Computational Analysis of Interacting Web Services: a Logical Approach

Philippe Balbiani, Fahima Cheikh-Alili

Computational Analysis of Interacting Web Services: a Logical Approach

Philippe Balbiani and Fahima Cheikh

Université Paul Sabatier, Institut de recherche en informatique de Toulouse
31062 Toulouse Cedex 9, France

Abstract. Web services composition is the interleaving of actions sequences in accordance with a client specification. In this paper we consider a Web services model where services are able to execute actions and send and receive messages. We define, for this model, the composition problem and study its decidability.

Keywords Web services, composition problem, decidability issues.

1 Introduction

Service oriented computing [13] 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 [1, 2, 11, 14]. 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 [1, 4]. 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 [11, 14]. In all cases, to compose services together is to interleave their actions sequences in accordance with a client specification. The composition problem is difficult to solve, as shown by theoretical complexity results obtained in the papers mentioned above [1, 4, 11, 14]. In this paper, we present and study a new model where services are able to execute actions and send and receive messages. The model we present in section 2 is a variant of the model proposed by [3]. In section 3, we define for this model the composition problem. We give, in section 4, theoretical results about its decidability. In section 5, we talk about interesting future works concerning the integration of security issues into Web services.

2 The model

In this section, we present information systems and Web services. Information systems are relational structures that allow the representation of a universe. Web services are conditional transition systems where the transitions correspond to a command execution, to a message's reception or to a message's emission. In subsection 2.3, we define two important notions: the client service and the mediator service.

2.1 Information systems

An information system can be seen as a set of objects characterized by a set of attributes. Formally, an information system is a structure of the form $IF = (Obj, Att, Val, f)$ where:

- Obj is a finite set of objects,
- Att is a finite set of attributes,
- Val is a nonempty set of values and
- $f : Obj \times Att \rightarrow 2^{Val}$ is a function which associates, to each object $o \in Obj$ and to each attribute $a \in Att$, a set $f(o, a) \subseteq Val$ of values.

If the value $v \in Val$ belongs to the set $f(o, a)$ then we say that v is a possible value of object o for attribute a.

Example 1. The information system defined by the following table describes in terms of attributes *name*, *price*, *composition*, *size* and *color* a set $\{o_1, o_2\}$ of manufactured goods.

f	*name*	*price*	*composition*	*size*	*color*
o_1	{*sweater*}	{40}	*acrylic*	{XL}	*blue*
o_2	{*skirt*}	{80}	{*wool*, *cotton* }	{S}	{*black*, *white*}

Table 1. Information system

2.2 Web services

Web services update information systems by executing commands. They also obtain information by communicating together. In this paper, Web services are considered as conditional transition systems: the transition from one state to another is possible only if certain conditions are satisfied. Formally, relatively to an information system $IF = (Obj, Att, Val, f)$, a Web service is a structure of the form $S = (Q, I, F, VarL, P, \delta)$ where:

- Q is a finite set of states,

- $I \subseteq Q$ is a set of initial states,
- $F \subseteq Q$ is a set of final states,
- $VarL$ is a finite set of local variables,
- P is a finite set of ports and
- δ is a transition function.

We will see, in section 3, how the local variables of S receive elements of Val as values. We will also see how services operate. $VarL$ variables and Val elements constitute the terms of S. A port is a structure of the form (M, d, m) where: M is the port's name, $d \in \{in, out\}$ is the port's type and $m \in \mathbb{N}$ is the port's size. If $d = in$ then (M, d, m) is an input port and if $d = out$ then (M, d, m) is an output port. The size m of the port indicates the length of messages received or sent through this port each time it is used. The transition function δ associates, to each pair (q, q') of states in S, a finite set $\delta(q, q')$ of possible transitions from q to q'. These transitions are structures of the form (C, α) where C is a logical expression and α is a finite sequence of primitive operations. Logical expressions are defined as follows:

- $C := \top \mid (\theta_1 = \theta_2) \mid (\theta \in f(z, a)) \mid Empty(M) \mid \neg C \mid (C_1 \wedge C_2) \mid \exists z\ C$

where θ, θ_1 et θ_2 are terms of S, z is a variable ranging over Obj, $a \in Att$ is an attribute and M is the name of an input port of S or of the client service. Empty(M) is satisfied when the port named M is empty, see subsection 2.4 for more details. Note that, contrary to our model, the transition function of services is deterministic in [3]. Primitive operations that can be used by S are defined as follows:

- create object z,
- destroy object z,
- add θ to $f(z, a)$,
- delete θ from $f(z, a)$,
- $x := \theta$,
- $?M(\theta_1, \ldots, \theta_m)$ and
- $!M(\theta_1, \ldots, \theta_m)$.

In these primitive operations, z is a variable ranging over Obj, $a \in Att$ is an attribute, $\theta, \theta_1, \ldots, \theta_m$ are terms of S, x is a local variable of S and M is the name of a port in S of size m. Of course, the primitive operation $?M(\theta_1, \ldots, \theta_m)$, that consists in receiving a package of m values by a port of S named M, has a meaning for S only if $(M, in, m) \in P$ and the primitive operation $!M(\theta_1, \ldots, \theta_m)$, that consists in sending a package of m values by a port of S named M, has a meaning for S only if $(M, out, m) \in P$. We assume that for each sequence α of primitive operations used by S, if α contains a primitive operation of the form $?M(x_1, \ldots, x_m)$ or of the form $!M(\theta_1, \ldots, \theta_m)$, then this sequence has length 1. Consequently, for each sequence α of primitive operations, one of the three following conditions is satisfied: (1) α is a sequence of primitive operations without exchange of messages, (2) α is a sequence of primitive operations composed

of only 1 primitive operation of the form $?M(\theta_1, \ldots, \theta_m)$ and (3) α is a sequence of primitive operations composed of only 1 primitive operation of the form $!M(\theta_1, ..., \theta_m)$. A transition (C, α) is atomique if the sequence α of its primitive operations is composed of only 1 primitive operation.

