LABELED GENERALIZED STOCHASTIC PETRI NET BASED APPROACH FOR WEB SERVICES COMPOSITION

Sofiane Chemaa[1], Mouna Bouarioua[2] and Allaoua chaoui[3]

MISC Laboratory, Department of Computer Science and its applications, University of Constantine 2,Constantine, Algeria
[1]chemaa@misc-umc.org
[2]bouarioua@misc-umc.org
[3]a_chaoui2001@yahoo.com

ABSTRACT

A Web Service is a component written in any language, deployed on any platform, which has a standard wrapping layer based on XML.The component can interact with other applications which themselves comply with the Web Services standards. Generally, a single service does not satisfy the users needs that are more and more complex. Therefore, services must be made able to becomposed to build new value added services. This process is called Web services Composition. This latter is a critical subject that requires formal techniques for its completion. In this paper, we show how basic and existent services can be composed to create a composite service which offers a new functionality. In this context, we propose an expressive Labeled generalized stochastic Petri net based algebra that succeeds in the complex Web services composition. Basic and advanced constructs which are supported by the proposed algebra are syntactically and semantically defined.

KEYWORDS

Web Services, Web services composition, LGSPN, Algebra, formal techniques.

1. INTRODUCTION

Nowadays, web services present a new vision of the provision of software components for information systems. The concept of Web service essentially refers to an application made available on Internet by a service provider and accessible by clients through standard Internet protocols such asSimple Object Access Protocol (SOAP) [1], Universal Description, Discovery and Integration (UDDI) [2], and Web Service Description Language (WSDL) [3].The composition of Web services is a natural evolution of this technology and has enormous potential in reorganizing business-to-business or enterprise application integration. Current technologies based on WSDL, UDDI and SOAP offer solutions for description, publication, discovery and interoperability of Web services; but do not accomplish their complex composition that remains a very complex task, and requires formal techniques for its accomplishment.

In this paper we address the Web service composition problem using a Petri net based framework called Labeled Generalized Stochastic Petri Net (LGSPN) [4]. Compared to other approaches, this concept offer efficient and powerful mechanisms for complex systems modeling which are not supported even by high level Petri nets. In this context, we propose a LGSPN based algebra for composing Web services that successfully solves complex composition. The proposed

approach provides not only a formalism to describe Web services structure and behavior but also a representative set of operators presented by means of syntax, semantics and the resulting composite service. The choice of the LGSPN models is justified by many reasons. For example:

- The use of visual modeling techniques such as LGSPN in the design of complex Web services helps us to easily understand the modeled services.
- The modeling based on LGSPN allows the specification and prototyping of complex Web services with taking into account the time constraints.
- $LGSPN_s$ provide a formalism that allows the construction of models that can be used for behavioral analyses based on the classical theory of Petri nets, and also for Performance analyses based on the time specifications and the probabilistic models.

The remainder of this paper is organized as follows. Web service modeling and specification using LGSPN are presented in Section 2. An expressive algebra that permits to compose Web services modeled with $LGSPN_s$ is given in Section 3. In Section 4, we give a brief overview of some related work. Finally, we discuss in Section 5 some prospects and work in progress.

2. Web services as Labeled generalized stochastic Petri nets

A Petri net (PN) [5] isa graphical and mathematical modeling tool. This is a directed bipartite graph with two types of nodes: places are represented by circles and transitions are represented by rectangles. The arcs of the graph connect places and transitions in such a way that places can only be connected to transitions and vice versa. Places in a Petri net may contain a discrete number of tokens. The distribution of tokens over the places is called a marking.

When Carl Adam Petri has introduced Petri nets in his thesis for the first time, they had served to describe concurrent systems in terms of cause / effect relations without considering the notion of time. The introduction of temporal concepts into Petri nets was proposed several years later by other researchers. The transition firing is the result of either a logical condition becoming true in the system, or the achievement of an activity. The latter interpretation is the reason for associating timing with transitions. The resulting paradigm from the introduction of temporal concepts into classical PN is a sub model called stochastic Petri nets (SPN) [4]. When stochastic timing is mixed with deterministic null delays, we obtain generalized stochastic Petri net models (GSPN). In the case where each transition is labeled, the model is called labeled generalized stochastic Petri nets (LGSPN). For more details about the LGSPN, the reader is referred to [4] and [6].

Web services are assimilated to a distributed system; that consists of a set of loosely coupled modules, which communicate through messages exchange. The behavior of a web service is defined by a set of operations. Thus, modeling Web services using LGSPN is straightforward. Every operation in the service is modeled by a transition; the state of the web service is modeled by the position of tokens in the LGSPN and the arrows between places and transitions are used to specify causal relations.

In the following, we give some formal definitions about LGSPN-Service and Web service.

2.1. LGSPN-Service

To simplify and facilitate the web services modeling using LGSPN models, we have made some modifications to the LGSPN notation defined in [6]. As a result we present the LGSPN-Service Models.

A LGSPN-Service is an 11-tuple $LGSPN - S = (P, T, \Pi, I, O, H, W, M_0, i, o, L)$ where:

- $P$is a finite and non-empty set of places.
- $T = IT \cup TT$is a finite set of transitions where: IT is a set of immediate transitions denoted by a black rectangle. TT is a set of timed transitions denoted by an empty rectangle.
- Π: $T \rightarrow \mathbb{N}$is the priority function. It associates a priority level to each transition. Let$t_i \in T$, $\Pi(t_i) = 0$ if t_i.type = TT and $\Pi(t_i) > 0$ if t_i.type = IT. (i.e. the firing of an immediate transition has priority over the firing of a timed transition).
- I, O, H: $T \rightarrow Set(p) \mid Set(p) \subseteq P$are the input, output, and inhibition functions.
- W: $T \rightarrow \mathbb{R}$ is a function defined on the set of transition. W allows the definition of the stochastic process associated with the model.
- $M_0$is the initial marking of the LGSPN-Service.
- $i$is the input place with $\bullet i = \{x \in T \mid i \in O(x)\} = \emptyset$.
- ois the output place with $o\bullet = \{x \in T \mid o \in I(x)\} = \emptyset$.
- $L : P \rightarrow O \cup \{\tau\}$is a labeling function where O is a set of operation names andτ is a silent operation (unobservable operation). In this work, we have modeled a silent transition (st) by an immediate transition with $\Pi(st) = n \mid n \in \mathbb{N}^*$ and$W(st) = V \mid V \in \mathbb{R}$.

