Specifying a model of semantic web service composition

Souleymane OUMTANAGA,
Laboratory for Informatics and Telecommunications Research (LARIT),
INPHB, 08 BP 475 Abidjan 08, Cote d'Ivoire
oumtana@nic.ci

Issa TRAORE,
Laboratory for Informatics and Telecommunications Research (LARIT),
INPHB, 08 BP 475 Abidjan 08, Cote d'Ivoire

Michel BABRI
Laboratory for Informatics and Telecommunications Research (LARIT),
INPHB, 08 BP 475 Abidjan 08, Cote d'Ivoire
michelbabri@yahoo.fr

***Abstract*-One of the major issues of semantic web is the discovery and identification of the best service that responds to a user request. In this article we specify a semantic web services composition model built through an ontology described in the logical description language ALN (Attributive Language with unqualified Number restrictions). We then show a method for automatic discovery of semantic web services based on graph theory. In order to make the composition model evolutive, we propose a solution for the insertion of new services, thus avoiding the rebuild of the minimum transversals tree.**

***Keywords:* Semantic web service; Composition; Services; Discovery; Hypergraph; Tree.**

1. INTRODUCTION

The issue of semantic web services discovery is a vibrant research area, as the research projects MKBEEM [1], D2CP [2],TCS-technique [3] and [4] show. The need for automation of the design process and the implementation of semantic web services are the same as those for the Web, namely, how to formally describe the knowledge in order to make it usable by machines [3]. Most of the projects referred to above do not include in their services discovery technique methods for removal or insertion of new web services. In this paper, we propose a semantic web services composition model; then a method for services discovery is presented. We also propose a method of inserting new services. Section 2 of this work presents a brief introduction to semantic web and the main techniques of semantic web services discovery. The third section is devoted to presenting our semantic web services composition model. Section 4 proposes a model for discovering the best services that match a user request. This model takes into account the possibility of integrating new services.

II. SEMANTIC WEB SERVICES DISCOVERY TECHNIQUES

The description of semantic web services handled by machines requires a rigorous formalism as provided by ALN (Attributive Language with unqualified Number restrictions) expressing (N) cardinality restriction [2]; it also requires an abstract model of what is described by the expression of ontologies (RDFS, OWL) based on the XML meta-language. For example:

$$S_l = \text{Vol} \sqcap \forall\text{lieuArr.Abidjan} \sqcap \text{Herberg} \sqcap \geq 2\text{stars} \qquad (1)$$

We have here an abstract service ontology [5] describing a flight to Abidjan and accommodation in an at least 2 star hotel, express ed in the logical description language ALN. Let us consider the services as conjunctions of atomic concepts. We present below in ALN five abstract services and a query Q from a user:

$$S_1 \equiv A_1 \sqcap A_3 \sqcap A_7 \sqcap A_{12},\ S_2 \equiv A_1 \sqcap A_4 \sqcap A_6 \sqcap A_{10},$$
$$S_3 \equiv A_1 \sqcap A_2 \sqcap A_9 \sqcap A_{10},\ S_4 \equiv A_2 \sqcap A_4 \sqcap A_{10} \sqcap A_7,$$
$$S_5 \equiv A_3 \sqcap A_{11} \sqcap A_{12} \sqcap A_8 \text{ and a query}$$
$$Q \equiv A_1 \sqcap A_2 \sqcap A_3 \sqcap A_4 \sqcap A_5$$

Fig. 1 Services and Concepts

To better address the problems of integration, an extended architecture of semantic web services is required. It consists of several layers, thus the name web services stack. The stack consists of several layers, each layer being based on a particular standard. The relation between Web services composition languages and other standards such as SOAP, WSDL, and UDDI. SOAP is a protocol for exchanging information in a decentralized (see Figure 2), distributed environment using typed message exchange and remote invocation [7].

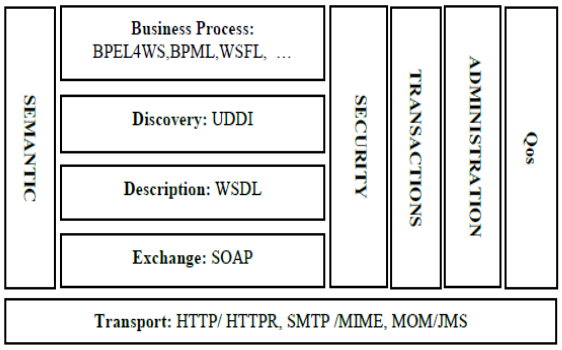


Fig. 2 Stack of semantic web services

Business interactions require long-running interactions that are driven by an explicit process model. This raises the need for Web services composition languages such as BPEL4WS, WSFL, and BPML. These languages are also known as Web services flow languages.