Example 2. The Web service $S_{goal} = (Q_g, I_g, F_g, VarL_g, P_g, \delta_g)$ described in Fig. 1 performs its computations on the information system $IF = (Obj, Att, Val, f)$ represented by Table 1. S_{goal} allows (1) to obtain the price, the composition, the size and the color of an object in the information system, (2) to update the information system by adding a new object under the condition that its price is equal to 100 and (3) to exchange messages between services.

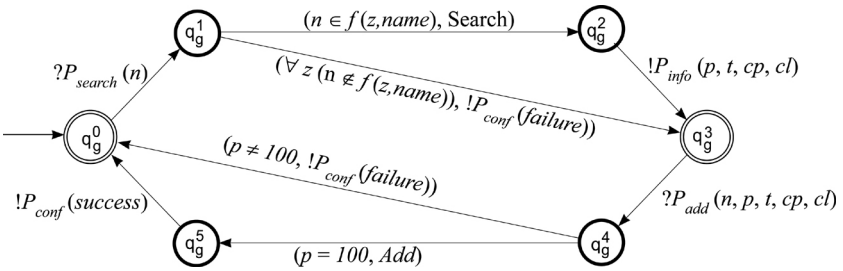

Fig. 1. Web service S_{goal}

In Fig. 1, z ranges over Obj, $name \in Att$ is an attribute, n, p, t, cp and cl are local variables of S_{goal}, success and failure are elements of Val and P_{search}, P_{info}, P_{add} and P_{conf} are names of ports of S_{goal}. $Search$ is the sequence of primitive operations defined as follows: $p := f(z, price)$; $t := f(z, size)$; $cp := f(z, composition)$; $cl := f(z, color)$ and Add is the sequence defined as follows: create object z; add n to $f(z, name)$; add p to $f(z, price)$; add cp to $f(z, composition)$; add t to $f(z, size)$; add cl to $f(z, color)$. We recall that $price$, $size$, $composition$ and $color$ are attributes in Att. To execute the transition $\delta(q_g^1, q_g^2)$ is first to check for the existence of an object $o \in Obj$ such that $n \in f(o, name)$ and second to execute the sequence $Search$ of primitive operations. To execute the transition $\delta(q_g^0, q_g^1)$ is to receive a package of one value by the port P_{search}. To execute the transition $\delta(q_g^2, q_g^3)$ is to send a package of four values by the port P_{info}.

2.3 Clients and mediators

Web services will be used by particular services called client services. Client services are services whose only purpose is to obtain informations about the information system. Consequently, two states are sufficient to completely define them. From the first state, only emission of packages of values can be performed and from the second state, only receptions of packages of values can

be performed. Logical expressions conditioning the client transitions have always the *true* value. More precisely, a client service is a Web service of the form $S_0 = (\{q_0^0, q_0^1\}, \{q_0^0\}, \{q_0^0\}, \emptyset, P_0, \delta_0)$ where the transition function δ_0 is as follows:

- $\delta_0(q_0^0, q_0^0) = \emptyset$,
- $\delta_0(q_0^1, q_0^1) = \emptyset$,
- $\delta_0(q_0^0, q_0^1)$ is a finite set of transitions of the form $(\top, !M(v_1, \ldots, v_m))$, where $(M, out, m) \in P_0$ and $v_1, \ldots, v_m$ are constants in Val and
- $\delta_0(q_0^1, q_0^0)$ is a finite set of transitions of the form $(\top, ?M(v_1, \ldots, v_m))$, where $(M, in, m) \in P_0$ and $v_1, \ldots, v_m$ are constants in Val.

Example 3. In Fig. 2, $P0_{search}$, $P0_{add}$, $P0_{conf}$and $P0_{info}$ are names of ports of S_0.

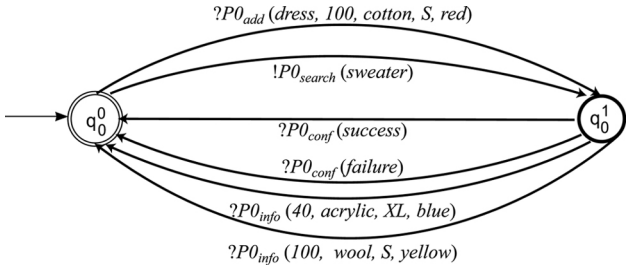

Fig. 2. Client service S_0

A mediator service carries out only exchanges of messages. Its role is to interpose itself between the client service and the available services. More precisely, a mediator service is a service of the form S_{med} =(Q_{med}, I_{med}, F_{med}, $VarL_{med}$, P_{med}, δ_{med}) where the transition function δ_{med} is such that $\delta(q, q')$ is a finite set of transitions of the form $(C, !M(\theta_1, \ldots, \theta_m))$ or of the form $(C, ?M(\theta_1, \ldots, \theta_m))$. The logical expressions conditioning the mediator transitions are as follows:

- $C := \top \mid (\theta_1 = \theta_2) \mid \neg C \mid (C_1 \wedge C_2)$

where θ_1 et θ_2 are terms of S_{med}.

Example 4. In Fig. 3, n', p', t', cp', cl' and res are local variables of the mediator service S_{med} and PM_{search}, $PM1_{search}$, $PM1_{info}$, PM_{info}, PM_{add}, $PM2_{add}$, $PM2_{conf}$ and PM_{conf} are names of ports of S_{med}.