In figure 1(a), we propose an example that illustrates the principle of LGSPN-Service models. The service reproduces the behavior of a translation system. After reading an English text, the system activates two other local processes in parallel. The first process allows to translate the text in French. The other process enables the translation of the same text in Spanish. Once the two translations are completed, a correctness check process is activated.

In this example we have 4 temporal transitions (t1, t3, t4, t5) and an immediate transition (t2). This latter represents the action of the activation of the two translation processes in parallel. This action is modeled by an immediate transition because its execution time is negligible.

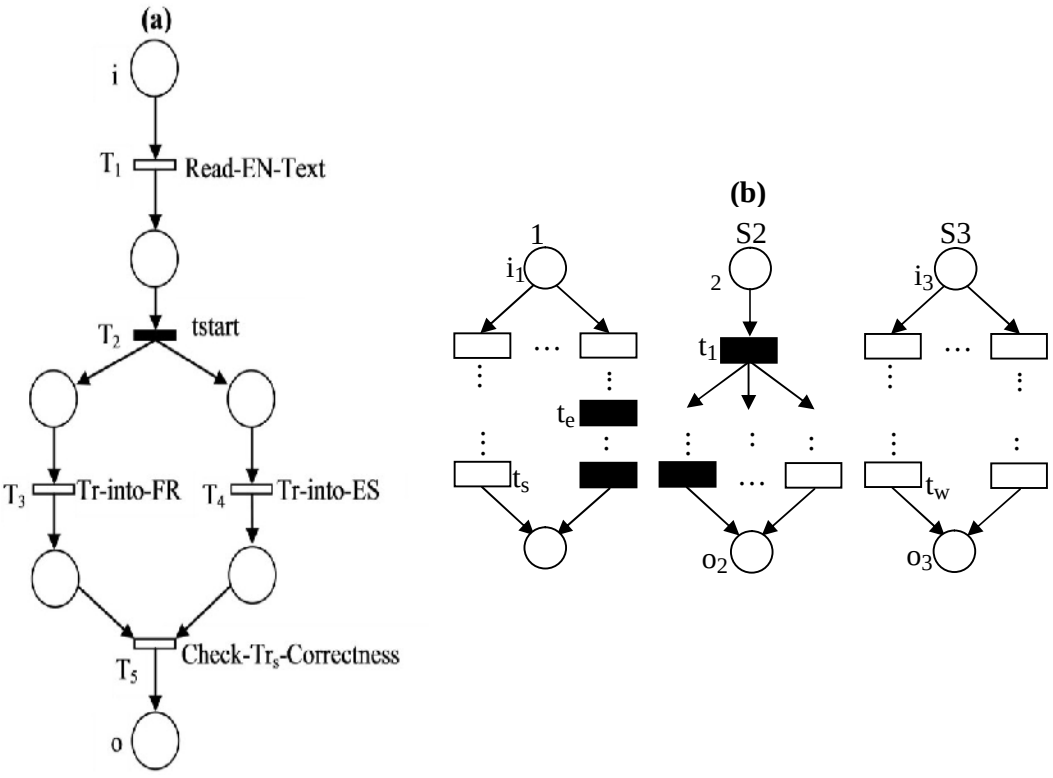

Figure 1. LGSPN-Service Example (a); Services S1,S2,S3 (b)

2.2. Web Service

A Web service is a tuple [7] where:

- NameS is the name of the service used as its unique identifier.
- Desc is the description of the provided service. It summarizes what functionalities the service offers.
- Loc is the server in which the service is located.
- URL is the invocation of the Web service.
- CS is a set of the component services of the Web service, if CS = {NameS} then S is a basic service, otherwise S is a Composite service.
- is the LGSPN-Service modeling the dynamic behavior of the service.

In the next section, we propose an algebra that permits to incrementally compose existing web services modeled by$LGSPN_s$. The aim is to create a new composite service that provides new functionalities.

3. Web services Composition

Web services as presented, are components, conceptually limited, with relatively simple functionalities, which are modeled by a set of operations [8]. Given two or more Web services, each with a specific task,they sometimes cooperate in order to achieve a new task.This will result in a new value added service. For example aservice of hotel booking can collaborate with a Web mappingservice like Google Maps API to show the location of hotels tothe customers. The

collaboration of these services generates acomposed Web service which performs the original individualtasks as well as a new one.

In this section we present an algebra that combines existing Web services for building more complex ones. We will take Sequence, Parallel, Alternative, Iteration and Arbitrary Sequence as basicconstructs. Moreover, two more developed constructs which are Discriminator and refinement are defined. After that, each operator formal semantics in terms of LGSPN-Service is given.

3.1. LGSPN-Service based algebra

The BNF-like notation below describes a grammar defining the set of services that can be generated using algebra's operators.

$S::= \varepsilon \mid X \mid S \blacktriangleright S \mid S \blacktriangleleft \blacktriangleright S \mid \circlearrowleft S \mid S \Leftrightarrow S \mid S//S \mid (S \boxdot S) \gg S \mid Ref(S, a, A).$

Let $S_i = (NameS_i, Desc_i, Loc_i, URL_i, CS_i, SLGSPN_i)$ with $SLGSPN_i = (P_i, T_i, \Pi_i, I_i, O_i, H_i W_i, M_{0i}, i_i, o_i, L_i)$ for i = 1...3 be three Web Services such that $P_i \cap P_j = \emptyset$ and $T_i \cap T_j = \emptyset$ for $i \neq j$.