III SEMANTIC WEB SERVICES DISCOVERY

The discovery of semantic web services consists in mapping user queries with available services on the basis of their ALN logic description. In the following, we present the recent semantic web services discovery techniques.

A. ALN logic description and structural subsumption

Definition 1 The semantic distance is composed of $rest_E(Q)$ and $miss_E(Q)$, defined by :
$rest_E(Q) \equiv Q - lcs(Q,E)$ *and* $miss_E(Q) \equiv E - lcs(Q,E)$ where $E \equiv S_1 \sqcap S_2 \sqcap \ldots \sqcap S_n$
$rest_E(Q)$ contains information from the query that can not be satisfied, and $miss_E(Q)$ contains information of E which is not relevant to the user request.

The search for the best services consists in minimizing, $rest_E(Q) \not\equiv Q$ and $misst_E(Q)$ relatively to its syntax size. Thereafter, "misss (Q)" is preferred to "$rest_{Si}(Q)$" because it offers more choice for eventual negotiation.

Definition 2 (reduced clause form and structure equivalence) Let L be a description logic.

- **A clause** in L is a description A with the following property:

  $$(A \equiv B \sqcap A') \quad (B \equiv \top) \quad (B \equiv A).$$

  Every conjunction $A_1 \sqcap \ldots \sqcap A_n$ of clauses can be represented by the clause set $\{A_1, \ldots, A_n\}$.
- $A = \{A_1, \ldots, A_n\}$ is called a **reduced** clause set if either $n = 1$, or no clause subsumes the conjunction of the other clauses: $1 \leq i \leq n : A_i \not\sqsupseteq A \setminus A_i$.

  The set A is then called a **reduced clause form (RCF)** of every description, $B \equiv A_1 \sqcap \ldots \sqcap A_n$.
- Let $A = \{A_1, \ldots, A_n\}$ and $B = \{B_1, \ldots, B_m\}$ be reduced clause sets in a description logic L. A and B are **structure equivalent** (denoted by $A \approx B$) iff: $n = m \quad 1 \leq i \leq n \quad 1 \leq j, k \leq n$:
  $A_i \equiv B_j \quad B_i \equiv A_k$
- If in a description logic for every description all its RCFs are structure equivalent, we say that RCFs are **structurally unique** in that logic. The structural difference operation is defined as being the set difference between clause sets where clauses are compared on the basis of the equivalence relationship. Let us now introduce the notion of structural subsumption as defined in [2].

Definition 3 The subsumption relation in a description logic L is said **structural** iff for any clause A   L and any description $B = B_1 \sqcap \ldots \sqcap B_m$ L which is given by its RCF, the following holds:

$$A \sqsupseteq B \qquad 1 \leq i \leq m : A \sqsupseteq B_i$$

Definition 4 (syntaxical size of a concept) The syntaxic cost of the T-concept ALN is defined by the quantity $|.|$ given by:

(i) $|\perp| = |\top| = 0$

(ii) $|A| = |\neg A| = 1$

(iii) $|\exists R.B| = |\forall R.B| = 2 + |B|$

(iv) $|A \sqcap B| = |A \sqcup B| = |A| + |B|$

The discovery approach presented below is based on scheduling of the the 1-1 arity reasonings [4] .

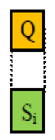
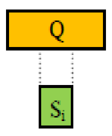
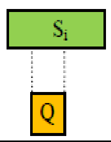
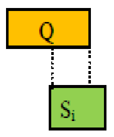
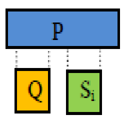
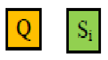
Name	*Equivalence*	*Subsomption*	*Plug-in*	*Intersection*	*Super- Concept*	*Disjunction*
Relation	$Q \equiv S_i$ et $Q \sqcap S_i \not\equiv \perp$	$S_i \sqsubseteq Q$ et $Q \sqcap S_i \not\equiv \perp$	$Q \sqsubseteq S_i$ et $Q \sqcap S_i \not\equiv \perp$	$Q \sqcap S_i \not\equiv \perp$	$\exists P \neq Top$ tel que : $S_i \sqsubseteq P$ et $Q \sqsubseteq P$	$Q \sqcap S_i \equiv \perp$
Diagram						

Fig. 3 1-1 arity reasonings soning

We first look for equivalent services, then for subsumed (or plug-ins) services and no intersection empty services with the user query, and finally services with a super concept that matches the user query. It was a case of disjunction where no service satisfying the preceding relations.

