Composition of Services with Constraints

Philippe Balbiani, Fahima Cheikh Alili, Pierre-Cyrille Héam, Olga Kouchnarenko

Composition of Services with Constraints

Balbiani, Cheikh [1,2]

IRIT-CNRS, Toulouse, France

Héam[3]

LSV-CNRS-INRIA, Cachan, France

Kouchnarenko[4]

INRIA-CASSIS-LIFC, Besançon, France

Abstract

Composition of Web services consists of the interleaving of the sequence of actions executed by the elementary services in accordance with a client specification. We model Web services as automata executing actions and also sending and receiving messages. This paper provides a theoretical study for three service composition problems, and in particular for the problem of computing a Boolean formula which exactly characterises the conditions required for services to answer the client's request. New complexity results are established for these problems within the framework of service composition with constraints.

Keywords: Composition, Décidability.

1 Introduction and Related Work

Service oriented computing [23] 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 [3,2,17]. Often, services are seen as finite automata. In this case, the client specification is given by a finite automaton which represents all computations that a client wants the services to execute. By executing their transitions, services modify their environment and that of the client. The problem of combining services becomes that of

[1] Email: Philippe.Balbiani@irit.fr
[2] Email: Cheick@irit.fr
[3] Email: pcheam@lsv.ens-cachan.fr
[4] Email: Olga.Kouchnarenko@lifc.univ-fcomte.fr

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

composing automata, like in [3,2]. In other cases, services are able to send and to receive messages. Client specification is then given by a logical formula which represents client's goals he wants services to reach. By communicating together, services modify their knowledge and those of their client (see, e.g., [17]). This paper follows the line of reasoning suggested in [5,6,7,8], which consists in giving the semantics of services by means of automata. The paper particularly focuses on the following problem: Given services $\mathcal{A}_1$, ..., $\mathcal{A}_n$ and the request $\mathcal{A}$ of a client, can $\mathcal{A}_1$, ..., $\mathcal{A}_n$ 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 $\mathcal{A}_1$, ..., $\mathcal{A}_n$. For instance, a conditional action may be, for some $i, j \in \{1, \ldots, n\}$, 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. A conditional action may also be, for some $i, j \in \{1, \ldots, n\}$, 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 appear when specifying services. As far as we know, the composition problems studied in the literature do not handle this kind of conditions. This paper provides a theoretical study for three service composition problems, and in particular for the problem of computing a Boolean formula ϕ which exactly characterises the conditions required for $\mathcal{A}_1$, ..., $\mathcal{A}_n$ to answer the client's request. The paper 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. Finally we conclude in section 6.

Related Work In the context of finite automata, many research works on complexity results for various finite automata compositions have been done.

In [10], the authors investigated the following problem: given $n + m$ finite automata $\mathcal{A}_1, \ldots, \mathcal{A}_{n+m}$, what is the complexity of deciding whether $\mathcal{A}_1 \times \ldots \times \mathcal{A}_n$ is equivalent to $\mathcal{A}_{n+1} \times \ldots \times \mathcal{A}_{n+m}$. The class of problems considered in [10] is for non flat systems, i.e. one requires that $n \geq 2$ and $m \geq 2$. Another crucial difference w.r.t. our work concerns the product. In [10], the product is partially synchronised but, contrarily to our work, synchronisation does not produce ε-transition but labeled transitions. Their main result shows that the above decision problem is EXPTIME-hard for any relation between the simulation preorder and bisimulation, and that it is EXSPACE-hard for any relation between trace inclusion and the intersection of ready trace equivalence.

A more general problem is considered in [20] where the product is closer to ours since some actions can be *hidden*, i.e. replaced by an ε-transition. The author proved that for non flat systems the equivalence checking is PSPACE-hard for any relation between bisimilarity and trace equivalence, and that the problem is EXPSPACE-complete for trace equivalence, and EXPTIME-complete for bisimilarity. It was also conjectured in [20] that the problem is EXPTIME-hard for any relation between bisimilarity and trace equivalence. This conjecture was enhanced and proved in [22]: the problem is EXPTIME-hard for any relation between bisimilarity and trace preorder.

In [15], it is shown that deciding whether $\mathcal{A}$ is simulated by $\mathcal{A}_1 \times \ldots \times \mathcal{A}_n$ is still EXPTIME-hard. In this work, the considered product is asynchronous.