Empty Service (ε): The empty service ε is a service that performs no operation. It is used for technical and theoretical reasons.

Definition 1 The Empty service is defined as $\varepsilon = (NameS, Desc, Loc, URL, CS, SLGSPN)$ where:

- $NameS$ = Empty.
- $Desc$ = "Empty Web Service".
- Loc = Null, stating that there is no server for the service.
- URL = Null, stating that there is no invocation for the service.
- CS = {Empty}
- $SLGSPN = (p, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, p, p, \emptyset)$.

In Figure 2(a), we show the graphic representation of the empty service (ε) in terms of LGSPN-Service.

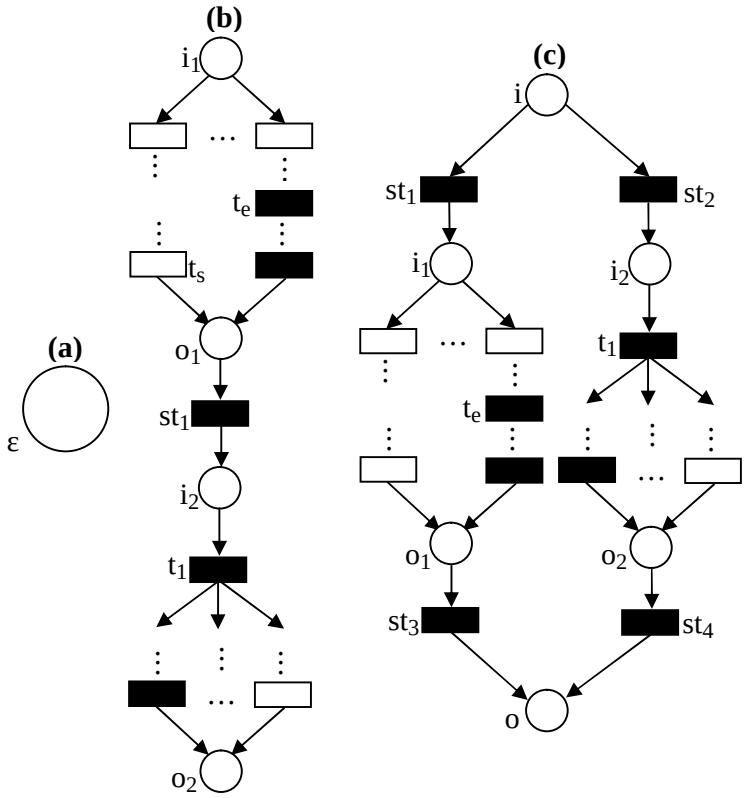

Figure 2. Empty Service(a); Sequence Service (b); Alternative Service (c)

Sequence (►): The sequence operator permitsto execute two services successively. This operator can be used in the case where a service depends on the execution results of another service. For example, in an online purchase operation, the service "*Order*" must be executed before the service "*Payment*".

Definition 2 The service$S_1 \blacktriangleright S_2$ is defined as$S_1 \blacktriangleright S_2 = (NameS, Desc, Loc, URL, CS, SLGSPN)$ where:

- $CS = CS_1 \cup CS_2$
- $SLGSPN = (P, T, \Pi, I, O, H, W, M_0, i, o, L)$ where:
  $P = P_1 \cup P_2, T = T_1 \cup T_2 \cup \{st_1\}, \Pi = \Pi_1 \cup \Pi_2 \cup \{(st_1, n) | n \in \mathbb{N}^*\}, I = I_1 \cup I_2 \cup \{(st_1, o_1)\},$
  $O = O_1 \cup O_2 \cup \{(st_1, i_2)\}, \quad H = H_1 \cup H_2, \quad W = W_1 \cup W_2 \cup \{(st_1, V) | V \in \mathbb{R}\},$
  $M_0 = M_{01} \cup M_{02}, i = i_1, o = o_2, L = L_1 \cup L_2 \cup \{(st_1, \tau)\}.$

Given the two services S_1 and S_2 shown in Figure 1(b), the composite service$S_1 \blacktriangleright S_2$ is represented by the LGSPN-Service shown in Figure 2(b).

Alternative (◄►): Given two services S_1 andS_2, the alternative operator reproduces either the behavior of S_1 orS_2, but not both. This operator is also called "*choice operator*".

Definition 3 The service $S_1 \blacktriangleleft\blacktriangleright S_2$ is defined as $S_1 \blacktriangleleft\blacktriangleright S_2 = (NameS, Desc, Loc, URL, CS, SLGSPN)$ where:

- $CS = CS_1 \cup CS_2$
- $SLGSPN = (P, T, \Pi, I, O, H, W, M_0, i, o, L)$ where:
  $P = P_1 \cup P_2 \cup \{i, o\}$, $T = T_1 \cup T_2 \cup \{st_1, st_2, st_3, st_4\}$, $\Pi = \Pi_1 \cup \Pi_2 \cup \{(st_1, n_1), (st_2, n_2), (st_3, n_3), (st_4, n_4)\}$, $I = I_1 \cup I_2 \cup \{(st_1, i), (st_2, i), (st_3, o_1), (st_4, o_2)\}$, $O = O_1 \cup O_2 \cup \{(st_1, i_1), (st_2, i_2), (st_3, o), (st_4, o)\}$, $H = H_1 \cup H_2$, $W = W_1 \cup W_2 \cup \{(st_1, V_1), (st_2, V_2), (st_3, V_3), (st_4, V_4) | V_1, V_2, V_3, V_4 \in \mathbb{R}\}$, $M_0 = M_{01} \cup M_{02}$, $L = L_1 \cup L_2 \cup \{(st_1, \tau), (st_2, \tau), (st_3, \tau), (st_4, \tau)\}$

Given the two services S_1 and S_2 shown in Figure 1(b), the composite service $S_1 \blacktriangleleft\blacktriangleright S_2$ is represented by the LGSPN-Service shown in Figure 2(c).