The discovery approach presented below is based on the minimum transversals technique: it consists in defining a hypergraph H(Σ, Γ) from registered web services and the user query (see Figure1).

Definition 5 (Transversal Hypergraph): A hypergraph is a pair H = (Σ, Γ) of a finished ensemble $\Sigma = \{S_1, S_2, \ldots, S_n\}$ and subset family Γ of Σ. The elements of Σ are called tops, the elements Γ are called edges. Unit $T \sqsubseteq \Sigma$ is transversal of H if it crosses all edges of H, i.e : $\forall E \in H,\ T \sqcap E \not\equiv \varnothing$.

A transversal T is minimum if it does not have a transversal subset. The ensemble of minimum transversals of H is noted Tr(H).

B. Construction of the hypergraph H' = (Σ',Γ').

The services $(S_i)_{1 \leq i \leq n}$ in the T-terminology are associated with $(V_{Si})_{1 \leq i \leq n}$,which constitute the tops of the hypergraph [12].

$\hat{H}_{T,Q} = (\hat{\Sigma}, \hat{\Gamma})$ therefore $\hat{\Sigma} = \{V_{Si}\}_{1 \leq i \leq n}$ and $\hat{\Gamma} = \{W_{Ai}\}_{1 \leq i \leq p}$

Each concept of clause $(A_i)_{1 \leq i \leq p} \in Q$ is associated with the edges of $\hat{H}_{T,Q}$ noted W_{Ai} and, $W_{Ai} = \{V_{Si} / S_i \in S_T$ et $A_i \in_{\equiv} lcs_T(Q, S_i)\}$ where $\in_{\equiv}$ is for adhesions modulo equivalence of the clauses and $lcs_T(Q, S_i)$ is given by its RCF (reduced forms of clauses) [8].

After the construction of the hypergraph (see Figure 5) and by using an algorithm, we look for minimum transversals that provide the combination of services that match the user's request.

IV. SEMANTIC WEB SERVICES COMPOSITION MODEL

The service may only provide part of the functionality desired by the user. For example, booking a trip may require both flight and hotel reservation, which are provided by two different services. In this case, the services need to be combined, or *composed*, to achieve all of the goals of the user (see [8], [9]).

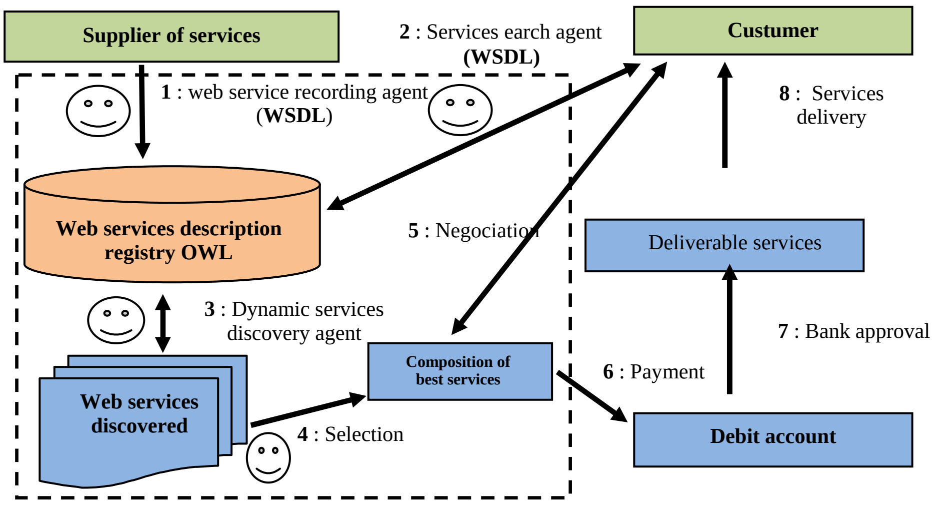

Supplier of services
2 : Services earch agent (WSDL)
Custumer
1 : web service recording agent (WSDL)
8 : Services delivery
Web services description registry OWL
5 : Negociation
Deliverable services
3 : Dynamic services discovery agent
7 : Bank approval
Composition of best services
6 : Payment
Web services discovered
4 : Selection
Debit account

Fig.4 Model operating of service Web

Concretely, the task of Web service composition is: given a number of services that potentially provide part of the desired functionality, combine the services in such a way that they together provide the desired functionality, and specify how they should interact.

