Fully Automated Web Services Discovery and Composition through Concept Covering and Concept Abduction

Azzurra Ragone†, Tommaso Di Noia†, Eugenio Di Sciascio†,
Francesco M. Donini‡, Simona Colucci†, Francesco Colasuonno†
SisInfLab–Politecnico di Bari †
Via Re David, 200, I-70125, Bari, Italy
{a.ragone,t.dinoia,disciascio,s.colucci,f.colasuonno}@poliba.it
Università della Tuscia ‡
via San Carlo, 32, I-01100, Viterbo, Italy
donini@unitus.it

ABSTRACT:

We propose a framework and polynomial algorithms for semantic-based automated Web service composition, fully compliant with Semantic Web technologies. The approach exploits the recently proposed Concept Abduction inference service in Description Logics to extend Concept covering definition to expressive logics and to solve Concept Covering problems in a significant subset of OWL-DL. We show how the proposed approach also deals with not-exact solutions, computing an approximate composition and providing an explanation of which part of the request is not covered by the composite service. We present the formalization of the approach, the proposed algorithms, a prototype system implementing the approach, and illustrate experiments carried out with it.

KEY WORDS:
Knowledge-representation, OWL-S, Description Logics, Semantic-based discovery, Concept Abduction, Concept Covering.

INTRODUCTION

Web service composition amounts to the orchestration of a certain number of existing Web services (or Web service providers) to provide a composite service able to satisfy the user's requirements, in case a single Web service is not adequate. Various approaches have been recently proposed able to provide the orchestration of a Web services set, based on different aspects of both Web service (Sirin et al., 2003; Fensel and Bussler, 2002; Motta et al., 2003) and composition flow (Laukkanen and Helin, 2003; Mecella et al., 2002; Narayanan and McIlraith, 2002; Korhonen et al., 2003) modeling. But, due to the lack of automatic procedures able to cope with non-exact matches, in many of such approaches there is the need for a human intervention in order to establish candidate execution flows. Fully-automation of Web services discovery and orchestration requires as a first step providing service descriptions –what a service offers– that have to be well defined, machine understandable and processable.
The Semantic Web initiative aims to provide web resources with a meaning, with the aid of formal languages and ontologies to allow reasoning on service description. Such type of web-services are now named semantic Web services: "a Semantic Web service (SWS) is a Web service whose description is in a language that has well-defined semantics. Therefore, it is unambiguously computer interpretable,and facilitates maximal automation and dynamism in Web

service discovery, selection, composition, negotiation ...” (Sycara et al., 2003); therefore the Semantic Web provides –or should provide– the infrastructure for the semantic interoperability of Web services (Cabral et al., 2004).
The approach we propose here models the discovery, retrieval and composition with respect to a resource retrieval scenario, that is, a scenario where there is a request and several available resources, which can satisfy the request in a one to one match (one resource retrieved for the request) or in a one to many match (several retrieved resources whose combination satisfies the request). Our target model is hence a user oriented and friendly framework where a typical request is like ”*I’d like to book a hotel provided with a swimming pool and a fitness center* ” rather than only ”Effects = **`HOTEL RESERVATION`**” and a typical service description is like ”*We book for you hotels near the sea provided with all the facilities: swimming pool, fitness center, children area and restaurants*” rather than only ”Preconditions = **`VALID CREDIT CARD`**; Effects = **`HOTEL RESERVATION`**”. What we would like to point out is that limiting the information provided by services to only preconditions (and / or inputs) and effects (and / or outputs) makes hard to detail the service provided. We believe that a service should provide two kinds of information: (1) description of the service behavior, even not a functional one, (2) execution information.

1. *Descriptive Information*. Description of the service behavior, even not a functional one. A simple free text description is not enough if you want to describe in a machine understandable way, that is in the Semantic Web vision, such behavior and its details.
2. *Execution Information*. *Effects*. How the Web service changes the state of the domain world – (*Outputs*. What it changes in the domain world). *Preconditions*. Which state of the world is needed to make the Web service producing such change – (*Inputs*. What it needs to change the state of the world) .

Composed Web services can be atomic ones (having a single step of execution) or composite services themselves. The service composition process can hence create services with an increasing level of complexity, built out of services at a lower abstraction level. Obviously, service composition is not based on the physical integration of its components (Alonso et al., 2003): each service has, in fact, an interface through which it can be invoked, and the base components of the service remain so separated from the composite service. The main issues in service composition are hence modeling –how to define the process model– and execution –the middleware that executes the process model.
In this paper we propose a framework and algorithms, fully compliant with the Semantic Web vision and its technologies, for the automated discovery and composition in a semantic web-services orchestration scenario.
Main contributions of the paper are:

- a logic based approach to service discovery and composition exploiting non standard inference services, i.e., Concept Covering (Hacid et al., 2002; Colucci et al., 2005a; Colucci et al., 2005b) and Concept Abduction (Di Noia et al., 2003). The approach is based on an extension of the Concept Covering definition, originally proposed by Hacid et al. (2002) for Description Logics endowed of limited expressivity, to a generalized one, which exploits the recently proposed Concept Abduction (Di Noia et al., 2003) inference service in Description Logic.
- efficient greedy algorithm, based on the generalized definition, to solve a Concept Covering for semantic service discovery.
- an extension of the concept covering algorithm, to compose a flow of discovered services, i.e., an orchestration. The algorithm takes into account both effects duplication and approximate solutions. Both algorithms we present are polynomial in time for the significant subset of OWL-

DL we adopt. Our approach, by means of concept abduction, also hypothesizes, when an exact solution is not found, what is still missing to complete the cover.