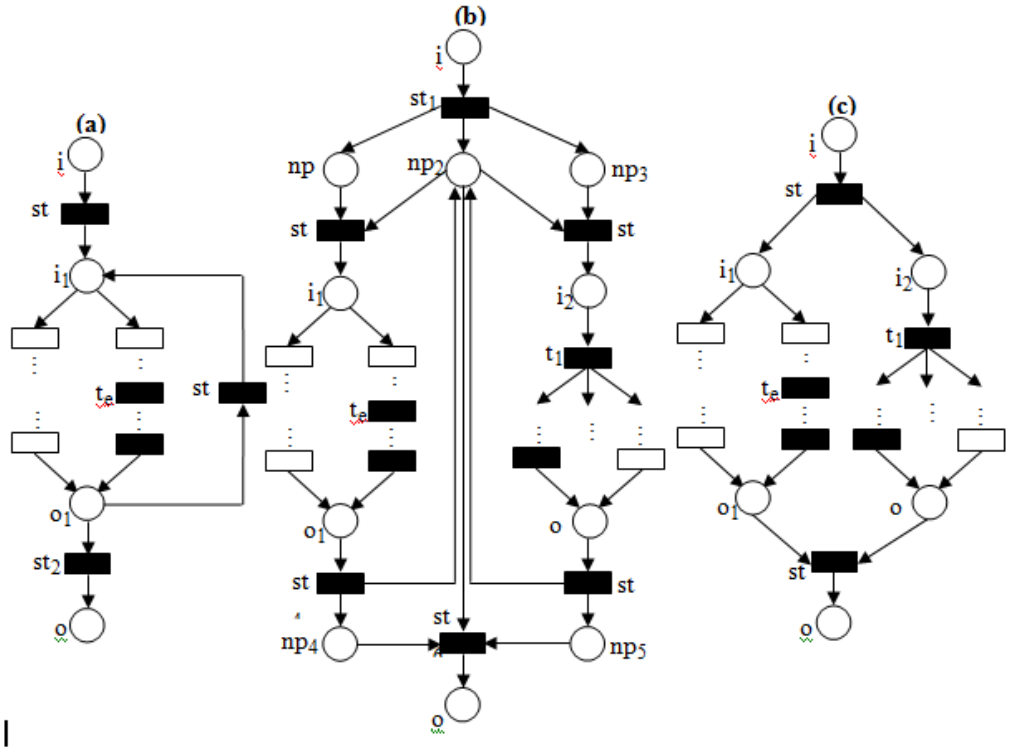

Figure 3. Iteration Service(a); Arbitrary Sequence Service (b); Parallel Service (c)

Iteration (↺): The iteration operator allows the service S to be performed a certain number of times in a row. For example, this operator can be used in the case where a client performs several successive orders.

Definition 4 The service $\circlearrowleft S_1$ is defined as $\circlearrowleft S_1 = (NameS, Desc, Loc, URL, CS, SLGSPN)$ where:

- $CS = CS_1$
- $SLGSPN = (P, T, \Pi, I, O, H, W, M_0, i, o, L)$ where:
  $P = P_1 \cup \{i, o\}$, $T = T_1 \cup \{st_1, st_2, st_3\}$, $\Pi = \Pi_1 \cup \{(st_1, n_1), (st_2, n_2), (st_3, n_3)\}$, $I = I_1 \cup \{(st_1, i), (st_2, o_1), (st_3, o_1)\}$, $O = O_1 \cup \{(st_1, i_1), (st_2, o), (st_3, i_1)\}$,

$H = H_1, W = W_1 \cup \{(st_1, V_1), (st_2, V_2), (st_3, V_3) | V_1, V_2, V_3 \in \mathbb{R}\}, M_0 = M_{01},$
$L = L_1 \cup \{(st_1, \tau), (st_2, \tau), (st_3, \tau)\}$

If we consider the service S_1 shown in Figure 1(b), the composite service $\circlearrowleft S_1$ is represented by the LGSPN-Service shown in Figure 3(a).

Arbitrary Sequence ($\Leftrightarrow$): Given two services S_1 andS_2, the Arbitrary Sequence operator builds thecomposite service $((S_1 \blacktriangleright S_2) \blacktriangleleft\blacktriangleright (S_2 \blacktriangleright S_1))$. The latter is useful when the services execution order is not important and there are no benefits to execute services in parallel.

Definition 5 The service $S_1 \Leftrightarrow S_2$ is defined as $S_1 \Leftrightarrow S_2 = (NameS, Desc, Loc, URL, CS, SLGSPN)$ where:

- $CS = CS_1 \cup CS_2$
- $SLGSPN = (P, T, \Pi, I, O, H, W, M_0, i, o, L)$ where:
  $P = P_1 \cup P_2 \cup \{i, o, np_1, np_2, np_3, np_4, np_5\}, T = T_1 \cup T_2 \cup \{st_1, st_2, st_3, st_4, st_5, st_6\},$
  $\Pi = \Pi_1 \cup \Pi_2 \cup \{(st_1, n_1), (st_2, n_2), (st_3, n_3), (st_4, n_4), (st_5, n_5), (st_6, n_6)\}, I = I_1 \cup I_2 \cup \{(st_1, i), (st_2, np_1), (st_2, np_2), (st_3, np_2), (st_3, np_3), (st_4, o_1), (st_5, o_2), (st_6, np_4), (st_6, np_5), (st_6, np_2)\}, O = O_1 \cup O_2 \cup \{(st_1, np_1), (st_1, np_2), (st_1, np_3), (st_2, i_1), (st_3, i_2), (st_4, np_2), (st_5, np_2), (st_4, np_4), (st_5, np_5), (st_6, o)\}, H = H_1 \cup H_2, W = W_1 \cup W_2 \cup \{(st_1, V_1), (st_2, V_2), (st_3, V_3), (st_4, V_4), (st_5, V_5), (st_6, V_6) | V_1, V_2, V_3, V_4, V_5, V_6 \in \mathbb{R}\}, M_0 = M_{01} \cup M_{02}, L = L_1 \cup L_2 \cup \{(st_1, \tau), (st_2, \tau), (st_3, \tau), (st_4, \tau) (st_5, \tau) (st_6, \tau)\}$

Given the two services S_1 and S_2 shown in Figure 1(b), the composite service $S_1 \Leftrightarrow S_2$ is represented by the LGSPN-Service shown in Figure 3(b).