This model (see Figure. 4) presents a set of diagrams that describe in a formal and comprehensive way the chosen example.

(1) The service providers register their services in the service description registry WSDL (Web Service Description Language), (2) The costumer (user) submits a request in natural language, which is then translated by an agent into WSDL in order to ease the interactions between agents, (3) and (4) The dynamic services discovery agent looks for the services that are close to the user request and the best services among those discovered. (5) The best services composition is transmitted to the user, and once (s)he agrees, (6) the amount corresponding to the cost of the services is debited from his/her account (7), and the requested services are delivered (8).

V. COMPOSITION MANAGER

During composition, new services need to be discovered and/or selected. This might require additional discovery and/or selection and negotiation activities during the composition process. There are a number of challenges in Web service composition. First of all, the language that is used to represent the composition of services [6, 10], must be able to represent dependencies between the services, and it must be possible to verify that the composition of services indeed provides the desired functionality and that at any point in the process the conditions for invoking the next service (e.g., input data is available) are satisfied.

The service composition manager is made up of a planner engine and an execution engine. When an instance of a composite service is initiated, the planner engine contacts the Web services registry to search for candidate component services, and, based on the candidate services retrieved, it generates an execution plan, i.e., an assignment of component services to the activities in the schema of the composite service [11, 12]. Based on the execution plan, the adaptive execution engine then orchestrates the component services to execute the instance of the composite service.

An ontology describes knowledge that is valid only for a particular modeling context; our architecture provides a solution for combining several modeling contexts in the same.

VI. SEMANTIC WEB SERVICE DISCOVERY

C. The hypergraph and the cost of the services

In this section we illustrate our technique for best services search by taking again the example in Figure 1.

In this example, there are services that have non-empty intersections with Q. This is illustrated in Figure 4, which shows the various intersections between the services offered and those requested by the user. We construct the hypergraph H = (Σ ', Γ').
The algorithm in Figure 4 allows to find the minimum transversal [12] that give the set of services that are closest to the services requested by the user.

$H'_{T,Q} = (\Sigma', \Gamma')$ where
$\Sigma' = \{V_{S1}, V_{S2}, V_{S3}, V_{S4}, V_{S5}\}$ and
$\Gamma' = \{W_{A1}; W_{A2}; W_{A3}; W_{A4}; W_{A5}\}$
It gives, $W_{A1} = \{V_{S1}, V_{S2}, V_{S3}\}$, $W_{A2} = \{V_{S3}, V_{S4}\}$, $W_{A3} = \{V_{S1}, V_{S5}\}$ and $W_{A4} = \{V_{S2}, V_{S4}\}$.
Let E be the conjunction of all the services offered: $E \equiv S_1 \sqcap S_2 \sqcap \ldots \sqcap S_n$ We also have:

Table 1. The cost of the services

	The missing	The cost
$miss_{VS1}(Q)$	$\{A_7, A_{12}\}$	6
$miss_{VS2}(Q)$	$\{A_6, A_{10}\}$	2
$miss_{VS3}(Q)$	$\{A_9, A_{10}\}$	2
$miss_{VS4}(Q)$	$\{A_7, A_{10}\}$	4
$miss_{VS5}(Q)$	$\{A_8, A_{11}, A_{12}\}$	2

We note: $H_{T,Q} = (\Sigma, \Gamma)$, where $\Gamma = \{W_{A1}, W_{A2}, W_{A3}, W_{A4}\}$.

This new writing of the hypergraph presents the edges ensemble of edges Γ, from which W_{A4} is removed; the latter will be subject to negotiation or to a new search. The algorithm below allows in one hand to generate a tree structure, and in another hand to determine the minimum transversals of a hypergraph Tr $(H_{T,Q})$, [12].

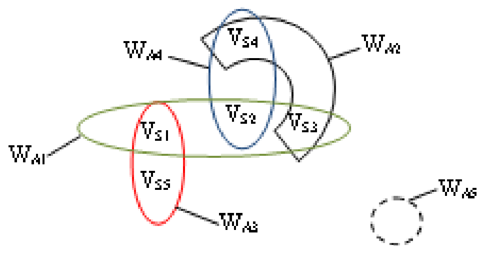


Fig. 5 Hypergraph graphical example

D Algorithm of minimum transversals tree

Let H = (Σ, Γ) be a hypergraph. We note $\Sigma = \{S_i\}_{i \in \mathbb{N}}$ the ensemble of tops, $\Gamma = \{e_j\}_{j \in \mathbb{N}}$ the edges of H, where $i \leq p$, $j \leq n$ are finished indices. During the generation of the tree [11], Algorithm 1 generates the left son and the right brothers constituting the nodes at level j. We have the nodes: $N_{j,1}$, $N_{j,2}$,…,$N_{j,k}$, with k being indices defining the brothers at level j.