- complete prototype system developed in the MaMaS-tng framework, based on an enhanced extension of OWL-S 1.1 model, and an extended version of JUDDI, namely JUDDI+, and experiments carried out with it.

The remaining of the paper is structured as follows: next Section summarizes relevant related work and briefly recalls basics of description logics; then we present our extension of Concept Covering definition and a greedy algorithm, which exploits Concept Abduction, to determine a Concept Cover. After that we show how such algorithm can be at the basis of a semantic Web services composition. Subsequently, the approach and behavior of the related algorithms are thoroughly explained with the aid of an example. Finally a section is devoted to the description of our prototype implementing the approach, its integration in the OWL-S 1.1 framework, and to experimental results obtained with it. Last section draws the conclusions and outlines future research directions.

RELATED WORK

In this section we briefly report on relevant related work, and on basics of Description Logics, the formal framework we adopt throughout the paper.
We start noting that the automation of service discovery and composition processes calls for clearly establishing what a service actually is, how we can define its description and at which level of abstraction. As defined in W3C Web Services Architecture (WSA) (http://www.w3.org/tr/ws-arch/) a service is "an abstract resource that represents a capability of performing tasks" and it has a service description describing its semantics. Yet there the semantics of a service is defined as "the behavior expected when interacting with the service. The "semantics expresses a contract (not necessarily a legal contract) between the provider entity and the requester entity. It expresses the intended real-world effect of invoking the service. A service semantics may be formally described in a machine readable form, identified but not formally defined, or informally defined via an out of band agreement between the provider entity and the requester entity." Preist (2004) correctly argues "how can something which is defined as an agreement between the two entities be advertised prior to the two entities communicating?" He then suggets to make a distinction between three categories of services: *abstract service, agreed service* and *concrete service*. An *abstract service* can provide many different possible concrete service, so a *concrete service* is a kind of realization of an abstract service. An *agreed service* is an abstract service agreed between two parties, which completely defines the service requester and the service provider, with their associated interfaces. An abstract service can be used in the discovery phase, because it shows "what capabilities" the service provides, while not already requiring a definition of what a requester and provider have to agree upon.
In this paper we refer, in accordance with Preist's categorization, to an abstract service description, that is a machine-processable description specification of a set of provided services.

Background work

In recent years commercial systems have emerged based on service-oriented architectures, including IBM webshpere and BEA WebLogic, which anyway currently adopt discovery and composition approaches that usually rely on UDDI and tModels for pure keyword-based search. Inspired by the Semantic Web initiative, various frameworks have been proposed to semantically enrich service descriptions, thus supporting sophisticated discovery and orchestration techniques. Striving to become "de facto" standards are OWL-S (formerly DAML-S)

(http://www.w3.org/Submission/OWL-S/.) WSMO (H. Lausen and D. Roman and U. Keller., 2004), IRS-III, and METEOR-S (Patil et al., 2004). The main objective of OWL-S is to provide an OWL-based Web service ontology in order to give an unambiguous and machine-interpretable description of a Web service, allowing software agents to discover, invoke, compose and monitor web resources offering specific services. WSMO also is a formal ontology for describing several aspects related to Semantic Web services. WSMO is based on WSMF (Web Service Modeling Framework), proposed by Fensel and Bussler (2002) with the aim to provide a conceptual model for developing and describing services. It consists of four elements: *ontologies*, represent the terminology, which give meaning to the other elements; *goals*, the desires of the service requester, provide the means to express a high level description of a concrete task; *Web services*, define capability description, interface, used mediators and non functional properties; and *mediators*, that deal with the interoperability problems. Differently from the previous two frameworks, METEOR-S does not introduce a dedicated upper ontology, and utilizes existing Web service technologies enriched with RDF-S to. map WSDL messages and operations to concepts in domain ontologies. Another framework developed to describe and execute semantic Web services is IRS-II (Motta et al., 2003) (Internet Reasoning Service). IRS-II is based on the UPML (Unified Problem Solving Method Development Language) framework. UPML has four categories of components specified by means of an appropriate ontology: domain models, the domain of an application; task models, which specify the input, output, goal to be solved, pre and post-conditions; Problem Solving Methods (PSMs), relate to reasoning processes applied to solve specific tasks; bridges, which provide mappings between the components. The IRS-III (J. Domingue and Cabral L. and Hakimpour H. and Sell D. and Motta E., 2004) platform, built upon IRS-II, allows to create WSMO-based Semantic Web services, extending UPML based types of knowledge models with the types exposed in WSMO, in order to incorporate and extend the WSMO ontology. Semantic-based discovery and composition is currently a widely researched area.