Several recent works focus on the use of finite automata based models to address Web services composition problems. In [18,17], the authors propose a model where Web services compositions expressed in BPEL are formally defined by state transition systems with communicating and internal (unobservable) actions. In [17], this model was enriched using a knowledge base, and in [9] using a theory. These approaches were applied to practical applications in [24,12,11]. In [1,21], Web services are defined by PWL-S documents and modelled by guarded finite automata. A similar approach is investigated in [16] where Web services are defined in BPEL. In [14], the authors model by Input/Output automata Web services interfaces described in BPEL, OWLS and WSDL.

2 Preliminaries

Let P be a finite set of Boolean variables (with typical members denoted p, p', ...), Σ_a be a countable set of asynchronous actions (with typical members denoted α, β, ...) and Σ_s be a countable set of synchronous actions (with typical members denoted σ, τ, ...). 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 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. Given a weakly asynchronous automaton $\mathcal{A}$, 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 $(q_0, q_1, \ldots, q_m), (r_0, r_1, \ldots, r_n) \in Q^+$ 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 $(\alpha_1, \ldots, \alpha_k) \in \Sigma_a^*$ is a sequence $(q_0, q_1, \ldots, q_k) \in Q^+$ 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 traces of $\mathcal{A}$, denoted $tr(\mathcal{A})$, is the set of all sequences $(\alpha_1, \ldots, \alpha_k) \in \Sigma_a^*$ such that there is a run of $\mathcal{A}$ on $(\alpha_1, \ldots, \alpha_k)$.

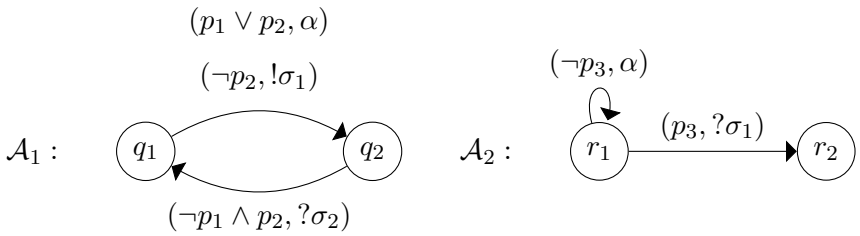

Figure 1. Boolean automata $\mathcal{A}_1$ and $\mathcal{A}_2$.

Boolean automata

The set $\mathcal{B}(P)$ of all Boolean formulas (with typical members denoted ϕ, ψ, ...) is defined by: $\phi ::= p \mid \bot \mid \neg\phi \mid (\phi \vee \phi)$, with $p \in P$. The other constructs are defined as usual. In particular, $\top = \neg\bot$ and $(\phi \wedge \psi) = \neg(\neg\phi \vee \neg\psi)$. 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 Fig. 1, with $\Sigma_a = \{\alpha\}$ and $\Sigma_s = \{\sigma_1, \sigma_2\}$.

Synchronisation

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

- $Q = Q_1 \times \ldots \times Q_n$,
- 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 $\boldsymbol{q}, \boldsymbol{r} \in Q$, for all $\phi \in \mathcal{B}(P)$,
  - for all $\alpha \in \Sigma_a$, $(\phi, \alpha) \in E(\boldsymbol{q}, \boldsymbol{r})$ iff there is $i \in \{1, \ldots, n\}$ such that $\boldsymbol{q} \equiv_i \boldsymbol{r}$ and $(\phi, \alpha) \in E_i(q_i, r_i)$,
  - $(\phi, \epsilon) \in E(\boldsymbol{q}, \boldsymbol{r})$ iff there are $i_1, i_2 \in \{1, \ldots, n\}$, there are $\phi_{i_1}, \phi_{i_2} \in \mathcal{B}(P)$, there is $\sigma \in \Sigma_s$ such that $i_1 \neq i_2$, $\boldsymbol{q} \equiv_{i_1, i_2} \boldsymbol{r}$, 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 $\phi = \phi_{i_1} \wedge \phi_{i_2}$.
- $I = I_1 \times \ldots \times I_n$ and $F = F_1 \times \ldots \times F_n$,

the binary relations $\equiv_i, \equiv_{i_1, i_2} \subseteq Q \times Q$ being such that for all $\boldsymbol{q}, \boldsymbol{r} \in Q$, $\boldsymbol{q} \equiv_i \boldsymbol{r}$ iff for all $j \in \{1, \ldots, n\}$, if $i \neq j$ then $q_j = r_j$ and $\boldsymbol{q} \equiv_{i_1, i_2} \boldsymbol{r}$ iff for all $j \in \{1, \ldots, n\}$, if $i_1 \neq j$ and $i_2 \neq j$ then $q_j = r_j$. Consider, as example, the Boolean automata A_1 and $\mathcal{A}_2$ from Fig. 1 and $\mathcal{A}_1 \otimes \mathcal{A}_2$ from Fig. 2.