Parallel ($//$): Given two services S_1 andS_2, the parallel operator builds a composite service performing the two services (S_1 andS_2) in parallel. The accomplishment of the resulting service is achieved when the two services are completed.

Definition 6 The service $S_1 // S_2$ is defined as $S_1 // S_2 = (NameS, Desc, Loc, URL, CS, SLGSPN)$ where:

- $CS = CS_1 \cup CS_2$
- $SLGSPN = (P, T, \Pi, I, O, H, W, M_0, i, o, L)$ where:
  $P = P_1 \cup P_2 \cup \{i, o\}, T = T_1 \cup T_2 \cup \{st_1, st_2\}, \Pi = \Pi_1 \cup \Pi_2 \cup \{(st_1, n_1), (st_2, n_2)\},$
  $I = I_1 \cup I_2 \cup \{(st_1, i), (st_2, o_1), (st_2, o_2)\}, O = O_1 \cup O_2 \cup \{(st_1, i_1), (st_1, i_2), (st_2, o)\},$
  $H = H_1 \cup H_2, \quad W = W_1 \cup W_2 \cup \{(st_1, V_1), (st_2, V_2) | V_1, V_2 \in \mathbb{R}\}, M_0 = M_{01} \cup M_{02},$
  $L = L_1 \cup L_2 \cup \{(st_1, \tau), (st_2, \tau)\}$

Given the two services S_1 and S_2 shown in Figure 1(b), the composite service $S_1 // S_2$ is represented by the LGSPN-Service shown in Figure 3(c).

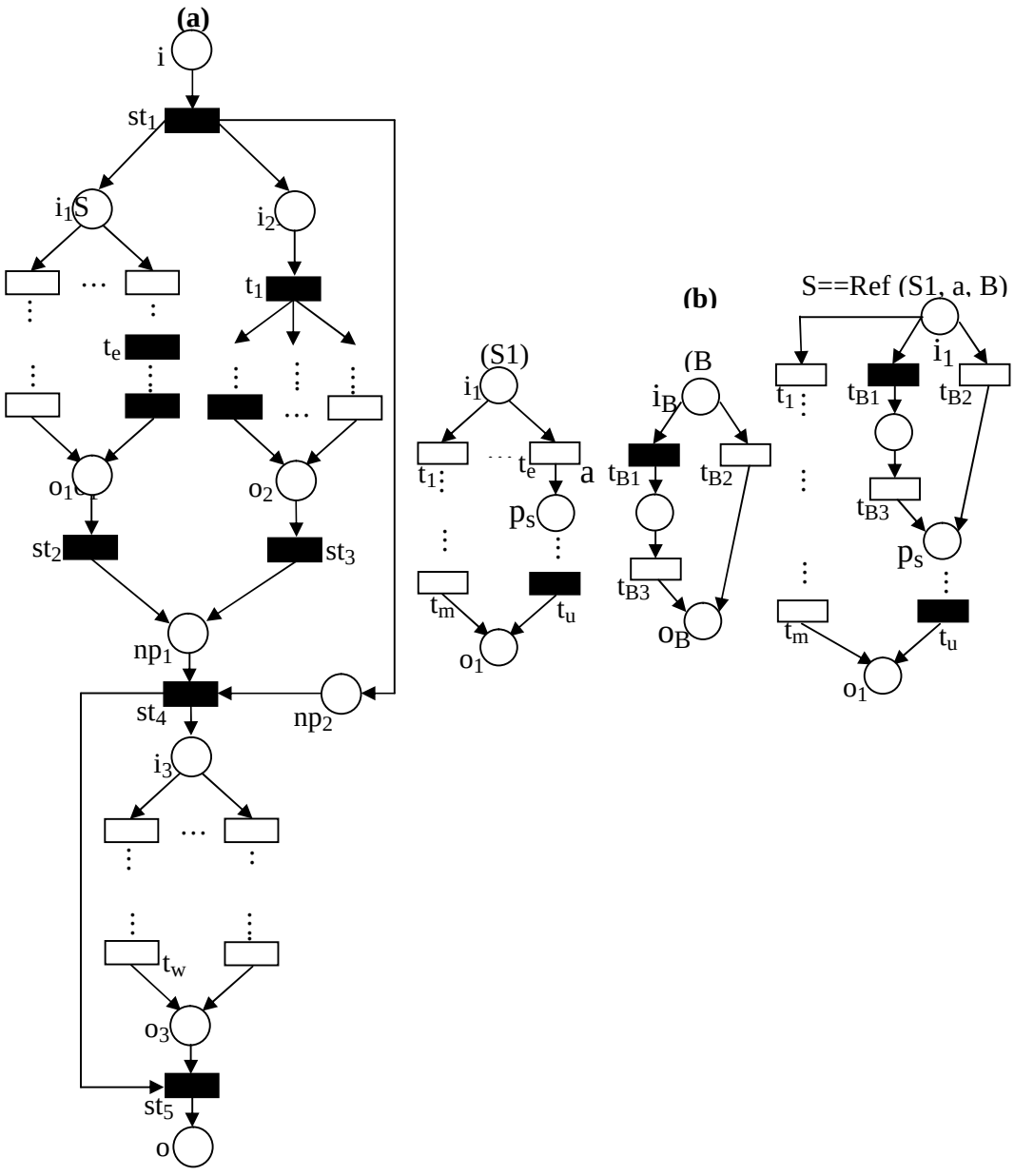

Figure 4. Discriminator Service (a); An abstract exampleof Refinement (b)

Discriminator (($\boxdot$) $\gg$**):**The discriminator operator allows tosolicit different services that perform the same task. The first service that responds activates another service. The other late responses will be ignored. The discriminator operator allows to benefit from services that respond in optimal time.