Algorithm 1: Generate a tree and determines the minima transversals.

Algorithm (''Transverses tree" function):

***Description**: Generate a tree and determines the transverse minima of hypergraph H)*

***Input :** Hypergraph $H = (\Sigma, \Gamma)$*

***Output:** tree and transverse minima H.*

```
1- $N_{j,T} \leftarrow \emptyset$;   /* initialisation; no node found */
2-        loop on a element $e_j$ ;    /* $e_j$ edges of $\Gamma$.  $j = 1, 2, ..., n$  */
3-              $N_{1,T} \leftarrow e_1$;   /* to break up into singleton $N_{1,i}$, $i=1,2, ... p$ */
4-                 if ($N_{j,i} \cap e_{j+1} \neq \emptyset$)    /* $N_{j,i}$ has a common node with $e_{j+1}$ */
5-                  {
6-                   $N_{j+1,i} \leftarrow N_{j,i}$;     /* $N_{j+1,i}$ is a node of $e_{j+1}$ */
7-                  }
8-                   else
9-                     {
10-                   $N_{j+1,M} \leftarrow$ { $N_{j,i}$   $e_{j+1}$ the cartesian product}
                        /* is carried out one-size larger supersets of $N_{j,i}$ using the vertices of $e_{j+1}$ */
11-                    };
12-                   if ($N_{j+1,M} \neq \emptyset$)
13-                 {
14-                   $N_{j+1,k} \leftarrow$ Clean
15-                  Supernodes ($N_{j+1,M}$, $N_{j,i}$ );
                 /* removing non-minimal nodes from $N_{j+1,M}$*/
16-                }
17-             End If
18-            $N_{j+1,T} \leftarrow N_{j+1,i} \cup N_{j+1,k}$   ;
16-           }
17-         End If
18-       }
19-     turn over  $N_{n,T}$ ;   /* all nodes of the   last  level */
20- End
```

Fig. 6 Algorithm of the minimums transversals tree.

Temporal and spatial complexity: The problem of minimum transversals search is generally of NP-difficulty. for the spatial complexity, it should be noted that the tree (nodes and branches) will be stored.

Example: In this example, $H_{T,Q} = (\Sigma,\Gamma)$ where $\Gamma = \{ W_{A1} , W_{A2} , W_{A3} ,W_{A4} \}$,
with $e_j = W_{Aj}$, For i = 1, 2,…, 4, we obtain after incrementation the following result:
Let, $W_{A1} = \{V_{S1}, V_{S2}, V_{S3}\}$, $W_{A2} = \{V_{S3} , V_{S4}\}$, $W_{A3} = \{V_{S1} ,V_{S5}\}$ and $W_{A4} = \{V_{S2},V_{S4}\}$.

Step 1: First, the algorithm1 takes Γ_{A1} and computes its minimal transversals that are,
$e_1 =\{V_{S1},V_{S2},V_{S3}\}$.
Where, $\{V_{S1}\} = N_{1,1}$, $\{V_{S2}\} = N_{1,1}$, $\{V_{S3}\} = N_{1,1}$.
Then the minimal transversal is: $Tr (H_{TQ}) = N_{1,T} = \{N_{1,1}, N_{1,2}, N_{1,3}\}$

Fig.7 Tree graphical example for k = 1

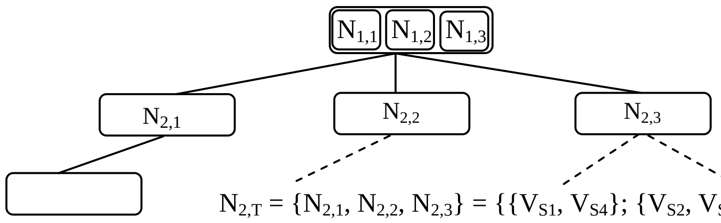
Step 2: The algorithm continues with processing W_{A2}. Each time when a new element of H is handled, the already found minimal transversals are tested.