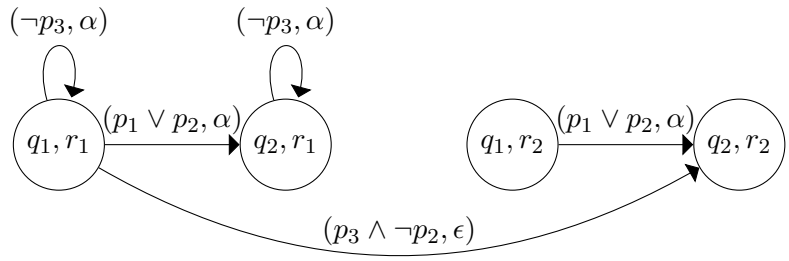

Figure 2. Boolean automata $\mathcal{A}_1 \otimes \mathcal{A}_2$.

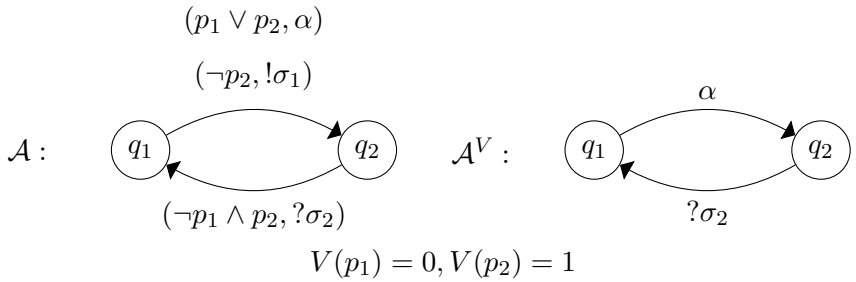

Figure 3. Boolean automaton $\mathcal{A}$ and the associated automaton $\mathcal{A}^V$.

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 Fig. 3.

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 $dom(Z) \cap I \neq \emptyset$ and $ran(Z) \cap I' \neq \emptyset$ 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: $dom(Z)$ and $ran(Z)$ 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

3.1 Formal definitions

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 $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1)$, ..., $\mathcal{A}_n = (Q_n, E_n, I_n, F_n)$,
- output: check if there is a valuation V such that $\mathcal{A}\ R\ (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$.

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 $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1)$, ..., $\mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ and a Boolean formula ϕ,
- output: check if for all valuations V, $\mathcal{A}\ R\ (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ 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 $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1)$, ..., $\mathcal{A}_n = (Q_n, E_n, I_n, F_n)$,
- output: find out a Boolean formula ϕ such that for all valuations V, $\mathcal{A}\ R\ (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$.

The questions we are interested in are: Is there a valuation of these services such that the desired composition is possible (VD problem)? How to compute a boolean formula ϕ over these predicates that is true iff the desired composition is possible (BFS problem)?

3.2 Motivating Example

Consider three services S_1, S_2, S_3.

Service S_1 provides an on-line booking for European football match places that will take place in Madrid. This service is modeled by the automaton $\mathcal{A}_1$ in Fig. 4.

When S_1 receives a request for a football place ($ticket\_req$) booking, it checks whether there is available places having the desired price (action c_v). This check can be done if and only if the request is provided by a non black listed supporter (predicate $\neg p_{black}$). If there is no available place, S_1 informs the requester by the