Much work has been devoted to the basic discovery process, and we briefly recall some relevant work here, without any claim of completeness, before moving on to the composition stage. In (Paolucci et al., 2002) the location of Web services is based on the capabilities they provide. The matching is based only on the inputs and outputs of a Web service without taking into account the overall description of the service, which lacks in this approach, the discovery is carried out mainly exploiting features of the LARKS approach. Description Logics based approaches include those in (Di Noia et al., 2004; Gonzales-Castillo et al., 2001; L. Li and I. Horrocks, 2004; S. Agarwal and S. Lamparter, 2005; Benatallah et al., 2003; Klusch et al., 2005; U. Keller and R. Lara and H. Lausen and A. Polleres and D. Fensel, 2005), in which the discovery phase amounts to a semantic matchmaking process between requests and services. The need to overcome exact – though logic-based– matches, led researchers to study different possibilities. In (L. Li and I. Horrocks, 2004; Paolucci et al., 2002) matches are divided into categories, yet once they have been categorized there is still the need to determine most promising ones. To this aim in (Di Noia et al., 2004) matches are not only categorized, but also ranked according to a semantic-based score. In (S. Agarwal and S. Lamparter, 2005) the idea is to use a fuzzy description logic to obtain a ranking of matches, while in the recent (Klusch et al., 2005) it is proposed to adopt a mixed approach, using both logic-based classification and rankings determined using classical unstructured text retrieval measures. This approach derives from the LARKS matchmaker, and as the authors admit, prevents fully automated discovery and composition. In (U. Keller and R. Lara and H. Lausen and A. Polleres and D. Fensel, 2005) a model to automate service discovery was proposed. In this approach static and dynamic aspects of service descriptions are considered, providers of services are dynamic selected based on the semantic descriptions of their capability.

Semantic-based composition approaches have also been proposed by several authors. Various proposals derive from the wide experience of the AI planning community, although some issues,

related to the typical closed world assumption of planning approaches, have still to be clearly clarified . For a recent survey on such approaches please refer to (Rao and Su., 2004) and (J.Peer, 2005). In (Sirin et al., 2003) a prototype was developed guiding the user in the dynamic composition of Web services. The service filtering is based first on the profile descriptions, with reference to the ServiceProfile of OWL-S, then on the constraints that the user specifies at each step of the workflow. The main limits of this approach are primarily two: a human intervention is needed at each step for composing the workflow and the matching between two services is performed taking into account only the output of the first service that could be fed to the second service as input. A semi automated system has also been proposed by Liang et al. (2004), using WSFL and adopting a constraint-based approach. A rule-based system, which may also require user intervention in the selection is presented in (Medjahed et al., 2003). Also rule-based, but with discovery carried out using the matchmaker proposed by Paolucci et al. (2002), is the OWLS-UDDI system presented in (Sycara et al., 2003). In (Gomez-Perez et al., 2004) a PSM-based approach is proposed for services modeling, where the functional features of a service are described as tasks, while the internal structure of the services are modeled as the methods that solve those tasks. Here only functional information is taken into account for composition. No descriptive information is provided neither used. The same is in the following related works. Narayanan and McIlraith (2002) focus on OWL-S ontology to describe the capabilities of Web services. The semantic of OWL-S specification is translated to a first-order logic language to allow reasoning on the features of the same services. Encoding service description in a Petri Net formalism provides decision procedures for Web service simulation, verification and composition. Petri Nets were proposed both for modeling and orchestration of services in (Mecella et al., 2002; Mecella and Pernici, 2002) , too.

Berardi et al., (2003) proposed an approach based on finite state machine for automatic e-service composition. In (Berardi et al., 2004) also incomplete specification of the client action sequences was taken into account. In (S. Grimm and B. Motik and C. Preist, 2004) several inferences are discussed based on different ways of handling variance during the matching process. This variance is, according to the authors, due to a gap between the formal semantics of service descriptions and the modeler's intuition. Hence the quality of the discovery process depends on the control of service variance in service discovery. In (Haller et al., 2005) the principles of the Web Service Modeling Ontology (WSMO) (H. Lausen and D. Roman and U. Keller., 2004) are applied to SSOA (Semantic Service Oriented Architecture), extending the notion of Service-Oriented Architectures by Semantic Web services, with the aim to enable dynamic discovery and invocation in enterprise application integration (EAI). In (Benatallah et al., 2004) a fully description logic based service discovery approach is presented, which adopts a non-standard inference service, namely concept covering, to discover one or more service able to fulfill a given request. Unfortunately the concept covering, as defined there, needs the Difference operator, which makes sense only for a limited set of logics –we discuss this thoroughly in later sections– and, furthermore, the orchestration process is not dealt with.

DESCRIPTION LOGICS BASICS

Description Logics (DLs) (Baader et al., 2002) are a family of logic formalisms for knowledge representation. There are close relationships between DLs and OWL, the Web Ontology Language (www.w3.org/TR/owl-features/.); in this paper we use DLs as formal framework, and we briefly recall here basics of the formalism.

Basic syntax elements are *concept* names, e.g., `Hotel`, `Airport`, `Country`, and *role* names, such as `allowedCountries`, `hasBeds`. Intuitively, concepts stand for sets of objects, and roles link objects in different concepts. Formally, concepts are interpreted as subsets

of a domain of interpretation Δ, and roles as binary relations (subsets of Δ × Δ). Basic elements can be combined using constructors to form concept and role expressions, and each DL has its distinguished set of constructors. Every DL allows one to form a conjunction of concepts, usually denoted as ⊓; some DL include also disjunction ⊔ and complement ¬ to close concept expressions under boolean operations. Roles can be combined with concepts using existential role quantification, e.g., `Bedroom⊓∃hasBeds.DoubleBed`, which describes the set of bedrooms with double bed, and universal role quantification, e.g., `Flight⊓∀allowedCountries.NorthAmerica`, which describes flights only for countries in North America. Other constructs may involve counting, as number restrictions: `Bedroom⊓(≤1hasBeds)` expresses bedrooms with only one bed, and `Bedroom⊓(≥4hasBeds)` describes bedrooms endowed with at least four beds. Many other constructs can be defined, increasing the expressive power of the DL, up to n-ary relations (Calvanese et al., 1998).