Let $W_{A2} = \{V_{S3}, V_{S4}\}$; $N_{13} \cap W_{A2} = \{V_{S3}\} \neq \emptyset$. Thus $N_{13} \subset N_{2T}$, However N_{11} and N_{12} has no common part with W_{A2}, which means that N_{11} and N_{12} are not a minimal transversal of W_{A2}. Thus the following candidates are generated:

N_{11} $W_{A2} = \{V_{S1}\}$ $\{V_{S3},V_{S4}\} = \{\{V_{S1},V_{S3}\}, \{V_{S1}, V_{S4}\}\}$
N_{12} $W_{A2} = \{\{V_{S2}, V_{S3}\},\{V_{S2}, V_{S4}\}\}$.

$N_{2,M} =\{\{V_{S1},V_{S3}\}, \{V_{S1}, V_{S4}\},\{V_{S2}, V_{S3}\},\{V_{S2}, V_{S4}\}\} \neq \emptyset$

The algorithm removes item sets $\{V_{S3},V_{S1}\}$ and $\{V_{S1},V_{S3}\}$, that have subsets in $\{V_{S3}\}$ since they are not minimal transversals.
Thus we have: $N_{2,1} = \{V_{S1},V_{S4}\}$, $N_{2,2} = \{V_{S2},V_{S4}\}$ and $N_{2,3} =\{V_{S3}\}$



$N_{2,T} = \{N_{2,1}, N_{2,2}, N_{2,3}\} = \{\{V_{S1}, V_{S4}\}; \{V_{S2}, V_{S4}\}, \{V_{S3}\}\}$;

Fig. 8 Tree graphical example for k = 2

The same steps are repeated with the other elements W_{A3}, W_{A4} of $H = (\Sigma, \Gamma)$. When the algorithm terminates, all minimal transversals of the hypergraph H are discovered.

$N_{4,T} = \{\{V_{S1}, V_{S4}\}; \{V_{S2}, V_{S4}, V_{S5}\}; \{V_{S1}, V_{S2}, V_{S3}\}; \{V_{S2}, V_{S3}, V_{S5}\}; \{V_{S3}, V_{S4}, V_{S5}\}\}$.

We can observe that: $N_{4,T} = Tr(H_{T,Q})$, i.e the set of leaves of the generated tree, is equal to the aggregation of the minimum transversals hypergraph $N_{T,Q}$. If only one set of services is discovered, it is chosen by the discovery agent as the best service; if several sets of services are discovered, the best service will be the one with the lowest cost.

Definition 6 That is to say the kth the top brother of level j of the tree. We note, the cost of the top such as:

$$N^*_{j,k} = \left| \sum_{VS_i \in N_{j,k}} miss_{VS_i}(Q) \right|$$

VII. BEST SEMANTIC WEB SERVICES DISCOVERY ALGORITHM

The algorithm below named A* strategy is a version of the best-first algorithm. It allows to find the minimum transversal with the lowest cost, i.e. the compute best covers of a given request. We have:

- $N_{1,T}$ is the initial node (see step1 of algorithm1).
- $c(N_{j+1,k}, N_{i,j})$ is the weight over the edge that connect $N_{i,j}$ and $N_{j+1,k}$.
- $f(N_{j+1,k})$ is the cumulative cost to reach $N_{j+1,k}$.
- $h(N_{i,j})$ is the heuristic value for the node $N_{i,j}$ to reach the node $N_{j+1,k}$.

The A* algorithm works as shown in the flow of Algorithm 2.

Algorithm 2: Determination of lowest cost minimum transversal.

```
Algorithm (''minimum cost'')
*Description:* Find the node with optimal cost
*Input:* Tree generated by Algorithme 1
*Output :* minimum node with the lowest cost
1- Open ← $N_{1,T}$ ;                     /*initialization of Open */
2-  Closed ← Ø;                           /*initialization of Closed*/
3-  $N^*_{1,T} \leftarrow 0$ ;            /*initialization of cost */
4-   $f(N_{1,T}) \leftarrow h(N_{1,T})$;
5-    While Open ≠ Ø  Do
6-     To extract from Open the node $N_{i,j}$ such as $f(N_{i,j})$ is minimal
7-       To insert $N_{i,j}$ in Closed;
8-         if $N_{i,j} \in N_{n,T}$          /* $N_{i,j}$ is  a sheet of the tree */
9-              Then End
10-              else
11-                For $N_{j+1,k} \in succ(N_{i,j})$        /* $N_{j+1,k}$ successor of $N_{i,j}$ */
12-               if $N_{j+1,k} \notin Closed \cup Open$  or $N^*_{j+1,k} > N^*_{i,j} + c(N_{j+1,k}, N_{i,j})$ ;
13-                    Then
14-                     $N^*_{j+1,k} \leftarrow N^*_{i,j} + c(N_{j+1,k}, N_{i,j})$ ;
                    /* the sum of the costs of the services of node $N_{j+1,k}$ */
15-                   $f(N_{j+1,k}) \leftarrow N^*_{j+1,k} + h(N_{j+1,k})$ ;
16-                   father ($N_{j+1,k}$) ← $N_{i,j}$ ;
17-               To insert according to f, $N_{j+1,k}$ in Open;
18-             End If
19-           End For
20-        End If
21-  End While
22- End
```

Fig. 9. Algorithm of best semantic web service.