Definition 7 The service $(S_1 \boxdot S_2) \gg S_3$ is defined as$(S_1 \boxdot S_2) \gg S_3 = (NameS, Desc, Loc, URL, CS, SLGSPN)$ where:

- $CS = CS_1 \cup CS_2 \cup CS_3$
- $SLGSPN = (P, T, \Pi, I, O, H, W, M_0, i, o, L)$ where:

$$P = P_1 \cup P_2 \cup P_3 \cup \{i, o, np_1, np_2\}, \quad T = T_1 \cup T_2 \cup T_3 \cup \{st_1, st_2, st_3, st_4, st_5\},$$
$$\Pi = \Pi_1 \cup \Pi_2 \cup \Pi_3 \cup \{(st_1, n_1), (st_2, n_2), (st_3, n_3), (st_4, n_4), (st_5, n_5), \}, I = I_1 \cup I_2 \cup I_3 \cup \{(st_1, i), (st_2, o_1), (st_3, o_2), (st_4, np_1), (st_4, np_2), (st_5, np_1), (st_5, o_3)\}, O = O_1 \cup O_2 \cup O_3 \cup \{(st_1, i_1), (st_1, i_2), (st_1, np_2), (st_2, np_1), (st_3, np_1), (st_4, i_3), (st_5, o)\}, H = H_1 \cup H_2 \cup H_3, W = W_1 \cup W_2 \cup W_3 \cup \{(st_1, V_1), (st_2, V_2), (st_3, V_3), (st_4, V_4), (st_5, V_5) | V_1, V_2, V_3, V_4, V_5 \in \mathbb{R}\}, M_0 = M_{01} \cup M_{02} \cup M_{03}, L = L_1 \cup L_2 \cup L_3 \cup \{(st_1, \tau), (st_2, \tau), (st_3, \tau), (st_4, \tau) (st_5, \tau)\}$$

Given the servicesS_1,S_2 and S_3 shown in Figure 1(b), the composite service ($S_1 \boxdot S_2$) $\gg S_3$ is represented by the LGSPN-Service shown in Figure 4(a).

Refinement (Ref): The refinement operator allows to modify certain service operations and to replace them by other more detailed operations. This operator allows to change the service abstraction level from a high level to a lower level more concrete.

Definition 8 The service Re $Ref(S, a, B)$ is defined as $Ref(S, a, B) = (NameS, Desc, Loc, URL, CS, SLGSPN)$ where:

- $CS = \{CS_1 \cup CS_B if a \in L_1(T_1), \quad CS_1 otherwise.$
- $SLGSPN = (P, T, \Pi, I, O, H, W, M_0, i, o, L)$ where:

$$P = P_1 \cup P_B \setminus \{i_B, o_B | \bullet i_B = \emptyset \wedge o_B \bullet = \emptyset\} \; if a \in L_1(T_1), P_1 otherwise. \; T = T_1 \setminus L_1^{-1}(a) \cup T_B if a \in L_1(T_1), T_1 otherwise.$$
$$\Pi = \Pi_1 \setminus \{\Pi_1(t) \mid t \in L_1^{-1}(a)\} \cup \Pi_B if a \in L_1(T_1), \Pi_1 otherwise.$$
$$I = I_1 \setminus \{(t,p) \mid p \in P_1 \wedge t \in T_1 \wedge t \in L_1^{-1}(a)\} \cup I_B \setminus \{(t, i_B) | \; t \in T_B \wedge i_B \in P_B \wedge \bullet i_B = \emptyset\} \cup \{(t,p) | \; t \in T_B \wedge t \in i_B \bullet \wedge p \in P_1 \wedge (\exists \, tr \in T_1 \wedge tr \in L_1^{-1}(a) \wedge (tr, p) \in I_1)\} if a \in L_1(T_1), I_1 otherwise.$$
$$O = O_1 \setminus \{(t,p) \mid p \in P_1 \wedge t \in T_1 \wedge t \in L_1^{-1}(a)\} \cup O_B \setminus \{(t, o_B) | \; t \in T_B \wedge o_B \in P_B \wedge o_B \bullet = \emptyset\} \cup \{(t,p) | \; t \in T_B \wedge t \in \bullet o_B \wedge p \in P_1 \wedge (\exists \, tr \in T_1 \wedge tr \in L_1^{-1}(a) \wedge (tr, p) \in O_1)\} if a \in L_1(T_1), O_1 otherwise.$$
$$H = H_1 \setminus \{(t,p) | \; p \in P_1 \wedge t \in T_1 \wedge t \in L_1^{-1}(a)\} \cup H_B \setminus \{(t, i_B) | \; t \in T_B \wedge i_B \in P_B \wedge \bullet i_B = \emptyset\} \cup \{(t,p) | \; t \in T_B \wedge t \in i_B \bullet \wedge p \in P_1 \wedge (\exists \, tr \in T_1 \wedge tr \in L_1^{-1}(a) \wedge (tr, p) \in H_1)\} if a \in L_1(T_1), H_1 otherwise.$$
$$W = W_1 \setminus \{W_1(t) \mid t \in L_1^{-1}(a)\} \cup W_B if a \in L_1(T_1), W_1 otherwise.$$
$$M_0 = M_{01} \cup M_{0B} \setminus \{M_{0B}(i_B), M_{0B}(o_B)\} \; if a \in L_1(T_1), M_{01} otherwise.$$
$$i = i_1, o = o_1, \quad L = L_1 \setminus \{(t,a) | t \in T_1\} \cup L_B if a \in L_1(T_1), L_1 otherwise.$$

Figure 4(b) shows an abstract example of a refined service.