DL syntax	*name*	*description*
⊤	universal concept	All the objects in the domain.
⊥	bottom concept	The empty set.
A	atomic concepts	All the objects belonging to the set represented by A.
¬A	atomic negation	All the objects not belonging to the set represented by A.
C ⊓ D	intersection	The objects belonging both to C and D.
∀R.C	universal restriction	All the objects participating in the R relation whose range are all the objects belonging to C
∃R	unqualified existential restriction	There exists at least one object participating in the relation R.
(≥n R) (≤n R) (= n R)	unqualified number restriction	Respectively the minimum, the maximum and the exact number of objects participating in the relation R.

Table 1: Attributive Language with unqualified Number restrictions $\mathcal{ALN}$ DL basic constructs.

An ontology can be modeled in a DL using a set of inclusion assertions, and definitions, which impose restrictions on possible interpretations according to the knowledge elicited for a given domain or task. For example, we could impose that a generic location can only be either in mountain or sea using the two inclusions: `Location⊑Sea⊔Mountain` and `Sea⊑ ¬Mountain`. Definitions are useful to give a meaningful name to particular combinations, as in `DoubleRoom ≡ Bedroom ⊓ (= 2 hasBeds)`. Historically, sets of such inclusions are called Tbox (Terminological Box) $\mathcal{T}$. The basic reasoning problems for concepts in a DL, taking into account a Tbox $\mathcal{T}$, are satisfiability, which accounts for the internal coherency of the description of a concept (no contradictory properties are present), and subsumption, which accounts for the more general/more specific relation among concepts, that forms the basis of a taxonomy. Different DLs can be identified depending on the allowed constructors and consequently their expressiveness.

Obviously, as expressiveness grows, so does computational complexity of inference services, and usually some trade-off is needed. We use a *simple − TBox* in order to express the relations among

objects in the domain. A *simple − TBox* allows only axioms (for both inclusion and definition) where the left side is represented by a concept name. We point out that our theoretical framework is generic, but we refer, for services actually implemented, to $\mathcal{ALN}$ (Attributive Language with unqualified Number restrictions), which is a subset of OWL-DL. Constructs allowed in $\mathcal{ALN}$ are reported, together with their description, in Table 1. OWL-DL Tags corresponding to $\mathcal{ALN}$ are reported in Table 2[1]. In the rest of the paper we will use DL syntax instead of OWL-DL syntax, to make expressions much more compact.

OWL syntax	*DL syntax*
<owl:Thing/>	$\top$
<owl:Nothing/>	$\bot$
<owl:Class rdf:ID="C"/>	C
<owl:ObjectProperty rdf:ID="R"/>	R
<rdfs:subClassOf/>	$\sqsubseteq$
<owl:equivalentClass/>	$\equiv$
<owl:disjointWith/>	$\neg$
<owl:intersectionOf/>	$\sqcap$
<owl:allValuesFrom/>	$\forall$
<owl:someValuesFrom/>	$\exists$
<owl:maxCardinality/>	$\leq$
<owl:minCardinality/>	$\geq$
<owl:cardinality/>	$=$

Table 2: Correspondance existing between OWL-DL markup elements and $\mathcal{ALN}$ DL syntax

CONCEPT COVERING VIA CONCEPT ABDUCTION

Standard inference services in DLs –subsumption and satisfiability– can be used in several frameworks, including Semantic Web services discovery. For example, given a Web service S and a service request $\mathcal{D}$ modeled as concept expressions in a DL $\mathcal{L}$ w.r.t. an ontology $\mathcal{T}$, Concept Satisfiability can determine whether the request is compatible with the service,i.e., S models information which is not in conflict with the one modeled by $\mathcal{D}$. This task can be performed checking the satisfiability of the concept $\mathcal{T} \models S \sqcap \mathcal{D} \not\sqsubseteq \bot$.

On the other hand subsumption can be used to verify if a Web service described by S can satisfy a service request D. In fact if the relation $\mathcal{T} \models S \sqsubseteq \mathcal{D}$ holds, then S is more specific than $\mathcal{D}$ and it contains at least all the requested features.