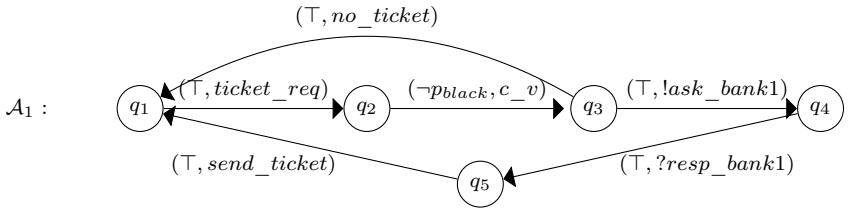

Figure 4. Boolean automaton $\mathcal{A}_1$

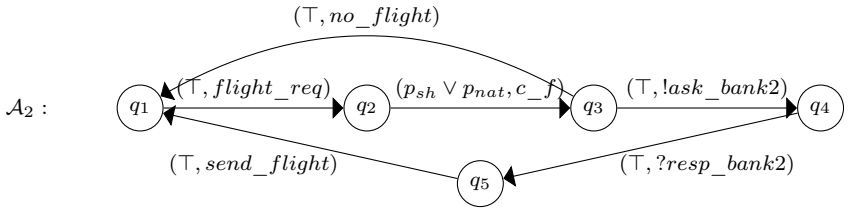

Figure 5. Boolean automaton $\mathcal{A}_2$

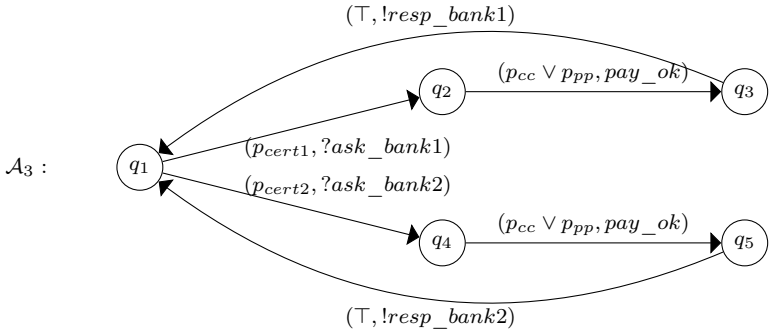

Figure 6. Boolean automaton $\mathcal{A}_3$

action no_ticket. If there is a ticket, S_1 asks the bank service (S_3) for the ticket payment (actions $!ask\_bank1$). If the bank answers the payment is Ok (message $?resp\_bank1$), the ticket is sent to the requester (action $send\_ticket$).

Service S_2 sells flight tickets. It is modeled by the automaton $\mathcal{A}_2$ in Fig. 5.

S_2 works like S_1 but sells tickets only for people who do not need a Visa for coming to Spain, thus either people whose nationality is in a given list (predicate p_{nat}), or if the starting point of the flight is in the Shengen Space (predicate p_{sh}).

The **service** S_3 is the bank service modeled by the automaton $\mathcal{A}_3$ in Fig. 6.

S_3 accepts request from services having a security certificate (predicate p_{cert_i} means that S_i has such a certificate). Then, if the buyer has either a credit card (predicate p_{cc}) or a paypal account (predicate p_{pp}), and if the payment is OK (in order to not overload the example, we do not model this communicating point; we just encode it by the action pay_ok), the bank validates the payment to the related service.

As the reader can see, services can be composed to provide the intented service S (depicted in Fig. 7), with several conditions on the value of predicates $p_{nat}, p_{cert1},$, etc.

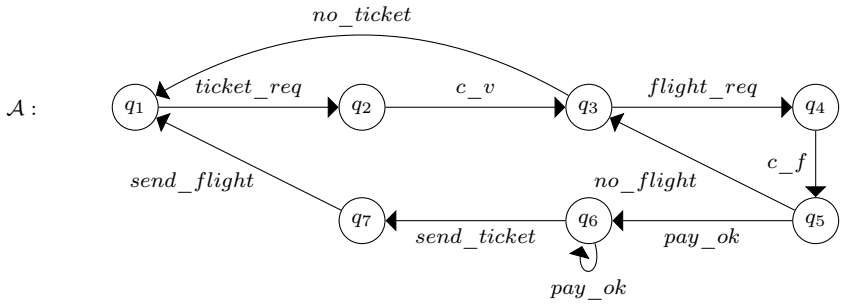

Figure 7. Automaton $\mathcal{A}$

4 Trace Inclusion and Trace Equivalence

Let us recall that given two finite automata $\mathcal{A}_1$ and $\mathcal{A}_2$, testing whether $\mathcal{A}_1 \sqsubseteq (\mathcal{A}_2$ can be done in $PSPACE$.