2.4 Links

Let us consider a finite set $C = \{S_0, ..., S_n\}$ of services and let us denote by P_i, $i \in \{0, ..., n\}$, the set of ports of S_i. A C-link can be seen as a mean to associate

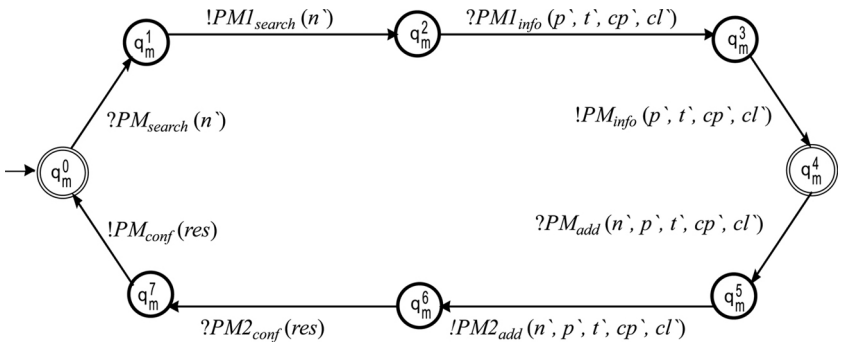

Fig. 3. Mediator service S_{med}

the input ports and the output ports of different services. Formally, a C-link is a binary relation L in $P_0 \cup ... \cup P_n$ such that:

- if (M, d, m) L (M', d', m') then $d = in$, $d' = out$ and $m = m'$,
- if (M, d, m) L (M', d', m') and (M, d, m) L (M'', d'', m'') then $M' = M''$ and
- if (M, d, m) L (M'', d'', m'') and (M', d', m') L (M'', d'', m'') then $M = M'$.

Consequently, if $(M_i, in, m) \in P_i$ and $(M_j, out, m) \in P_j$ are such that (M_i, in, m) L (M_j, out, m) then $S_i \neq S_j$, S_i can only receive, by the port named M_i, packages of m values from the port (M_j, out, m) whereas S_j can only send, by the port named M_j, packages of m values to (M_i, in, m). Moreover, we will see, in section 3, how a queue of packages, denoted $EntF(M_i, M_j)$, contains packages of values sent by (M_j, out, m) to (M_i, in, m) but not yet received. For S_i, to receive a package of m values on port (M_i, in, m) is to remove a first package of m values from the queue $EntF(M_i, M_j)$ while, for S_j, to send a package of m values on port (M_j, out, m) is to add a last package of m values to the queue $EntF(M_i, M_j)$. Note that if $(M_i, in, m)L(M_j, out, m)$ then we will say that $Empty(M_i)$ is true iff $EntF(M_i, M_j)$ is empty.

Example 5. We can take as an example a link L for S_0 and S_{goal} respectively described in Fig. 2 and Fig. 1. We consider $L = \{(P_{search}, P0_{search}), (P_{add}, P0_{add}), (P0_{conf}, P_{conf}), (P0_{info}, P_{info})\}$.

3 Execution trees

3.1 Definition

In what follows, we first define the notion of execution tree which allow to represent all computations performed by services from an information system and all exchanges of messages between services. Let us consider a finite set $C = \{S_0, ..., S_n\}$ of Web services and a C-link L. For all $i \in \{0, ..., n\}$, let us denote $S_i = (Q_i, I_i, F_i, VarL_i, P_i, \delta_i)$. A global state for C and L will be a structure of the form $\Delta = (IF, q_0, ..., q_n, int_0, ..., int_n, EntF, cl)$ where:

- $IF = (Obj, Att, Val, f)$, is an information system,
- for any $i \in \{0, ..., n\}$, $q_i \in Q_i$,
- for any $i \in \{0, ..., n\}$, int_i associates, to each local variable $x \in VarL_i$ of S_i, a value $int_i(x)$ in Val,
- $EntF$ is a function that associates to each pair $((M, in, m), (M', out, m))$ of L, a queue of packages of m values and
- cl is a finite set of values.

Let us note that, unlike [3], we do not limit the length of the queues defined by $EntF$. A global state for C and L gives us information about the value of the information system, the value of the services current states, the value of the local variables of each service and the value of queues where the packages of values already sent but not yet received are kept in stock. The field cl contains the set of all values that have already been received by the client up to now. To describe the way in which the global state changes, we introduce the concept of execution tree. An execution tree T for C and L is a tree whose nodes are labelled by global states and whose edges are labelled by the transitions performed by services. More precisely, the root of T is labelled by a global state of the form $\Delta_0 = (IF, q_0, ..., q_n, int_0, ..., int_n, EntF, cl)$ such that $q_0 \in I_0, ..., q_n \in I_n$ and for all links $((M, in, m), (M', out,), m') \in L$, $Ent(M, M') = \emptyset$. Moreover if $\Delta = (IF, q_0, ..., q_n, int_0, ..., int_n, EntF, cl)$ and $\Delta' = (IF', q'_0, ..., q'_n, int'_0, ..., int'_n, EntF', cl')$ are the labels of two consecutive nodes of T, then there exists $i \in \{0, ..., n\}$ such that for any $j \in \{0, ..., n\}$, if $i \neq j$ then $q'_j = q_j$ and $int'_j = int_j$ and one of the three following conditions is satisfied:

1. In $\delta_i(q_i, q'_i)$ there exists a transition of the form (C, α) without exchange of messages and there exists a substitution sub of the variables in (C, α) ranging over Obj such that:
   - $sub(C)$ has the "$true$" value for IF and int_i,
   - int'_i and IF' are obtained from int_i and IF by performing primitive operations of the sequence $int_i(sub(\alpha))$,
   - $EntF' = EntF$ and
   - $cl' = cl$,
2. In $\delta_i(q_i, q'_i)$ there exists a transition of the form $(C, ?M(\theta_1, \ldots, \theta_m))$ and there exists a substitution sub of the variables in C ranging over Obj such that:
   - $sub(C)$ has the "$true$" value for IF and int_i,
   - there exists a port (M', out, m) such that $((M, in, m), (M', out, m)) \in L$ and $EntF(M, M')$ is nonempty,
   - int'_i is obtained from int_i by unifying $\theta_1, \ldots, \theta_m$ with the m values of the first package in $EntF(M, M')$,
   - IF'=IF,
   - $EntF'$ is obtained from $EntF$ by removing from $EntF(M, M')$ the first packages of m values and
   - if $i = 0$ then cl' is obtained from cl by adding to it the first package of m values in $EntF(M, M')$ else $cl' = cl$,

3. In $\delta_i(q_i, q_i')$ there exists a transition of the form $(C, !M(\theta_1, ..., \theta_m))$ and there exists a substitution sub of the variables in C ranging over Obj such that:
   - $sub(C)$ has the "$true$" value for IF and int_i,
   - there exists a port (M', in, m) such that $((M', in, m), (M, out, m)) \in L$,
   - $int_i' = int_i$
   - IF'=IF,
   - $EntF'$ is obtained from $EntF$ by adding to $EntF(M, M')$ a last package of m values $(int_i(\theta_1), ..., int_i(\theta_m))$ and
   - $cl' = cl$.