$n_1, n_2, n_3, n_4, n_5, n_6 \in \mathbb{N}^*$ and $V_1, V_2, V_3, V_4, V_5, V_6 \in \mathbb{R}$.
The proposed algebra verifies the closure property. This latter ensures that the product of any operation on services is itself a service to which we can apply algebra operators.

4. Related work

During the past years, several approaches for Web service composition have been suggested in literature. All of these propositions try to provide languages, semantic models, and platforms in order to propose efficient solutions for this problem.

Composition languages, such as BPEL [9] consist to provide a set of primitives that allows interaction between services being composed. Regrettably, it comes to textual and executable languages designed to satisfy the composite web service implementation phase that neglects, actually, the specification step, which is important, because it facilitates the global

comprehension of the system, and the development task. Furthermore, since they lack of the formalism, the formal analysis of the proposed languages is impossible.

Many researches are devoted to the modeling and the composition of web services. Most of these works provide models expressed in different formalisms.

In [10], Reiko Heckel, and Marc Lohmann have proposed to use the notion of contracts. The last mentioned refer to graph transformation rules. They are characterized by the assertions expressing the providers, or the customers' rights and duties. In addition, this approach uses the UML notions [11], signatures, and data models to add the behavioral information for the services web specification. The graph transformation rules have ability, to bring advanced operational interpretations that cannot be expressed with simple logical expressions,This approach has some limitations when it comes to a web services complex composition.

Process algebras are a mathematical formalism for describing and studying the concurrent systems. Several researchers have used different process algebras to specify and compose web services. For example, Brogi et al. [12] have proposed a formalization technique based on CCS, for the choreography of web services. In [13], semantics of the orchestration language BPEL is, this time, specified by using pi-calculus. Nevertheless, this work does not deal with some parts of BPEL, such as data management; and this is not surprising, because pi-calculus does not allow data manipulation.

IN [14], the authors have proposed an approach that allows the composite web servicedevelopment following MDA principles using UML 1.4 as a modeling means. [15] has used the same idea; this work is based on UML 2.0. However, the authors have pointed out that the model is not expressive enough. Besides, it is worth to note that UML is a semi-formal language.

In the field of the formal systems specification, automata are very recognized models. The use of automata for web services modeling and composition has been suggested by several researchers. The Columbo model [16], which is based on FSA (Finite State Automata), is one of these approaches. In [17], Fan et al. have proposed to model the web services using the AFA (Alternating finite automata). An interaction with the developer to provide, in advance, a detailed specification of the composite service, is required by these approaches. The use of DFSA (deterministic finite state automata) is proposed by the authors, in [18], to solve this problem. In this approach, the developers only define the exactness constraints on the composition. The behavior of the composite service is automatically synthesized. The use of input output automata (I/O automata) for modeling the web services composition, has been suggested by Mitra et al., in [19].

Actually, the disadvantage of using automata is that the competition modeling is not allowed by their semantics. A solution for this problem has been proposed by Yan and Dague, in [20], the idea is to model each parallel execution branch by independent automaton, and define events to realize their interconnection; but, unfortunately, this requires the input and the output state duplication in each parallel branch.

Concerning Petri-nets, a complete translation of BPEL in Petri-nets has been presented by Onyang et al., in [21]. Moreover, the ability of the last mentioned in modeling the web services composition has also been demonstrated by them. Similarly to [7], our approach is a Petri Net based framework for Web services composition. [7] proposed a Petri Net Algebra. Its model is enough expressive, but the data types cannot be distinguishable, because an elementary Petri net model is used. The work of [22] also deals with this problem, by modeling Web services and their composition using Colored Petri nets [23]. The proposal of [24] is also based on modeling the process of Web service composition by a kind of Object-Oriented Petri Nets. However, our

approach is formally defined and is based on a well-founded framework namely LGSPN. In [25] the authors have proposed an approach for modeling and composition of web services using G-Nets, which are a kind of high-level Petri nets. The main advantage of our approach compared to [7], [22],[24] and [25], is the fact that the model used in our work takes into account the time constraints.$LGSPN_s$ are models that can be used for behavioral analyses based on the classical theory of Petri nets, and also for Performance analyses based on the time specifications and the probabilistic models.

Ontology-driven approaches for Web services composition like OWL-S [26], SAWSDL [27] and [28] use terms from preagreed ontologies to declare preconditions and effects of the concerned services. In the two first works, the inputs and outputs are expressed by concepts, while [28] describes them in terms of instance-based graph patterns. If these approaches present the advantage of clearly understanding the meaning of the messages, their main drawback remains the difficulty to discover the explicit goal of the services. This latter constitutes a key element when composing by AI planners [29].

5. Conclusion

In this paper, we have presented a simple yet powerful approach, which offers solutions for both modeling web services and composing them. This approach has the major advantage of benefiting from the formal aspect of Labeled Generalized Stochastic Petri Nets (LGSPN). Indeed, modeling based on LGSPN allows the specification and prototyping of complex Web serviceswith taking into account the time constraints. A LGSPN-Servicebased algebra for Web service composition is developed. In this vein, we define the formal semantics of the composition operators by means of LGSPN-Service. Using this underlying framework provides a rigorous approach to verify the properties and detect inconsistencies between web services.

In a future work, we plan to extend our approach with advanced operators that can support more complex Web service combination. Another perspective of our work is to develop a Java prototype tool implementing the introduced operators. We may also use reduction techniques to optimize the models before the analysis and the verification of certain properties by using appropriate tools such as GreatSPN [30].