Satisfiability and subsumption, return a Boolean yes/no answer, so one is left with the need for results explanation. The Concept Abduction Problem (CAP) was introduced and defined as a non standard inference problem for DLs in (Di Noia et al., 2003), to provide an explanation

[1] Since in $\mathcal{ALN}$ only *unqualified existential restriction* is allowed, then the OWL `<owl:someValuesFrom/>` restriction on an *ObjectProperty* must be `<owl:Thing/>`.

hypothesis when subsumption does not hold, thus someway extending the subsumption relation. We recall its definition for the sake of completeness:

Definition 1 *Let $\mathcal{S}$, $\mathcal{D}$, be two concepts and $\mathcal{T}$ a set of axioms in a Description Logic $\mathcal{L}$, where both $\mathcal{S}$ and $\mathcal{D}$ are satisfiable in $\mathcal{T}$. A Concept Abduction Problem (CAP), denoted as <$\mathcal{L}$, S, $\mathcal{D}$, $\mathcal{T}$>, is finding a concept $\mathcal{H}$ such that $\mathcal{T} \not\models \mathcal{S} \sqcap \mathcal{H} \sqsubseteq \bot$, and $\mathcal{T} \models S \sqcap \mathcal{H} \sqsubseteq \mathcal{D}$.*

Given a CAP, if $\mathcal{H}$ is a conjunction of concepts and no sub-conjunction of concepts in $\mathcal{H}$ is a solution to the CAP, then $\mathcal{H}$ is an irreducible solution. In (Di Noia et al., 2003) also minimality criteria for $\mathcal{H}$ and a polynomial algorithm, for a bushy TBox, to find solutions which are irreducible, for an $\mathcal{ALN}$ DL, have been proposed.

The solution to a CAP can be read as why $S \sqsubseteq \mathcal{D}$ does not hold?, what is a possible explanation with respect to the ontology $\mathcal{T}$? In other words $\mathcal{H}$ represents what is expressed, explicitly or implicitly (that is, entailed because of the ontology), in $\mathcal{D}$ and is not present in S, or also which part of $\mathcal{D}$ is not covered by S?

Di Noia et al., (2004) introduced the *rankPotential* algorithm, such that, given a set of $\mathcal{ALN}$ axioms $\mathcal{T}$ and two $\mathcal{ALN}$ concepts S and $\mathcal{D}$ both satisfiable in $\mathcal{T}$, it computes a semantic distance of S from $\mathcal{D}$ with respect to the ontology $\mathcal{T}$.

Notice that we write the distance of $\mathcal{D}$ from S rather than the distance between S and $\mathcal{D}$ because of the non-symmetric behavior of *rankPotential* (see (Di Noia et al., 2004) for further details). Recalling the definition, *rankPotential* (S, $\mathcal{D}$) corresponds to a numerical measure of what is still missing in S w.r.t. $\mathcal{D}$. If S = $\top$ we have the maximum value for *rankPotential* (S, $\mathcal{D}$), that is the maximum (potential) mismatch of S from $\mathcal{D}$.

In order to better illustrate the approach we consider the following trivial example, where the ontology is just a simple taxonomy[2]: $\mathcal{T}$ = { B⊑A , C⊑B ⊓ E, D≡A ⊓ F}

With respect to the previously defined $\mathcal{T}$ consider the service request $\mathcal{D}$ = C ⊓ D and the Web service description S = E ⊓ B ⊓ G. Referring to the above classification we note that $\mathcal{D}$ and S are in a potential match. Unfolding $\mathcal{T}$ in both $\mathcal{D}$ and S we obtain $\mathcal{D}$ = C ⊓ B ⊓ A ⊓ E ⊓ F and S = E ⊓ B ⊓ A ⊓ G. Since in the ontology the third one is an equivalence axiom, we rewrite D with A ⊓ F instead of expanding it as for B and C. Once we have the unfolded version of $\mathcal{D}$ and S we can observe that two pieces of information {C, F} are missing in S in order to completely satisfy $\mathcal{D}$ (and then reach a full match). Since the maximum number of missing pieces of information is equal to the length of the unfolded $\mathcal{D}$, in this case 5, we assign a normalized semantic similarity score of (1 −2/5) to the previous example match.

[2] For the sake of simplicity in this example we do not consider roles even if *rankPotential* is able to deal with them for $\mathcal{ALN}$ ontologies.

The value returned by rankPotential($\mathcal{T}$, $\top$,$\mathcal{D}$) hence amounts to how specific is a complex concept expression $\mathcal{D}$ with respect to an ontology $\mathcal{T}$, what we call the depth of $\mathcal{D}$: *depth($\mathcal{D}$)*. Such a measure is not trivially the depth of a node in a tree for at least two main reasons:
1. An $\mathcal{ALN}$ ontology, typically, is not a simple terms taxonomy tree, i.e., its structure is not limited to simple IS-A relations between two atomic concepts[3].
2. An $\mathcal{ALN}$ complex concept is generally the conjunction of both atomic concepts and role expressions.

It is straightforward to show that, given a concept description C and an $\mathcal{ALN}$ ontology $\mathcal{T}$, rankPotential($\mathcal{T}$,$\top$, C) can be used as a total order relation over $\mathcal{ALN}$, which takes into account the domain knowledge modeled within the ontology $\mathcal{T}$. How can a solution to a CAP be useful in semantic Web services discovery and composition? It is intuitive that $\mathcal{H}$, in a CAP solution, is an explanation hypothesis for the missing part in the available service S needed to completely satisfy a request $\mathcal{D}$. A not-full match with the request is then due to $\mathcal{H}$. An observation we can also express defining $\mathcal{H}$ as: what is not covered by S with respect to $\mathcal{D}$.