In the first case, the edge (Δ, Δ') of T is labelled by $int_i(sub(\alpha))$. In the second case, it is labelled by $?M(int_i'(\theta_1), ..., int_i'(\theta_m))$. In the third case, it is labelled by $!M(int_i(\theta_1), ..., int_i(\theta_m))$.

3.2 Equivalence between execution trees

In order to compare computations performed by two distinct sets of services, we define the concept of equivalence between execution trees. More particularly, we are interested by the exchange of messages performed by the client and the sequences of primitive operations without exchanges of messages performed by available services. For this reason, we define in this section, the notion of reduced tree and the notion of equivalence between execution trees. Let us consider an execution tree T for a set $C = \{S_0, ..., S_n\}$ of Web services containing a unique client service S_0. The reduced tree of T will be the tree obtained from T by removing edges, in order to keep only edges labelled by transitions concerning transmissions of messages to a client service, receptions of messages by the client service and sequences of primitive operations without exchanges of messages performed by available services. Formally, the reduced tree T_r of T is built in the following way. The root does not change. Its label is $(IF, q_0, EntF^*, cl)$ where $EntF^*$ is the restriction of $EntF$ to ports concerning the client service S_0. If v_1 is a node in T and in T_r and $v_1, ..., v_n, v_{n+1}$, $n \geq 1$, is a path in T such that (1) for any integer $i \in \{1, ..., n-1\}$ labels of edges (v_i, v_{i+1}) are send or receptions of messages that do not concern the client and (2) the label of the edge (v_n, v_{n+1}) is either a sequence of primitive operations without exchange of messages, or a send of message by the client or a reception of message by the client, then we add to T_r the node v_{n+1} with the label $(IF', q_0', EntF'^*, cl')$ such that IF' is the value of the information system at the node v_{n+1}, q_0' is the state of the client service at the node v_{n+1}, $EntF'$ is the value of $EntF$ at the node v_{n+1}, $EntF'^*$ is the restriction of $EntF'$ to the ports concerning S_0 and cl' is the value of cl at the node v_{n+1}. We also add to T_r the edge (v_1, v_{n+1}) with the label of (v_n, v_{n+1}) deprived of the primitive operations of the form $x := \theta$.

Now, we define two kinds of equivalence between trees, the embedding equivalence and the weak equivalence. Two execution trees T and T' are embedding equivalent, denoted $T \subseteq T'$, when they are defined for sets of services containing the same client service and when T_r is included in T_r'. More precisely, if T is a

tree defined by a set of nodes V and a set of edges E and T' a tree defined by a set of nodes V' and a set of edges E' then T is included in T' if there exists an injective function $g : E \rightarrow E'$, which associates to each edge of E, an edge of E' such that:

- for any edge $e \in E$, its label is equal to that of the edge $g(e) \in E'$,
- the label of the initial node of $e \in E$ is equal to that of the initial node of $g(e) \in E'$ and
- the label of the final node of $e \in E$ is equal to that of the final node of $g(e) \in E'$.

Two execution trees T and T' are weakly equivalent, denoted $T \cong T'$, when they are defined for sets of services containing the same client service and when T_r and T'_r are similar. More precisely, let T and T' be trees, $Path$ be the set of all finite paths from the root in T, and $Path'$ be the set of all finite paths from the root in T'. Let us define the label of a path to be the concatenation of the labels of the edges composing this path. T and T' are similar if there exists a functions $g : Path \rightarrow Path'$, which associates to each path of $Path$, a path of $Path'$ and a function $h : Path' \rightarrow Path$ which associates to each path of $Path'$, a path of $Path$ such that:

- for any path p of $Path$, its label is equal to that of the Path $g(p)$,
- the label of the final node of $p \in Path$ is equal to that of the final node of $g(p)$,
- for any path p' of $Path\prime$, its label is equal to that of the Path $h(p')$ and
- the label of the final node of $p' \in Path\prime$ is equal to that of the final node of $h(p')$.

On the set of all trees the reader may easily verify that $\cong$ is an equivalence relations whereas $\subseteq$ is reflexive and transitive.

3.3 Web services composition problem

In this section, we define the embedding composition problem and the weak composition problem. When a client wants to performs computations from the information system and there is no available service which can performs alone these calculus, a solution to satisfy the client is to determine if there exists a composition of services that allows the execution of the computations. Thus, the Web services composition problem consists to find available services and to bind these services together. Formally, the embedding (resp. weak) composition problem is the decision problem defined as follows:

Input: a finite set $C = \{S_1, ..., S_n\}$ of services, a client service S_0, a goal service S_{goal} and a link L for S_0 and S_{goal},

Output: determine if there exists a mediator service S_{med}, a subset U of C, a link L' for S_0 and S_{med} and a link L'' for S_{med} and U such that for any information system IF, the execution tree for $\{S_0, S_{goal}\}$ and L, denoted $tree(S_0, S_{goal}, L, IF)$ is embedding (resp. weakly) equivalent to the execution tree for $\{S_0, S_{med}\} \cup U$ and $L' \cup L''$, denoted $tree(S_0, S_{med}, L', U, L'', IF)$.

Example 6. Let us consider the following instance of the Web services composition problem. The set $C = \{S_1, S_2\}$ where S_1 and S_2 are described in Fig. 4 and perform their computations on the information system described in Table 1. , the service S_{goal} described in Fig. 1, the service S_0 described in Fig. 2 and the link L considered as an example at the end of subsection 2.4. Let $L' = \{(PM_{search}, P0_{search}), (PM_{add}, P0_{add}), (P0_{conf}, PM_{conf}), (P0_{info}, PM_{info})\}$ be a link for S_0 and S_{med}. Let $L'' = \{(P1_{search}, PM1_{search}), (PM1_{info}, P1_{info}), (P2_{add}, PM2_{add}), (PM2_{conf}, P2_{conf})\}$ be a link for S_{med} and U. It is easy to verify that S_{med}, described in Fig. 3, $U = C$, L' and L'' are a solution to the problem when the weakly equivalence or the embedding equivalence are considered.