Upper bound

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

(i) for all valuations V
  (a) compute $(\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$;
  (b) check if $\mathcal{A} \sqsubseteq (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$;

(ii) 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 $\mathcal{A} \equiv (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ can be defined. As a result,

Proposition 4.1 *Let $R \in \{\sqsubseteq, \equiv\}$. The VD problem for R is in $EXPSPACE$.*

Similarly,

Proposition 4.2 *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 $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1)$, ..., $\mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ be ϵ-free Boolean automata. We now define a deterministic algorithm which returns a Boolean formula ϕ such that for all valuations V, $\mathcal{A} \sqsubseteq (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$:

(i) $\phi := \bot$;

(ii) for all maximal consistent conjunctions ψ of P-literals
  (a) compute the unique valuation V such that $\widehat{V}(\psi) = 1$;
  (b) compute $(\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$;
  (c) if $\mathcal{A} \sqsubseteq (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ 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, $\mathcal{A} \equiv (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$ can be defined. As a result,

Proposition 4.3 *Let $R \in \{\sqsubseteq, \equiv\}$. The BFS problem for R is solvable by means of an exponential-space-bounded deterministic algorithm.*

Lower bound

To prove that the VD problem for $\sqsubseteq$ is $EXPSPACE$-hard, we give a polynomial time reduction of the universality problem for regular expressions with squaring, which is known to be $EXPSPACE$-hard [13], to the VD problem for $\sqsubseteq$. The set of all regular expressions with squaring (with typical members denoted exp, exp', ...) 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 ϵ, $\circ$, $\cup$, $^+$ 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 σ_1, σ_2, ... 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: $Q_i = \{q_{1i}, q_{2i}\}$, 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 $\phi = \top$, $x = ?$, $\sigma = \sigma_i$, $q = q_{1i}$ and $r = q_{2i}$ or $\phi = \top$, $x = !$, $\sigma = \sigma_i$, $q = q_{2i}$ and $r = q_{1i}$,

$I_i = \{q_{1i}\}$ and $F_i = \{q_{1i}\}$. 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 Fig. 8. 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.

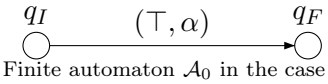

Figure 8. Finite automaton $\mathcal{A}_0$ in the case $exp = \alpha$.

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, $lang(exp') = tr((\mathcal{A}'_0 \otimes \mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_{n'})^V)$ 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, $lang(exp'') = tr((\mathcal{A}''_0 \otimes \mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_{n''})^V)$ where $n'' = q \times op(exp'')$. 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 similar.

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 Fig. 9. The reader may easily verify that for all valuations V, $lang(exp) = tr((\mathcal{A}_0 \otimes \mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_{n'} \otimes \mathcal{A}_{n'+1} \otimes \mathcal{A}_{n'+2})^V)$. Remark that $n' + 2 = n$. Note also that I_0 and F_0 are singletons.

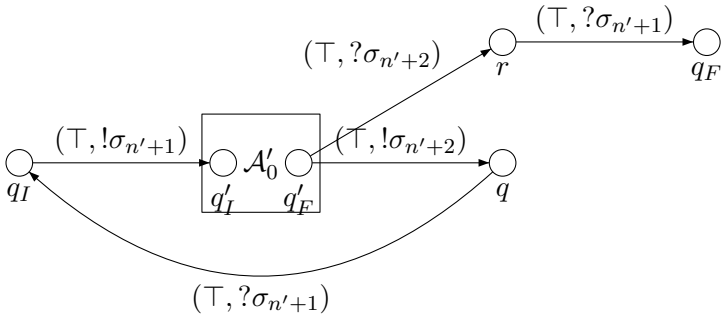

Figure 9. Finite automaton $\mathcal{A}_0$ in the case $exp = exp'^2$.

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

Proposition 4.4 *Let $R \in \{\sqsubseteq, \equiv\}$. The VD problem for R is EXPSPACE-hard.*

Similarly,

Proposition 4.5 *Let $R \in \{\sqsubseteq, \equiv\}$. The BFD problem for R is EXPSPACE-hard.*

According to Meyer and Stockmeyer [13], for all deterministic Turing machines M solving the universality problem for regular expressions with squaring, there exists

a constant $c > 1$ such that M needs at least space c^n on some input of length n for infinitely many n. Suppose that there is a deterministic algorithm f solving the BFS problem for R and such that for all constants $c > 1$, f needs at least space c^n on some input of length n for finitely many n only. Let M_f be the deterministic Turing machine that behaves as follows given a regular expression exp with squaring:

(i) M_f computes $n = 2 \times op(exp)$;

(ii) M_f computes the strongly asynchronous finite automaton $\mathcal{A} = (Q, E, I, F)$ defined as above;

(iii) for all positive integers i, if $i \leq n$ then M_f computes the ϵ-free Boolean automaton $\mathcal{A}_i = (Q_i, E_i, I_i, F_i)$ defined as above;

(iv) M_f computes the ϵ-free Boolean automaton $\mathcal{A}_0 = (Q_0, E_0, I_0, F_0)$ defined by induction on exp as above;

(v) M_f simulates f on input $\mathcal{A}$ and $\mathcal{A}_0$ and $\mathcal{A}_1$, ..., $\mathcal{A}_n$ until it is about to return a value ϕ_f;

(vi) if ϕ_f is a Boolean tautology then return the value "accept" else return the value "reject";

Obviously, M_f solves the universality problem for regular expressions with squaring and for all constants $c > 1$ M_f needs at least space c^n on some input of length n for finitely many n only: a contradiction. As a result,

Proposition 4.6 *Let $R \in \{\sqsubseteq, \equiv\}$. For all deterministic algorithms f solving the BFS problem for R, there exist a constant $c > 1$, such that f needs at least space c^n on some input of length n for infinitely many n.*

5 Simulation and Bisimulation

Let us recall that given two finite automata $\mathcal{A}_1$ and $\mathcal{A}_2$, testing whether $\mathcal{A}_1$ simulates $\mathcal{A}_2$ can be done in polynomial time.

Upper bound

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

(i) for all valuations V
  (a) compute $(\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$;
  (b) check if $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$;

(ii) 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 $\mathcal{A} \longleftrightarrow (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ can be defined. As a result,

Proposition 5.1 *Let $R \in \{\longleftarrow, \longleftrightarrow\}$. The VD problem for R is in $EXPTIME$.*

Let $\mathcal{A} = (Q, E, I, F)$ be a strongly asynchronous finite automaton, $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1)$, ..., $\mathcal{A}_n = (Q_n, E_n, I_n, F_n)$ 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, $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$:

(i) for all valuations V
  (a) compute $(\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$;
  (b) compute $\widehat{V}(\phi)$;
  (c) check if $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$;

(ii) 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, $\mathcal{A} \longleftrightarrow (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$ can be defined.

Proposition 5.2 *Let $R \in \{\longleftarrow, \longleftrightarrow\}$. The BFD problem for R is in $EXPTIME$.*

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

(i) $\phi := \bot$;

(ii) for all maximal consistent conjunctions ψ of P-literals
  (a) compute the unique valuation V such that $\widehat{V}(\psi) = 1$;
  (b) compute $(\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$;
  (c) if $\mathcal{A} \longleftarrow (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ 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, $\mathcal{A} \longleftrightarrow (\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n)^V$ iff $\widehat{V}(\phi) = 1$ can be defined. As a result,

Proposition 5.3 *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 [15], 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 $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1)$, ..., $\mathcal{A}_n = (Q_n, E_n, I_n, F_n)$, let $\mathcal{A}'_1 = (Q'_1, E'_1, I'_1, F'_1)$, ..., $\mathcal{A}'_n = (Q'_n, E'_n, I'_n, F'_n)$ be the ϵ-free Boolean automata 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 $\mathcal{A} \longleftarrow \mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n$ iff there is a valuation V such that $\mathcal{A} \longleftarrow (\mathcal{A}'_1 \otimes \ldots \otimes \mathcal{A}'_n)^V$. As a result,

