


Fig. 8. Finite automaton \mathcal{A}_0 in the case $exp = exp' \cup exp''$.

Case $exp = exp'^+$. In this case, $\mathcal{A}'_0 = (Q'_0, E'_0, I'_0, F'_0)$ being the ϵ -free Boolean automaton associated to exp' , $\mathcal{A}_0 = (Q_0, E_0, I_0, F_0)$ is the ϵ -free Boolean automaton defined as follows:

- $Q_0 = Q'_0 \cup \{q_I, q, q_F\}$,
- E_0 is the function from $Q_0 \times Q_0$ into the set of all finite subsets of $\mathcal{B}(P) \times (\Sigma_a \cup (\{?, !\} \times \Sigma_s))$ such that for all $r, s \in Q_0$, for all $\phi \in \mathcal{B}(P)$,
 - for all $\beta \in \Sigma_a$, $(\phi, \beta) \in E_0(r, s)$ iff $r, s \in Q'_0$ and $(\phi, \beta) \in E'_0(r, s)$,

- for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \in E_0(r, s)$ iff $r, s \in Q'_0$ and $(\phi, x\sigma) \in E'_0(r, s)$ or $r = q_I, s = q'_I, \phi = \top, x = !$ and $\sigma = \sigma_{n'+1}$ or $r = q'_F, s = q_F, \phi = \top, x = ?$ and $\sigma = \sigma_{n'+1}$ or $r = q_F, s = q, \phi = \top, x = !$ and $\sigma = \sigma_{n'+2}$ or $r = q, s = q_I, \phi = \top, x = ?$ and $\sigma = \sigma_{n'+2}$,
- $I_0 = \{q_I\}$,
- $F_0 = \{q_F\}$.

The Boolean automaton \mathcal{A}_0 is represented in figure 9. The reader may easily verify that for all valuations V , $\text{lang}(exp) = \text{tr}((\mathcal{A}_0 \otimes \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_{n'} \otimes \mathcal{A}_{n'+1} \otimes \mathcal{A}_{n'+2})^V)$. Remark that $n' + 2 = n$. Note that I_0 and F_0 are singletons.

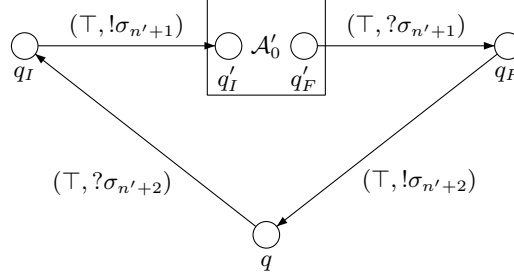


Fig. 9. Finite automaton \mathcal{A}_0 in the case $exp = exp'^+$.

Proof of proposition 5. By giving a polynomial time reduction of the equivalence problem for regular expressions with squaring, which is known to be *EXPSPACE*-hard [7], to the Boolean formula decision problem for \sqsubseteq , we prove that the Boolean formula decision problem for \sqsubseteq is *EXPSPACE*-hard. Let $\sigma_1, \sigma_2, \dots$ be an enumeration of Σ_s . Given a regular expression exp with squaring, let $n = 2 \times op(exp)$. Let $\mathcal{A} = (Q, E, I, F)$ be the strongly asynchronous finite automaton defined as above. For all positive integers i , if $i \leq n$ then let $\mathcal{A}_i = (Q_i, E_i, I_i, F_i)$ be the ϵ -free Boolean automaton defined as above. Let $\mathcal{A}_0 = (Q_0, E_0, I_0, F_0)$ be the ϵ -free Boolean automaton defined by induction on exp as above and $\phi = \top$. The reader may easily verify that $\text{lang}(exp) = \Sigma_a^*$ iff for all valuations V , $\mathcal{A} \sqsubseteq (\mathcal{A}_0 \otimes \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$. Similarly, the reader may easily verify that $\text{lang}(exp) = \Sigma_a^*$ iff for all valuations V , $\mathcal{A} \equiv (\mathcal{A}_0 \otimes \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$.