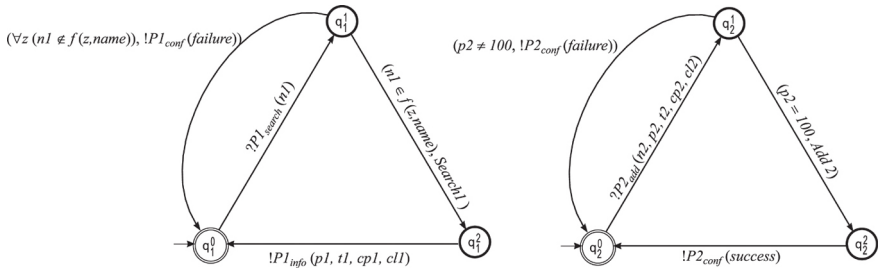

Fig. 4. Web services S_1 and S_2

In Fig. 4, $n1$, $p1$, $t1$, $cp1$ and $cl1$ are local variables of S_1 and $P1_{search}$, $P1_{conf}$ and $P1_{info}$ are names of ports of S_1. The sequence of primitive operations $Search1$ is as follows: $p1 := f(z1, price)$; $t1 := f(z1, size)$; $cp1 := f(z1, composition)$; $cl1 := f(z1, color)$.

In Fig. 4, $n2$, $p2$, $t2$, $cp2$ are $cl2$ are local variables of S_2 and $P2_{add}$ and $P2_{conf}$ are names of ports of S_2. The sequence of primitive operations $Add2$ is as follows: create object z; add $n2$ to $f(z, name)$; add $p2$ to $f(z, price)$; add $cp2$ to $f(z, composition)$; add $t2$ to $f(z, size)$; add $cl2$ to $f(z, color)$.

4 Decidability results

In this section, we give some results about the decidability of the embedding composition problem and the weakly composition problem. We prove that these two problems are undecidable in general. However, if some restrictions are considered, we prove that the weakly composition problem becomes decidable.

Theorem 1. *The embedding composition problem is undecidable.*

Proof. We prove this theorem by reducing the uniform halting problem of Minsky machines [10], which is undecidable, to the embedding composition problem (see annex for details).

Theorem 2. *The weakly composition problem is undecidable.*

Proof. The following decision problem, called 0-halting problem, is known [10] to be undecidable:

Input: a Minsky machine M
Output: does M halt when the initial values of the registers r and s are 0?

As for theorem 1, to prove theorem 2, we reduce the 0-halting problem of Minsky machines to the weakly composition problem (see annex for details).

Our next goal is to characterize special cases of services such that there is an algorithm to solve the composition problem. Let us consider the following restrictions:

- There is no condition in the transitions of services.
- Length of queues are limited to at most 1 message.
- There is no primitive operations of the form "destroy object z" or "$x := \theta$".
- Service mediator has at most k states and b ports.
  This restrictions are neither stronger nor weaker than the restrictions considered in [3].

Theorem 3. *The weakly composition problroblem is decidable, when the restrictions above are considered.*

Proof. In order to simplify the proof, we assume that (1) for all ports (M, d, m) in the considered services, $m = 0$ and (2) the transitions of services in C and the transitions of S_{goal} are atomic. One could easily show that our line of reasoning still applies when this assumption is lifted. Let U be a subset of C, S_{med} be a mediator service with at most k states and at most b ports, L' be a link for S_0 and S_{med} and L'' be a link for S_{med} and U. Seeing that services in C do not contain primitive operations of the form "destroy object z" or "$x := \theta$", the reader may easily verify that the following conditions are equivalent:

- for all information systems IF, $tree(S_0, S_{goal}, L, IF) \cong tree(S_0, S_{med}, L', U, L'', IF)$,
- for an arbitrary information system IF containing at least one object, $tree(S_0, S_{goal}, L, IF) \cong tree(S_0, S_{med}, L', U, L'', IF)$.

Let us consider an arbitrary information system IF containing at least one object and define $T = tree(S_0, S_{goal}, L, IF)$, $T' = tree(S_0, S_{med}, L', U, L'', IF)$. Let $L(T)$ (resp. $L(T')$) be the set of all finite sequences of labels corresponding to the ports in T (resp. T').

Lemma 1. *The languages $L(T)$ and $L(T')$ are rational.*

Proof. See annex.

Using the above lemma, one can elaborate a decision procedure solving the weak composition problem as follows:

1. Given S_0, S_{goal}, C and L, choose non deterministically a subset U of C, a mediator service S_{med} with at most k states and b ports, a link L' for S_0 and S_{med} and a link L'' for S_{med} and U.
2. Choose an arbitrary information system IF containing at least one object.
3. Compute the automata A and A' recognizing the languages $L(T)$ and $L(T')$ associated to $T = tree(S_0, S_{goal}, L, IF)$ and $T' = tree(S_0, S_{med}, L', U, L'', IF)$.
4. Decide if A and A' recognize the same languages or not.

This completes the proof of theorem 3.

5 Conclusion

We have seen how Web services are at the origin of a new paradigm of distributed programming which modifies the way the applications are specified, implemented and run. We have defined the problem of their composition and gave its complexity. However, the services oriented applications put challenges which must be raised, in particular at the level of the data protection [8].What are these challenges? Generally, in the practice, Web services interact together and with their clients by means of cryptographic protocols, to obtain their certificates and characterize their rights. Languages as WS-Policy [7] and WS-Security Policy [6] allow each Web service to express its safety policies at the level of the exchanged messages. More exactly, these languages allow to specify which certificates have to be added to messages and which cryptographic primitive have to be used in messages. Specification languages for safety policies as Rei [9] allow to specify standards of behavior by using the deontic concepts of prohibition, obligation and permission. There are also works which consider the specification of the composed service [5]. It thus seems interesting to define a high-level language allowing the expression, in terms of prohibition obligation and permission of safety policies for Web services. The search for the compatibility between policies is situated at this highest level. We are thinking of the integration of such language in our model. This integration will allow the expression of the access conditions to the information system by the services and their clients. To what extent these access conditions an influence on the complexity of the composition problem ? Until which point is it possible to modify these access conditions only by running the product of Web services composition? Services are independent software elements which can be composed in order to make collaborate distributed applications. In some case this collaboration causes information flow between services or between services and their client. How is it possible to control this flow? Is it possible to apply techniques developed in the context of concurrent programming to our model?