Temporal and spatial complexity: The time complexity as well as the space complexity is exponential. The optimality of the A* algorithm is guaranteed if we use a heuristic approach.

Example: The tree generated by Algorithm 1 according to the strategy A* allows to determine the leave with the lowest cost, which will be selected as the minimum transversal with the lowest cost.

Let us consider the tree above with the weighted cost of each node (see Figure 10). We use Algorithm 2 to find the best service.

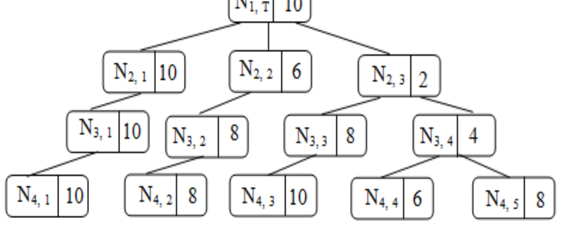


Fig. 10 Tree generated with our model

It should be noted that in our example, the distance between nodes is not taken into account.

Step 1:

We have: $N^{*}_{1,T} = \left|\mathrm{miss}_{VS_1}(Q)\right| + \left|\mathrm{miss}_{VS_2}(Q)\right| + \left|\mathrm{miss}_{VS_3}(Q)\right| = 6+2+2=10$ (see Table 1 and definition 6).
Closed : ($N_{1,T}$;10).
Open : ($N_{2,1}$;10) , ($N_{2,2}$;6) and ($N_{2,3}$;2).

Step 2 :
f($N_{2,1}$) =10 et f($N_{2,2}$) = 6 f($N_{2,3}$) = 2
Closed : ($N_{1,T}$;10), ($N_{2,3}$;2)
Open: ($N_{2,1}$;10), ($N_{2,2}$;6), ($N_{3,3}$; 8) and ($N_{3,4}$; 6)

The algorithm provides the closed nodes $N_{1,T}$, $N_{2,3}$, $N_{3,4}$ et $N_{4,4}$.
The node, $N_{4,4} = \{V_{S2}, V_{S3}, V_{S5}\}$.
We call such a combination of web services a *best profile cover* of Q using *T*-Concept. We have shown that the best covering problem can be mapped to the problem of computing the minimal transversals with minimum cost of a "weighted" hypergraph.

VIII. ADDING A NEW SEMANTIC WEB SERVICE

By storing the tree in Figure 7, whose leaves are minimum transversals of the hypergraph [12,10], our model allows to add services later. Let us assume that a new service $W_{A5} = \forall$Equip.Piscine = $\{V_{S6}\}$ contains the concept A_5 requested by the user. Let $\Gamma_{A5} = \{V_{S6}\}$ be a new edge of the hypergraph and let,
$W_{A5} \sqcap N_{4,k} = \varnothing$ i.e. this edge meets no other edge of $H_{T,Q} = (\Sigma,\Gamma)$, with $\Gamma = \{ W_{A1}, W_{A2}, W_{A3}, W_{A4}, W_{A5}\}$. We can add $\{V_{S6}\}$ as a leave of the tree through a Cartesian product. The cost $\{V_{S6}\}$ is that of S_6, as it is constant (Figure 11), the service will not change the previous choice of the best service.

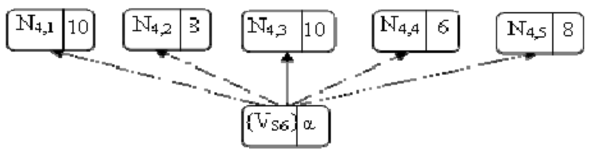


Fig. 11 Adding a new service

The best service is the node: $N_{4,4} = \{V_{S2}, V_{S3}, V_{S5}, V_{S6}\}$ with the cost 6 + α. Let us assume now that the new service requested by the user contains concepts other than A_5. In that case, Γ_{A5} contains one or more services already offered, i.e.:

$$k \quad \{1, 2, 3, 4\} \text{ such that } W_{A5} \sqcap N_{4,k} \neq \varnothing . \text{ If } W_{A5} \sqcap N_{4,k} \neq \varnothing ,$$

then this kth brother node is renewed at 5. Thus, it is possible to add services without regenerating the weighted tree of Figure 7, obtaining a new leave $N_{j+1,k}$ at level j +1.