Proof of proposition 8. Let $\mathcal{A} = (Q, E, I, F)$ be a strongly asynchronous finite automaton, $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ be ϵ -free Boolean automata and ϕ be a Boolean formula. We now define a deterministic algorithm which returns the value “accept” iff for all valuations V , $\mathcal{A} \leftarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$:

1. for all valuations V
 - (a) compute $(\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$;
 - (b) compute $\widehat{V}(\phi)$;
 - (c) check if $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$;
2. if all these calls returns the value “accept” then return the value “accept” else return the value “reject”;

Obviously, the deterministic algorithm above is exponential-time-bounded. Similarly, an exponential-time-bounded deterministic algorithm which returns the value “accept” iff for all valuations V , $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$ can be defined.

Proof of proposition 11. By giving a polynomial time reduction of the simulation problem of a strongly asynchronous finite automaton by means of a product of strongly asynchronous finite automata, which is known to be *EXPTIME*-hard [8], to the Boolean formula decision problem for \longleftarrow , we prove that the Boolean formula decision problem for \longleftarrow is *EXPTIME*-hard. Given a strongly asynchronous finite automaton $\mathcal{A} = (Q, E, I, F)$ and strongly asynchronous finite automata $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1), \dots, \mathcal{A}_n = (Q_n, E_n, I_n, F_n)$, let $\mathcal{A}'_1 = (Q'_1, E'_1, I'_1, F'_1), \dots, \mathcal{A}'_n = (Q'_n, E'_n, I'_n, F'_n)$ be the ϵ -free Boolean automaton defined as above and $\phi = \top$. Suppose that $\mathcal{A} \longleftarrow \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n$. Hence, there is a simulation Z of \mathcal{A} by $\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n$. Let V be a valuation. Obviously, Z is a simulation of \mathcal{A} by $(\mathcal{A}'_1 \otimes \dots \otimes \mathcal{A}'_n)^V$. Therefore, for all valuations V , $\mathcal{A} \longleftarrow (\mathcal{A}'_1 \otimes \dots \otimes \mathcal{A}'_n)^V$ iff $\widehat{V}(\phi) = 1$. Reciprocally, suppose that for all valuations V , $\mathcal{A} \longleftarrow (\mathcal{A}'_1 \otimes \dots \otimes \mathcal{A}'_n)^V$ iff $\widehat{V}(\phi) = 1$. Let V be a valuation. Thus, there is a simulation Z of \mathcal{A} by $(\mathcal{A}'_1 \otimes \dots \otimes \mathcal{A}'_n)^V$. Obviously, Z is a simulation of \mathcal{A} by $\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n$. Consequently, $\mathcal{A} \longleftarrow \mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n$.

Proof of proposition 13. By giving a polynomial time reduction of the acceptance problem of a linear-space-bounded deterministic Turing machine, which is known to be *PSPACE*-hard, to the valuation decision problem for \longleftrightarrow , we prove that the valuation decision problem for \longleftrightarrow is *PSPACE*-hard. Let $acc, \alpha_1, \alpha_2, \dots$ be an enumeration of Σ_a . Given a linear-space-bounded deterministic Turing machine $M = (Q_M, q_M^0, q_M^1, \Sigma_M, \delta_M)$ and a word $\mathbf{w} \in \Sigma_M^*$, let n be the length of \mathbf{w} , $\mathcal{A} = (Q, E, I, F)$ be the strongly asynchronous finite automaton defined as above and $\phi = \top$. Suppose that M does not accept \mathbf{w} . Let V be a valuation. Let $Z \subseteq Q \times (Q_1 \times \dots \times Q_n)$ be the binary relation such that

- $(q, i) Z ((q_1, u_1), \dots, (q_n, u_n))$ iff $q = q_i$ and $(q_M^0, 1, \mathbf{w}) \Longrightarrow_M^* (q, i, u_1 \dots u_n)$,
- $\cdot Z (\cdot, \dots, \cdot, \perp_i, \cdot, \dots, \cdot)$,
- $\perp Z (\cdot, \dots, \cdot)$.

Obviously, Z is a bisimulation between \mathcal{A} and $(\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$. Hence, for all valuations V , $\mathcal{A} \longleftrightarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$. Reciprocally, suppose that for all valuations V , $\mathcal{A} \longleftrightarrow (\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$. Let V be a valuation. Therefore, there is a bisimulation Z between \mathcal{A} and $(\mathcal{A}_1 \otimes \dots \otimes \mathcal{A}_n)^V$. Obviously, M does not accept \mathbf{w} .