Acknowledgement

We have realised this work within the framework of the project “Composition des politiques et des services” (Cops) financially supported by the GIP ANR under the program ARA SSIA.

References

1. D. Berardi. *Automatic Service Composition. Models, Techniques and Tools.* Phd *La Sapienza University*, Roma, 2005.
2. D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini and M. Mecella. Synthesis of underspecified composite e-services based on automated reasoning. In *Proc. of the 2nd Int. Conf. on Service Oriented Computing, ICSOC 2004*, 105-114, 2004.
3. D. Berardi, D. Calvanese, G. De Giacomo, R. Hull and M. Mecella. Automatic composition of transition-based semantic Web services with messaging. In *Proc. 31st Int. Conf. Very Large Data Bases, VLDB 2005*, 613-624, 2005.
4. D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini and M. Mecella. Automatic services composition based on behavioral descriptions. *Int. Journal of Cooperative Information Systems*, **14**, 333-376, 2005.
5. A. Charfi and M. Mezini. Using aspects for security engineering of Web service compositions. In *Proc. ICWS 2005*, Pages 59-66.
6. IBM and Al. Web Services Security Policy Language (WS-Security Policy). Décembre 2002.
7. IBM et al. Web Services Policy Frameworck (WS-Policy). Septembre 2002.
8. H. Kadima and V. Monfort. *Les Web Services, Techniques, Démarches et Outils*, Dunod, 2003.
9. L. Kagal, T. Finin and A. Joshi. Declarative policies for describing Web service capabilities and constraints. In *Proc. W3C Workshop on constraintes and capabilities for Web Services*, Octobre 2005.
10. M. Minsky. Computation Finite and Infinite Machines. Prentice-Hall, 1967.
11. M. Pistore, A. Marconi, P. Bertoli and P. Traverso. Automated composition of Web services by planning at the knowledge level. In *Proc. Int. Joint Conf. on Artificiel Intelligence, IJCAI 2005*, 1252-1259, 2005.
12. M. Pistore, P. Traverso and P. Bertoli. Automated composition of Web services by planning in asynchronous domains. In *Proc. Int. Conf. on Automated Planning and Schedulling, ICAPS 05*, 2-11, 2005.
13. M. Singh and M. Huhns. *Service-Oriented Computing. Semantics, Process, Agents.* Wiley, 2005.
14. P. Traverso and M. Pistore. Automated composition of semantic Web services into executable processes. In *Proc. 3rd Int. Semantic Web Conf.*, 2004.

Annex

Proof of theorem 1. We prove this theorem by reducing the uniform halting problem of Minsky machines [10], which is undecidable, to the embedding composition problem. For the sake of completeness, let us say that a Minsky machine M consists of 2 registers r and s taking their values in $\mathbb{N}$ together with a finite set $\{I_1, ..., I_n\}$ of operations of the form:

- r^+,
- s^+,
- $r^-(m)$,
- $s^-(m)$,

and a halting operation $I_{n+1} = halt$. If $I_i = r^+$ (resp. $I_i = s^+$) then to execute I_i is to increment register r (resp. s) and to go to the next operation I_{i+1}. If $I_i = r^-(m)$ (rep. $I_i = s^-(m)$) then to execute I_i is to decrement register r (resp. s) and to go to the next operation I_{i+1} if the current value of r (resp. s) is not equal to 0. In case that the current value of r (resp. s) is 0, then to execute I_i is simply to go to the next operation I_m. M stops when it reaches operation I_{n+1}. The reduction is as follows. Let us consider a Minsky machine M. The corresponding instance of the embedding composition problem is defined as follows. Let us consider the information system $IF = (\{o_1\}, \{a\}, \{r_1, s_1, t\}, f)$ where the set of objects contains 1 object o_1, a is an attribute, r_1 (resp. s_1) is a value associated to the register r (resp. s) and $f(o_1, a)$ is empty. Let S_0 be the client service described in Fig. 5.

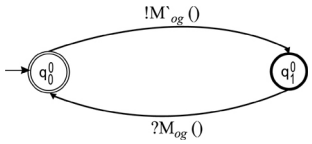

Fig. 5. Web service S_0

Let S_g be the goal service defined relatively to IF and described in Fig. 6. Let

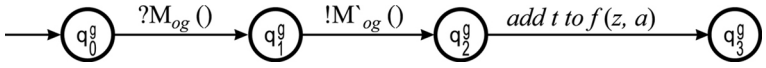

Fig. 6. Web service S_{goal}

$L = \{(M_{0g}, M'_{g0}), (M_{g0}, M'_{0g})\}$ be a link for S_0 and S_g, and $C = \{S_1\}$ be the set of available services defined relatively to IF, where $S_1 = (Q_1, I_1, F_1, VarL_1, P_1, \delta_1)$ is such that:

- $Q_1 = \{q_1, ..., q_n, q_{n+1}, q'_1, q'_{n+1}\}$,
- $I_1 = \{q'_0\}$,
- $F_1 = \emptyset$,
- $VarL_1 = \emptyset$,
- $P_1 = \{(M_{1med}, in, 0)\}$,

the transition function δ_1 of S_1 is defined as follows:

- $\delta_1(q'_1, q_1) = \{(T, ?M_1())\}$
- $\delta_1(q_{n+1}, q'_{n+1}) = \{(T, \mathsf{add}\ t\ \mathsf{to}\ f(z,a))\}$
- for all $i \in \{1, ..., n\}$:
  - if $I_i = r^+$ then $\delta_1(q_i, q_{i+1}) = \{(T,\ \mathsf{create\ object}\ z;\ \mathsf{add}\ r_1\ \mathsf{to}\ f(z,a))\}$ and for all $j \in \{1, ..., n\}$, if $j \neq i+1$ then $\delta_1(q_i, q_j) = \emptyset$
  - if $I_i = s^+$ then $\delta_1(q_i, q_{i+1}) = \{(T,\ \mathsf{create\ object}\ z;\ \mathsf{add}\ s_1\ \mathsf{to}\ f(z,a))\}$ and for all $j \in \{1, ..., n\}$, if $j \neq i+1$ then $\delta_1(q_i, q_j) = \emptyset$
  - if $I_i = r^-(m)$ then $\delta_1(q_i, q_{i+1}) = \{((r_1 \in f(z,a)),\ \mathsf{delete}\ r_1\ \mathsf{from}\ f(z,a))\}$, $\delta_1(q_i, q_m) = \{(\forall z(r_1 \notin f(z,a)), \mathsf{nil})\}$ and for all $j \in \{1, ..., n\} \setminus \{i+1, m\}$, $\delta_1(q_i, q_j) = \emptyset$
  - if $I_i = s^-(m)$ then $\delta_1(q_i, q_{i+1}) = \{((s_1 \in f(z,a)),\ \mathsf{delete}\ s_1\ \mathsf{from}\ f(z,a))\}$, $\delta_1(q_i, q_m) = \{(\forall z(s_1 \notin f(z,a)), \mathsf{nil})\}$ and for all $j \in \{1, ..., n\} \setminus \{i+1, m\}$, $\delta_1(q_i, q_j) = \emptyset$

To understand better the reduction, we represent the service S_1 by Fig. 7.

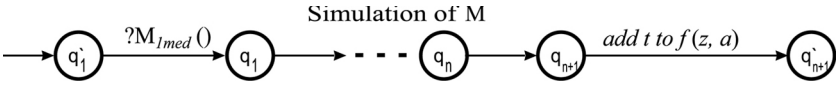

Fig. 7. Web service S_1

We have to prove now that M halts whatever the initial values of r and s are iff there exists $U \subseteq C$, there exists a mediator S_{med} and links L', L'' such that for all information systems IF', $tree(S_0,\ S_{goal}, L, IF') \subseteq tree(S_0, S_{med}, L', U, L'',\ IF')$.

For the left to right implication, we suppose that the Minsky machine M halts whatever the initial values of r and s are. Let us consider $U = \{S_1\}$, S_{med} represented by Fig. 8, $L' = \{(M_{med0}, M'_{0b}),\ (M_{0b}, M'_{med0})\}$ and $L'' = \{(M_{1med}, M'_{med1})\}$. The intuition here is that, to simulate the execution of its last command "add t to $f(z,a)$", S_1 has to be in the state q_{n+1}. Obviously, the service S_1 can reach the state q_{n+1} from the state q_0 only if the the Minsky machine M can reach the halt instruction from its initial instruction I_0. The service S_1 can reach the state q_1 only when it receives a message, that is why we use the mediator service S_{med} which also sends a message to the service S_1 and simulates communications between the client service and the goal service. Consequently, for an arbitrary IF', $tree(S_0,\ S_{goal}, L, IF') \subseteq tree(S_0,\ S_{med}, L',\ U, L'', IF')$.

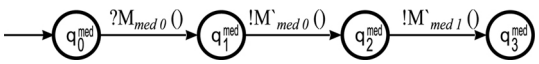

Fig. 8. Web service S_{med}

For the right to left implication, suppose that M does not halt for some initial values n_1, n_2 of r, s. The two possible cases for U are $U = \emptyset$ and $U = \{S_1\}$. Let us consider that $U = \emptyset$. In this case, let S_{med} be an arbitrary mediator, L' be an arbitrary link for S_0 and S_{med}, L'' be the empty link and IF' be an information system containing at least one object. Obviously, the edges of $tree(S_0, S_{med}, L', U, L'', IF')$ are labelled only by commands executed by S_{med} and by S_0. As for $tree(S_0, S_{goal}, L, IF')$, it contains an edge labelled by a command composed only of the primitive operation "add t to $f(z, a)$". Hence, when $U = \emptyset$ there is no S_{med} and L' such that for all information systems IF', $tree(S_0, S_{goal}, L, IF') \subseteq tree(S_0, S_{med}, L', U, \emptyset, IF')$. Now, let us consider that $U = \{S_1\}$. Let IF'' be the information system containing $n1$ objects with the value r_1 for the attribute a and $n2$ objects with the value s_1 for the attribute a. $tree(S_0, S_{goal}, L, IF'')$ contains an edge labelled by a command composed only of the primitive operation "add t to $f(z, a)$". As for $tree(S_0, S_{med}, L', U, L'', IF'')$, where S_{med}, L' and L'' are arbitray, it contains edges labelled by commands executed by the mediator service S_{med}, by the client service S_0 and by the service S_1. Among these services, only the service S_1 can execute the primitive operation "add t to $f(z, a)$". Thus the service S_1 must move from q_1 to q_{n+1} wich is not possible, seeing that M does not halt when given the initial values n_1, n_2. Hence, for any S_{med}, L' and L'', there exits IF'' such that $tree(S_0, S_{goal}, L, IF') \not\subseteq tree(S_0, S_{med}, L', U, L'', IF''))$. This completes the proof of theorem 1.

proof of theorem 2. As for theorem 1, to prove theorem 2, we reduce the 0-halting problem of Minsky machines to the weakly composition problem. The reduction is as follows. Let us consider a Minsky machine M with a finite set $\{I_1, ..., I_n\}$ of operations of the form r^+, r^+, $r^-(m)$, $s^-(m)$ and a halting operation I_{n+1}=halt. The corresponding instance of the weakly composition problem is defined as follows. Let S_0 be the client service described in Fig. 1. Let S_g be the goal service described in Fig. 9.