Based on this last remark we use solutions to sets of CAP to solve Concept Covering Problems as it is shown in the following. In particular, we show that we are able, through Concept Abduction, to extend the definition of Concept Covering to expressive DLs, and we introduce a greedy algorithm to compute a Concept Covering in terms of Concept Abduction (Colucci et al., 2005a; Colucci et al., 2005b). Such algorithm provides as output the obtained covering and, when such covering does not exist, also a logical explanation on what remains missing in the request.
Let us start recalling that in (Hacid et al., 2002) the best covering problem in DLs was introduced as "...a new instance of the problem of rewriting concepts using terminologies", and proposed in (Benatallah et al., 2004) to cope with automated semantic Web service composition. In order to define a concept covering two non standard inference services were used: the least common subsumer (*lcs*) (F.Baader, 2003) and the difference or subtraction operation (Teege, 1994). Unfortunately, as the authors admitted, concept difference (Teege, 1994), can be used only for DLs that have structurally unique Reduced Clause Forms (RCF), and the logic they adopted, namely $\mathcal{FL}_0 \cup$ ($\geq$ nR) has limited expressiveness w.r.t. other DLs, e.g., $\mathcal{ALN}$ (for a complete description see (Teege, 1994)). In a more formal way Hacid et al., (2002) defined cover as follows.

Definition 2 *Let $\mathcal{L}$ be a Description Logic with structural subsumption, $\mathcal{T}$ be a terminology using operator allowed by $\mathcal{L}$, $\mathcal{R}$ be the set of concept definitions in $\mathcal{T}$, $\mathcal{R} = \{S_i, i \in [1..n]\}$, and $\mathcal{D}$ be a concept in $\mathcal{L}$ such that $\mathcal{T} \not\models \mathcal{D} \equiv \bot$. A cover of a $\mathcal{D}$ using $\mathcal{T}$ is finding a set $\mathcal{R}_C \subseteq \mathcal{R}$ such that considering $\sqcap S_j$, conjunction of all the $S_j \in \mathcal{R}_C$, $D - \mathrm{lcs}_{\mathcal{T}}(\mathcal{D}, \sqcap S_j) \not\equiv D$ occurs.*

That is, a cover is finding a set of concepts defined in $\mathcal{T}$ such that they contain the information in $\mathcal{D}$. Notice that a DL with structural subsumption and Reduced Clause Forms (RCF) is needed in

[3] It can be better represented as a labeled graph.

order to use concept difference. We start presenting an extension to the previous definition of a Concept Covering, which withdraws limitations on the adopted DL, by exploiting Concept Abduction.

Definition 3 *Let $\mathcal{D}$ be a concept, $\mathcal{R} = \{S_1, S_2, ..., S_k\}$ be a set of concepts, and $\mathcal{T}$ be a set of axioms representing an ontology, all in a DL $\mathcal{L}$, where $\mathcal{D}$ and $S_1, S_2, ..., S_k$ are satisfiable in $\mathcal{T}$. Let also $\prec_T$ be an order relation over $\mathcal{L}$ taking into account the ontology $\mathcal{T}$.*

The Concept Covering Problem (CCoP) denoted as $\mathcal{V}$=<$\mathcal{L}$, $\mathcal{R}$, $\mathcal{D}$, $\mathcal{T}$> is finding a pair <$\mathcal{R}_C$, $\mathcal{H}$> such that:

1. $\mathcal{R}_C \subseteq \mathcal{R}$, and $\mathcal{T} \not\models \sqcap_{S_i \in \mathcal{R}_C} S_i \equiv \bot$;
2. $\mathcal{H} \in SOL(<\mathcal{L}, S, \mathcal{D}, \mathcal{T}>)$, and $\mathcal{H} \prec_T \mathcal{D}$.

We call <$\mathcal{R}_C$, $\mathcal{H}$> a solution for $\mathcal{V}$, and say that $\mathcal{R}_C$ covers $\mathcal{D}$. Finally, we denote SOLCCoP ($\mathcal{V}$) the set of all solutions to a CCoP $\mathcal{V}$.

In the above definition the elements for the solution <$\mathcal{R}_C$, $\mathcal{H}$> of a CCoP represent respectively:

- $\mathcal{R}_C$. Which concepts in $\mathcal{R}$ represent the cover for D w.r.t. $\mathcal{T}$.
- $\mathcal{H}$. What is still in $\mathcal{D}$ and is not covered by concepts in $\mathcal{R}_C$.

Intuitively, $\mathcal{R}_C$ is a set of concepts that completely or partially cover $\mathcal{D}$ w.r.t. $\mathcal{T}$, while the abduced concept $\mathcal{H}$ represents what is still in $\mathcal{D}$ and is not covered by $\mathcal{R}_C$. It is worth noticing that a Concept Covering Problem is similar, but has remarkable differences when compared to classical set covering. *CCoP* is not trivially a different formulation of a classical minimal set covering (SC) problem, as an exact solution to a CCoP may not exist. Furthermore in SC elements are not related with each other, while in CCoP elements are related with each other through axioms within the ontology, and while the aim of an SC is to minimize the cardinality of $\mathcal{R}_C$ the aim of a CCoP is to maximize the covering, hence minimizing $\mathcal{H}$.

There can be several solutions for a single CCoP, depending also on the strategy adopted for choosing concepts in $\mathcal{R}_C$. This consideration leads to the definition of *best cover* and *exact cover*.