Proposition 5.4 *The VD problem for $\longleftarrow$ is $EXPTIME$-hard.*

Similarly,

Proposition 5.5 *The BFD problem for $\longleftarrow$ is $EXPTIME$-hard.*

Proof We prove that the Boolean formula decision problem for $\longleftarrow$ is $EXPTIME$-hard 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 [15], to the Boolean formula decision problem for $\longleftarrow$. Given a strongly asynchronous finite automaton $\mathcal{A} = (Q, E, I, F)$ and strongly asynchronous finite automata $\mathcal{A}_1 = (Q_1, E_1, I_1, F_1)$, ..., $\mathcal{A}_n = (Q_n, E_n, I_n, F_n)$, let $\mathcal{A}'_1 = (Q'_1, E'_1, I'_1, F'_1)$, ..., $\mathcal{A}'_n = (Q'_n, E'_n, I'_n, F'_n)$ be the ϵ-free Boolean automaton defined as above and $\phi = \top$. Suppose that $\mathcal{A} \longleftarrow \mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n$. Hence, there is a simulation Z of $\mathcal{A}$ by $\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n$. Let V be a valuation. Obviously, Z is a simulation of $\mathcal{A}$ by $(\mathcal{A}'_1 \otimes \ldots \otimes \mathcal{A}'_n)^V$. Therefore, for all valuations V, $\mathcal{A} \longleftarrow (\mathcal{A}'_1 \otimes \ldots \otimes \mathcal{A}'_n)^V$ iff $\widehat{V}(\phi) = 1$. Reciprocally, suppose that for all valuations V, $\mathcal{A} \longleftarrow (\mathcal{A}'_1 \otimes \ldots \otimes \mathcal{A}'_n)^V$ iff $\widehat{V}(\phi) = 1$. Let V be a valuation. Thus, there is a simulation Z of $\mathcal{A}$ by $(\mathcal{A}'_1 \otimes \ldots \otimes \mathcal{A}'_n)^V$. Obviously, Z is a simulation of $\mathcal{A}$ by $\mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n$. Consequently, $\mathcal{A} \longleftarrow \mathcal{A}_1 \otimes \ldots \otimes \mathcal{A}_n$.□

We prove that the VD problem for $\longleftrightarrow$ is $PSPACE$-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 $\longleftrightarrow$. Let acc, α_1, α_2, ... be an enumeration of Σ_a. Given a linear-space-bounded deterministic Turing machine $M = (Q_M, q^0_M, q^1_M, \Sigma_M, \delta_M)$ and a word $\boldsymbol{w} \in \Sigma^*_M$, let n be the length of $\boldsymbol{w}$. Let $\Sigma_f = \{acc, \alpha_1, \ldots, \alpha_n\}$. Let $\mathcal{A} = (Q, E, I, F)$ be the strongly asynchronous finite automaton defined as follows: $Q = (Q_M \times \{1, \ldots, n\}) \cup \{\bot\}$, E is the function from $Q \times Q$ into the set of all finite subsets of $\mathcal{B}(P) \times \Sigma_f$ such that for all $(q, i), (r, j) \in Q$, for all $\phi \in \mathcal{B}(P)$,

- for all $\alpha \in \Sigma_f$, $(\phi, \alpha) \in E((q, i), (r, j))$ 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 $\phi = \top$, $\alpha = acc$, $q = q^1_M$, $r = q^1_M$ and $j = i$,
- for all $\alpha \in \Sigma_f$, $(\phi, \alpha) \in E((q, i), \bot)$ 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 $\alpha \in \Sigma_f$, $(\phi, \alpha) \notin E(\bot, (r, j))$,
- for all $\alpha \in \Sigma_f$, $(\phi, \alpha) \in E(\bot, \bot)$ iff $\phi = \top$ and $\alpha \neq acc$,

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

- for all $\alpha \in \Sigma_f$, $(\phi, \alpha) \in E_i((q, u), (r, v))$ 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 $\alpha \in \Sigma_f$, $(\phi, \alpha) \in E_i((q,u), \bot_i)$ 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 $\alpha \in \Sigma_f$, $(\phi, \alpha) \notin E_i(\bot_i, (r,v))$,
- for all $\alpha \in \Sigma_f$, $(\phi, \alpha) \in E_i(\bot_i, \bot_i)$ 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 $\sigma \in \Sigma_s$, $(\phi, x\sigma) \notin E_i((q,u), \bot_i)$,
- for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \notin E_i(\bot_i, (r,v))$,
- for all $x \in \{?, !\}$, for all $\sigma \in \Sigma_s$, $(\phi, x\sigma) \notin E_i(\bot_i, \bot_i)$,

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

- $(q,i)\ Z\ ((q_1,u_1),\ldots,(q_n,u_n))$ iff $q = q_i$ and $(q_M^0, 1, \boldsymbol{w}) \Longrightarrow_M^* (q, i, u_1 \ldots u_n)$,
- $\cdot\ Z\ (\cdot, \ldots, \cdot, \bot_i, \cdot, \ldots, \cdot)$,
- $\bot\ Z\ (\cdot, \ldots, \cdot)$.