REFERENCES

[1] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, “Simple object access protocol (soap) 1.1”, May 2000, [Online]. Available:http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[2] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, “Unraveling the web services web an introduction to soap, wsdl, and uddi”, IEEE INTERNET COMPUTING, pp. 86-93, Mar./Apr.2002

[3] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, “Web services description language (wsdl) 1.1”, Mar 2001, [Online]. Available: http://www.w3.org/TR/wsdl.

[4] M. AjmoneMarsan, G. Balbo, G. Conte, S. Donatelli and G. Franceschinis, “AnIntroduction to generalized stochastic Petri nets”, In International Journal of Microelectronics and reliability: Special issue on Petri nets and related graph models, Vol. 31(4), pp 699-725, Pergamon Press,1991.

[5] C. A. Petri, “Kommunikationmitautomaten (in german)”, Ph.D. dissertation, University of Bonn, Germany, 1962.

[6] M. AjmoneMarsan, G. Balbo, G. Conte, S. Donatelli and G. Franceschinis, “Modeling with generalized stochastic Petri nets”, In J. Wiley, 1995.

[7] R. Hamadi and B. Benatallah, “A Petri net based-model for web service composition”, in proc. the 14th Australasian database conference, adelaide. Darlinghurst: Australian

[8] T. MELLITI, “Interopérabilité des services Web complexes. Application aux systèmes multi-agents (in french)”, Ph.D. dissertation, University Paris IX Dauphine 2004.
[9] F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S.Weerawarana, “Business Process Execution Language for Web Service (BPEL4WS) 1.0”, Published on the World Wide Web by BEA Corp, IBM Corp and Microsoft Corp, Aug 2002.
[10] R. Heckel and M. Lohmann, “Towards contract based testing of web service”, in Electronic Notes in Theoretical Computer Science 116, (2005), pp. 145–156.
[11] S. Thö ne, R. Depke, and G. Engels, Process- oriented, flexible composition of web services with uml, in Proc. the Joint Workshop on Conceptual Modeling Approaches for e-Business (eCOMO), 2002.
[12] A. Brogi, C. Canal, E. Pimentel, and A. Vallecillo, Formalizing web service choreographies, Electron. Notes Theor. Comput. Sci., 105 (2004), pp. 73–94.
[13] R. Lucchi and M.Mazzara, A pi-calculus based semantics for ws-bpel, Journal of Logic and Algebraic Programming, 70 (2007), pp. 96–118.
[14] R. Grønmo and I. Solheim, Towards modeling web service composition in uml, in Proc. 2nd International Workshop on Web Services Modeling, Architecture and Infrastructure (WSMAI-2004), 2004, pp. 72–86.
[15] V. De Castro, E. Marcos, and M. L. Sanz, Service composition modeling: A case study, in Proc. the Seventh Mexican International Conference on Computer Science (ENC 06), San Luis Potosi, Mexico, Sep 2006, pp. 101–1081.
[16] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella, Automatic composition of transition-based semantic web services with messaging, in Proc. the 31st international conference on Very large data bases (VLDB 05), 2005, pp. 613–624.
[17] W. Fan, F. Geerts, W. Gelade, F. Neven, and A. Poggi, Complexity and composition of synthesized web services, in Proc. the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (POD08), New York, NY, USA, 2008, pp. 231–240.
[18] J. P. Huai, T. Deng, X. X. Li, Z. X. Du, and H. P. Guo, Autosyn: A new approach to automated synthesis of composite web services with correctness guarantee, Science in China Series F, 59, Number 9 (2009), pp. 1534–1549.
[19] S. Mitra, R. Kumar, and S. Basu, Automated choreographer synthesis for web services composition using i/o automata, in Proc. IEEE International Conference Web Services (ICWS 2007), Jul 2007, pp. 364–371.
[20] Y. Yan and P. Dague, Modeling and diagnosing orchestrated web service processes, in Proc. IEEE International Conferenceon Web Services (ICWS 2007), Jul 2007, pp. 51–59.
[21] C. Ouyang, E. Verbeek, W. M. P. van der Aalst, S. Breutel, M. Dumas, and A. H. M. terHofstede, Formal semantics and analysis of control flow in ws-bpel, Science of Computer Programming, 67, Number 2-3 (2007), pp. 162–198.
[22] Z. Zhang, F. hong, and H. xiao, A colored petri net-based model for web service composition, Journal of Shanghai University (English Edition), 12, Number 4 (2008), pp. 323–329.
[23] K. Jensen, Coloured petri nets- a high level language for system design and analysis, in Lecture Notes in Computer Science 483. Advances in Petri Nets 1990 Springer-verlag, 1990.
[24] X. Feng, Q. Liu, and Z.Wang, A web service composition modeling and evaluation method used petri net, in Proc. APWebWorkshops, 2006,pp. 905–911.
[25] S. Chemaa, f. Bachtarzi, and A. Chaoui, A high-level petri net based approach for modeling and composition of web services, in Proc. the International Conference on Computational Science (ICCS 2012), Procedia Computer Science, vol. 9, 2012, pp. 469–478, Elsevier.
[26] D. Martin, M. Burstein, J. Hobbs and al, “Owl-s: Semantic markup for web services”, [Online]. Available: http://www.w3.org/Submission/OWL-S/.
[27] R. Akkiraju and B. Sapkota, “Semantic annotations for wsdl and xml schema usage guide”, (2007), [Online]. Available: http://www.w3.org/TR/sawsdl-guide/.
[28] Z. Liu, A. Ranganathan, and A. Riabov, “Modeling web services using semantic graph transformations to aid automatic composition”, in IEEE International Conference on Web Services (ICWS 2007), Salt Lake City, Utah, Jul.13-19, 2007.
[29] B. Srivastava and J. Koehler, “Web service composition current solutions and open problems”, in ICAPS 2003 workshop on Planning for Web Services, Jul.22, 2003.
[30] G. Chiola. “GreatSPN 1.5 software architecture”. In Proc. 5th Int. Conf. Modeling Techniques and Tools for Computer Performance Evaluation, Torino, Italy, February 1991.