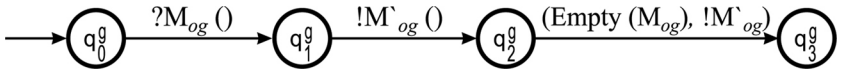

Fig. 9. Web service S_{goal}

Let $L = \{(M_{0g}, M'_{g0}), (M_{g0}, M'_{0g})\}$ be a link of S_0 and S_g and $C = \{S_1\}$ be a set of available services, where $S_1 = (Q_1, I_1, F_1, VarL_1, P_1, \delta_1)$ such that:

- $Q_1 = \{q_1, ..., q_n, q_{n+1}, q'_1, q'_{n+1}, q''_{n+1}, q'''_{n+1}\}$,
- $I_1 = \{q'_1\}$,
- $F_1 = \emptyset$,
- $VarL_1 = \emptyset$,
- $P_1 = \{(M_{1med}, in, 0),\ (M'_{1med}, out, 0),\ (M_1, in, 0),\ (M'_1, out, 0),\ (M_2, in, 0),\ (M'_2, out, 0)\}$,

the transition function δ_1 of S_1 is defined as follows:

- $\delta_1(q'_1, q_1) = \{(T, ?M_{1med}())\}$
- $\delta_1(q_{n+1}, q'_{n+1}) = \{(T, !M'_{1med}())\}$
- $\delta_1(q_{n+1}, q'_{n+1}) = \{(T, ?M_{1med}())\}$
- $\delta_1(q_{n+1}, q'_{n+1}) = \{((Empty(M_{0g})), !M'_{1med}())\}$
- for all $i \in \{1, ..., n\}$:
  - if $I_i = r^+$ then $\delta_1(q_i, q_{i+1}) = \{(T, !M'_1())\}$ and for all $j \in \{1, ..., n\}$, if $j \neq i + 1$ then $\delta_1(q_i, q_j) = \emptyset$
  - if $I_i = s^+$ then $\delta_1(q_i, q_{i+1}) = \{(T, !M'_2())\}$ and for all $j \in \{1, ..., n\}$, if $j \neq i + 1$ then $\delta_1(q_i, q_j) = \emptyset$
  - if $I_i = r^-(m)$ then $\delta_1(q_i, q_{i+1}) = \{(T, ?M_1())\}$, $\delta_1(q_i, q_m) = \{(Empty(M_1)), \mathsf{nil})\}$ and for all $j \in \{1, ..., n\} \setminus \{i + 1, m\}$, $\delta_1(q_i, q_j) = \emptyset$
  - if $I_i = s^-(m)$ then $\delta_1(q_i, q_{i+1}) = \{(T, ?M_2())\}$, $\delta_1(q_i, q_m) = \{(Empty(M_2), \mathsf{nil})\}$ and for all $j \in \{1, ..., n\} \setminus \{i + 1, m\}$, $\delta_1(q_i, q_j) = \emptyset$

To understand better the reduction, we represent the service S_1 by Fig. 10. Following the live of reasoning suggested in the proof of the theorem 1, we may

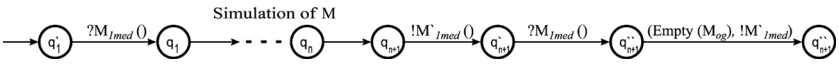

Fig. 10. Web service S_1

show that when the initial values of r and s are 0, M halts iff there exists $U \subseteq C$, there exists a mediator S_{med} and links L', L'' such that for all information systems IF', $tree(S_0,\ S_{goal}, L, IF') \cong tree(S_0,\ S_{med},\ L', U, L'', IF')$. This completes the proof of theorem 2.

proof of lemma 1. Let $W_L = \{0, 1\}^{|L|}$ where $\mid L \mid = card(L)$. Each element in L is denoted L_j, $j = 1, ..., \mid L \mid$. Let $w_L \setminus j = ({w_L^1}', ..., {w_L^{|L|}}')$ be associated to $w_L \in W_L$ as follows: $w_L \setminus j$ has the same value as w_L for all its components except for the j^{th} component. More precisly, if $w_L^j = 0$ then ${w_L^j}' = 1$ and if $w_L^j = 1$ then ${w_L^j}' = 0$. Let us consider the finite automaton $A = (\Sigma, Q, I, F, \delta)$ defined as follows.

- $\Sigma = \{$create object z, add v to $f(z,a)$, delete v from $f(z,a)$, $?M()$, $!M'()\}$,
- $Q = Q_0 \times Q_g \times W_L$,
- $I = (q_0^0, q_g^0, (0, ..., 0))$,
- $F = Q$,
- $\forall a \in \Sigma \setminus \{?M(), !M'()\}$:
  $\delta_a(q_0, q_g, w_L) = (q_0, q'_g, w_L)$ if $\delta_g(q_g, q'_g) = (T, a)$,
- $\delta_{?M()}(q_0, q_g, (w_L^1, ..., w_L^j, ..., w_L^{|L|}))$ is defined if $w_L^j = 1$ and $L_j = (M, M')$ in which case it is equal to $(q'_0, q_g, w_L \setminus j)$ if M is an input port of S_0 such that $\delta_0(q_0, q'_0) = (T, ?M())$ or it is equal to $(q_0, q'_g, w_L \setminus j)$ if M is an input port of S_{goal} such that $\delta_g(q_g, q'_g) = (T, ?M())$
- $\delta_{!M'()}(q_0, q_g, (w_L^1, ..., w_L^j, ..., w_L^{|L|}))$ is defined if $w_L^j = 0$ and $L_j = (M, M')$ in which case it is equal to $(q'_0, q_g, w_L \setminus j)$ if M' is an output port of S_0 such that $\delta_0(q_0, q'_0) = (T, !M'())$ or it is equal to $(q_0, q'_g, w_L \setminus j)$ if M' is an output port of S_{goal} such that $\delta_g(q_g, q'_g) = (T, !M'())$.

We recall that Q_0 and Q_g are respectively sets of states in S_0 and S_{goal}, q_0^0 and q_g^0 are respectively initial states in S_0 and S_{goal}, L is a link for S_0 and S_{goal}, $v \in Val$, M is an input port in $P_0 \cup P_g$, M' is an output port in $P_0 \cup P_g$. The reader may easily verify that $L(T)$ is equal to the language recognized by A. A similar construction can be obtained for $L(T')$. This ends the proof of lemma 1.