Definition 4 *A best cover for a CcoP $\mathcal{V}$, w.r.t. an order $\prec_T$ on the DL adopted, is a solution <$\mathcal{R}_C$, $\mathcal{H}b$> for $\mathcal{V}$ such that there is no other solution <$\mathcal{R}'_C$, $\mathcal{H}'$> for $\mathcal{V}$ with $\mathcal{H}' \prec_T \mathcal{H}b$.*

There is no solution <, $\mathcal{T}$> $\mathcal{R}'_C$ $\mathcal{H}'$ for $\mathcal{V}$ such that $\mathcal{H}'$, the remaining part of $\mathcal{D}$ still to be covered, is *smaller* than $\mathcal{H}b$ with respect to an order $\prec_T$. The definition of some minimality criteria is in (Di Noia et al., 2003; Colucci et al., 2003; Di Noia et al., 2004, Di Noia et al., 2007). Observe that, since the solution for a Concept Abduction Problem is not unique in general, there could be two solutions <$\mathcal{R}_C$, $\mathcal{H}1$>, <$\mathcal{R}_C$, $\mathcal{H}2$> such that the first one is a best cover, while the

second one is not. However, when a full cover exists, it is independent of any order $\prec_{\mathcal{T}}$ on the DL adopted. The *rankPotential* algorithm presented by Di Noia et al., (2004) can be used to compute an order $\prec_{\mathcal{T}}$ for $\mathcal{ALN}$.

Definition 5 *An exact cover for a CcoP* $\mathcal{V}$ *is a solution* $< \mathcal{R}_c, \mathcal{He}>$ *for* $\mathcal{V}$ *such that* $\mathcal{T} \models \mathcal{He} \equiv \top$.

It is well-known that even the basic set covering problem is NP-hard; Concept Covering to be of any practical use in a Semantic Web services composition process has to be reasonably fast. With this issue in mind we devised a tractable greedy Concept Covering algorithm, which is polynomial for $\mathcal{ALN}$[4], building on and extending a classical greedy set covering one (Cormen et al., 1990).

Algorithm $\mathcal{GREEDYsolveCCoP}$ $(\mathcal{R}, \mathcal{D}, \mathcal{T})$
input concepts $\mathcal{D}$, $S_i \in \mathcal{R}$, i = 1..k, where $\mathcal{D}$ and S_i are satisfiable in $\mathcal{T}$
output $<\mathcal{R}_c, \mathcal{H}>$
begin algorithm
$\mathcal{R}_c = \emptyset$;
$\mathcal{D}u = \mathcal{D}$;
$\mathcal{H}min = \mathcal{D}$;
do
$S_{MAX} = \top$;
/* [♣] Perform a greedy search among $S_i \in \mathcal{R}$ */
for each $S_i \in \mathcal{R}$
if $\mathcal{R}_c \cup \{ S_i \}$ is a cover for $\mathcal{D}u$ **then**
$\mathcal{H}$ = solveCAP ($<\mathcal{L}, S_i, \mathcal{D}u, \mathcal{T}>$);
/* [♦] Choose S_i based on an order $\prec_{\mathcal{T}}$ */
if $\mathcal{H} \prec \mathcal{H}min$ **then**
$S_{MAX} = S_i$;
$\mathcal{H}min = \mathcal{H}$;
end if
end if
end for each
/* [♠] If a new S_i is found then add S_i to $\mathcal{R}_c$ and remove it from $\mathcal{R}$ */
if $\mathcal{T} \models S_{MAX} \not\equiv \top$ **then**
$\mathcal{R} = \mathcal{R} \backslash \{ S_i \}$;
$\mathcal{R}_c = \mathcal{R}_c \cup \{ S_i \}$;

[4] Being greedy, the computed solution will not be optimal in general (neither a full cover, nor a best one).

$\mathcal{D}u = \mathcal{H}min$;

end if

/* [♥] Continue searching until no S_i is found */

while($S_{MAX} \not\equiv \top$);

return $<\mathcal{R}_C, \mathcal{D}u>$;

end algorithm

The algorithm tries to cover $\mathcal{D}$ as much as possible, using the concepts $S_i \in \mathcal{R}$.

♥ If it is not found any new useful $S_i \in \mathcal{R}$, that is any S_i such that it covers $\mathcal{D}$ more, then the algorithm terminates.

♣ A greedy approach is used to choose candidates for $\mathcal{R}_C$.

♦ Choose among the candidates the one such that $\mathcal{H}$, solution for the local CAP, is minimal w.r.t. the order $\prec_T$.

♠ If the greedy search returns a new S_i, it is removed from R and added to $\mathcal{R}_C$.

In (Cormen et al., 1990) it is proved that, for a set covering problem, the solution grows logarithmically in the size of the set to be covered with respect to the minimal one. In *GREEDYsolveCCoP* the novel complexity source is in the solution of the CAPs and the comparison in [♦]. For $\mathcal{ALN}$, in (Di Noia et al., 2003) a polynomial algorithm (*findIrred*) is proposed to find irreducible solutions for a CAP, and in (Di Noia et al., 2004) the tractable *rankPotential* is presented to rank concepts and establish an order relation among them. It is therefore trivial proving that also *GREEDYsolveCCoP* can be solved in polynomial time. Obviously we are not claiming that we solve a covering problem polynomially. The algorithm returns a cover, not the best one.