IX. CONCLUSION

One of the challenging problems that Web service technology faces is the ability to effectively discover services based on their capabilities.

The work done in this paper enabled us to propose a model for the composition of semantic web services based on the descriptive logic ALN. The aim of this paper is to propose a method for a dynamic discovery of semantic web services using graph theory. The particularity of this model is that it has the advantage of storing a tree with weighted nodes; these data will be used to create new services without recomputing the minimum transversal algorithm (Algorithme 1). Our future work will focus on optimizing Algorithm 2 in order to use less memory and to facilitate the underlying business processes. Indeed, the problems of discovery and composition are correlated; the real time resolution of the composition problem of semantic web services[10], requires currently the use of a reduced set of services.

ACKNOWLEDGEMENTS

This research has been supported by a grant from LARIT (Laboratoire en Recherche Information et Télécommunication) and by the Director Fund of the CFTIC (Centre de Formation en Télécommunication Informatique et Communication).

A preliminary version is published in the Proceedings of 10th African Conference on Research in Computer Science and Applied Mathematics (CARI'10),Yamoussokro, Côte d'Ivoire, 2010.

REFERENCES

[1] MKBEEM (2000-2002). The Mkbeem project.URL http://www.mkbeem.com/"W3C Semantic Web" http://www.w3.org/2002/ws/arch/

[2] Rey C., Benatallah B., Leger A., Toumani F.,2007. Covering concept using terminologies in the ALN language, technical report, LIMOS, Clermont-Ferrand, http:www.isima.fr/rey/alnBcov.pdf.

[3] Stollberg, Michael; Hepp, Martin; Hoffmann, Joerg, 2007. A Caching Mechanism for Semantic Web Service Discovery, Proceedings of the 6th International Semantic Web Conference (ISWC 2007), November 11-15, Busan, Korea, Springer LNCS, Vol. 4825, pp. 480-493, 2007.

[4] François Goasdoué, Alain Léger, 2009. Hermes Science. Web sémantique ; '' De la technologie aux services web''. Technique et Science Informatique Vol. 28. ISBN: 9782 7462 2317 2.

[5] Franz.B., Diego C., Deborah L. Mcguinness, Daniele N., Peter F., Patel-schneider, 2003. The description logic handbook. Theory, implementation, and applications. Cambridge University Press. ISBN :13 978 0 511 06694 8.

[6] Jos de Bruijn, Dieter Fensel, Mick Ker-rigan Uwe Keller, Holger Lausen, James Scicluna, 2008. Modeling Semantic Web Services. Springer-Verlag. Berlin Heidelberg.ISBN: 978 3 540 68169 4.

[7] Alonso G., Casati F., Kuno H., Machiraju V. 2004. "Web Services" (chapter 5). In Web Services – Concepts, Architectures and Applications, Springer Verlag. ISBN: 3 540 44008 9.

[8] G. Chafle, K. Dasgupta, A. Kumar, S. Mittal, and B. Srivastava, 2006. "Adaptation in web service composition and execution," in Proceedings of IEEE International Conference on Web Services, pp. 549-557.

[9] Cardoso Jorge, Hepp Martin, Lytras, Miltiadis, 2007. The Future of the Semantic Web for Enterprises, in: Cardoso, Hepp, Lytras (Eds.): Real-world Applications of Semantic Web Technology and Ontologies, ISBN:978 0 387 48530 0,Springer,pp. 3-15.

[10] Passant A., Bojars U., Breslin, J. G., Hastrup T., Stankovic M., Laublet P., et al., 2010. An Overview of SMOB 2: Open, Semantic and Distributed Microblogging. In 4th International Conference on Weblogs and Social Media, ICWSM, pp. 303-306.

[11] Filipowska Agata, Kaczmarek Monika, Hepp Martin, Markovic Ivan, April 27-29, 2009. Organisational Ontology Framework for Semantic Business Process Management, in: Abramowicz, Witold (Ed.): Proceedings of the 12th International Business Information Systems Conference, BIS 2009, Poznan, Poland, Springer LNBIP, Vol.21,pp. 1-12, ISBN: 978 3 642 01189 4.

[12] Claude Berge,1989.Hypergraphs: Combinatorics of Finite Sets. North Holland, ams terdam, ISBN: 0 444 87489 5.