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

Proposition 5.6 *The VD problem for $\longleftrightarrow$ is $PSPACE$-hard. The BFD problem for $\longleftrightarrow$ 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 $\longleftrightarrow$ and the Boolean formula decision problem for $\longleftrightarrow$: 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

[1] Berardi, D., D. Calvanese, G. De Giacomo, R. Hull, M. Mecella: *Automatic composition of transition-based semantic Web services with messaging.* In: Very Large Data Bases. ACM (2005) 613–624.

[2] Berardi, D., D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella: *Automatic services composition based on behavioral descriptions.* Int. Journal of Cooperative Information Systems **14** (2005) 333–376.

[3] Berardi, D., F. Cheikh, G. De Giacomo, F. Patrizi: *Automatic service composition via simulation.* Int. Journal of Foundations of Computer Science **19** (2008) 429–451.

[4] Berardi, D., M. Pistore, P. Traverso: *Automatic Web service composition by on-the-fly belief space search.* In: Proceedings of ICAPS'06, (2006) 358–361.

[5] Foster, H.: *A Rigorous Approach to Engineering Web Service Compositions.* Thesis of the University of London (2006).

[6] 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.

[7] Fu, X., T. Bultan, J. Su: *Analysis of interacting BPEL Web services.* In: Int. World Wide Web Conference. ACM (2004) 621–630.

[8] 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.

[9] Hoffman, J., P. Bertoli, M. Pistore: *Web service composition as planning, revisited.* In: Proceeedings of AAAI'07 AAAI (2007) 1013–1018.

[10] Laroussinie, F., Ph. Schnoebelen: *The state explosion problem from trace to bisimulation equivalence.* In: Foundations of Software Science and Computational Structures. Springer (2007) 192–207.

[11] Marconi, A., M. Pistore, P. Poccianti, P. Traverso: *Automated Web service composition at work: the amazon/mps case study* In: Proceedings of ICWS'07, IEEE, (2007) 767–774.

[12] Marconi, A., M. Pistore, P. Traverso: *Automated composition of Web services: the astro approach* In: IEEE Data Engineering Bulletin **31** (2008) 23–26.

[13] Meyer, A., L. Stockmeyer: *The equivalence problem for regular expressions with squaring requires exponential space.* In: Switching and Automata Theory. IEEE (1972) 125–129.

[14] Mitra, S., R.J. Kumar, S. Basu: *Automates CHoregrapher Synthesis for Web service composition using I/O Automata.* In: Proceedings of ICWS'07 IEEE (2007) 364–371.

[15] Muscholl, A., I. Walukiewicz: *A lower bound on Web services composition.* In: Foundations of Software Science and Computational Structures. Springer (2007) 274–286.

[16] Pathak, J., S. Basu, R. Lutz, V. Honavar: *Parallel web service composition in moscoe: A choregraphy based approach.* In: Proceedings of ECOWS'96, IEEE (2006) 3–12.

[17] Pistore, M., A. Marconi, P. Bertoli, P. Traverso: *Automated composition of Web services by planning at the knowledge level.* In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (2005) 1252–1259.

[18] Pistore, M., P. Traverso, P. Bertoli: *Automated composition of Web services by planning in asynchronous domains.* In: Proceedings of ICAPS'05, AAAI, (2005) 2–12.

[19] Pistore, M., P. Traverso, P. Bertoli, A. Marconi: *Automated synthesis of composite BPEL4WS web services..* In: Proceedings of ICWS'05, IEEE, (2005) 293–301.

[20] Rabinovich, A.: *Complexity of equivalence problems for concurrent systems of finite agents.* In: Information and Computation (1997). volume 139, 111–129.

[21] Sardina, S., F. Patrizi, G. De Giacomo: *Behaviour composition in the presence of failure.* In: Proceedings of KR'08, AAAI, (2008). 640–650.

[22] Sawa, D.: *Equivalence Checking of Non-flat Systems Is EXPTIME-Hard.* In: Int. Conf. on Concurrency Theory. Springer (2003), 237–250.

[23] Singh, M., M. Huhns: *Service-Oriented Computing. Semantics, Process, Agents.* Wiley (2005).

[24] Trainotti, M., M. Pistore, G. Calabrese, G. Zacco, G. Lucchese, F. Barbon, P. Bertoli, P. Traverso: *Sipporting composition and execution of Web services.* In: Proceedings of ICSOC'05, Springer, (2005). 495–501.