SEMANTIC WEB SERVICE DISCOVERY AND COMPOSITION

In this section we build on the previously introduced services and algorithms to carry out a fully Semantic Web service discoveryand composition. Let us point out that we refer to a general framework, compliant with all the ones modeling Semantic Web services with respect to a Input-Output-Precondition-Effect model. This implies that our approach can be reformulated using OWL-S (http://www.w3.org/Submission/OWL-S/) model with practically no changes. Notice that the need for common, shared, ontologies is usually the first objection towards approaches to SWS discovery and composition, a claim that obviously undermines the whole Semantic Web initiative. Nevertheless, it should be considered that effective integration approaches are being proposed for descriptions expressed in heterogeneous forms (Madhavan et al., 2001; Calì et al., 2004), while specific initiatives such as WSMO[5] also stress the role of Mediators to ease interoperability between heterogeneous service descriptions. As stated above, we believe that the service

[5] www.wsmo.org

discovery process is a subproblem of the more generic resource retrieval one: ”having a request and several available resources potentially matching it, if it is not possible to retrieve some resources so that they completely satisfy the request, is an approximate solution possible? Is it possible to have explanation on such approximation?”

To explain and motivate the approach, we start with a simple semantic web-service model and then add features, enriching the model. In the initial model we define both the request $\mathcal{D}$ and the description $\mathcal{WS}_{\mathcal{D}}$ of each Web service as DL concept descriptions w.r.t. an ontology $\mathcal{T}$. A classification schema such as UNSPSC can then be used to classify tasks and related ontologies (Colucci et al., 2004). We assume the existence of a registry where service providers store, for each service, both all the information needed for the invocation and the description $\mathcal{WS}_{\mathcal{D}}$ of the service itself. Notice that, as it is also stated in the OWL-S overview, (http://www.w3.org/Submission/OWL-S/), assuming the existence of a registry does not limit the approach.

Given a request $\mathcal{D}$ modeled w.r.t. an ontology $\mathcal{T}$, the steps needed in order to obtain a set of services satisfying as much as possible $\mathcal{D}$ are hence the following:

1. query the registry in order to obtain all the service descriptions $\mathcal{WS}_{\mathcal{D}}$ that refer to the same $\mathcal{T}$;
2. put all the retrieved services descriptions in a set $\mathcal{R}$;
3. call *solveCCoP* ($\mathcal{R}$, $\mathcal{D}$, $\mathcal{T}$);
4. with reference to $<\mathcal{R}_{\mathcal{C}}, \mathcal{H}>$ returned by *solveCCoP* in the previous step, return to the requester both $\mathcal{H}$ and, for each $\mathcal{WS}_{\mathcal{D}} \in \mathcal{R}_{\mathcal{C}}$, the invocation information of the corresponding service.

$\mathcal{R}_{\mathcal{C}}$ and $\mathcal{H}$ are, respectively, the set of resources representing an approximate solution to the retrieval problem for a composite Web service, and an explanation on why the solution is not an exact one.

Using the above approach, a discovery and selection is possible for the services available in the registry, which can be composed in order to satisfy $\mathcal{D}$ as far as possible.

Obviously, proposing a composition of the discovered services in the retrieved set requires taking into account also their execution information, i.e., inputs, outputs, preconditions and effects specification. Without loss of generality, in what follows we will consider only preconditions and effects (results), as it is straightforward to extend the approach also considering inputs and outputs.

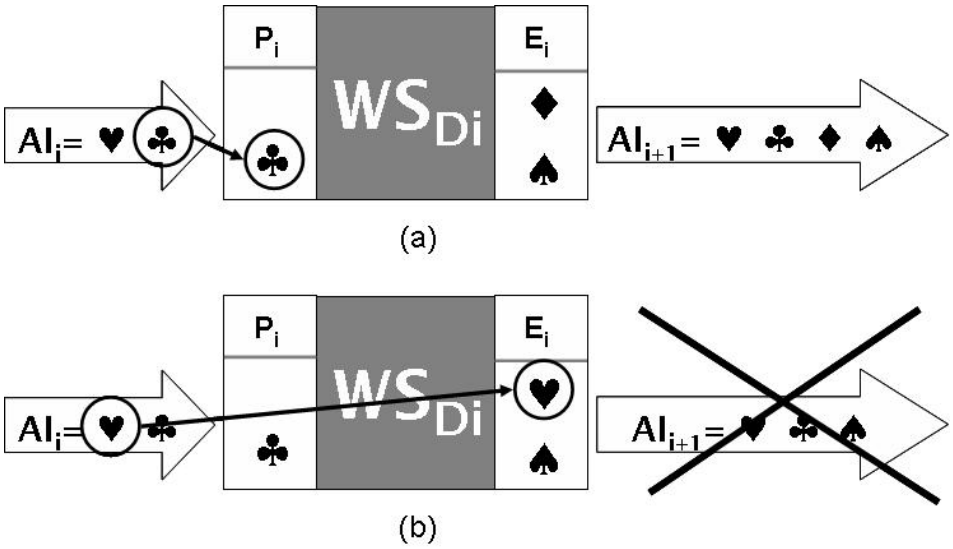

Figure 1: The role of preconditions and effects during Web service composition.

Dealing with Preconditions and Effects

In order to execute a Web service, its preconditions must be satisfied, possibly using information provided by other Web services. Moreover care has to be paid in avoiding the duplication of effects when composing services, which might be due to entailment relationships among different effects provided by services being composed. After the execution of the composite service we do not want two or more services providing the same results, even partially.
Turning to the classical scenario proposed in (Frauenfelder, 2001), a booking agent organizing a trip and composing two services, one able to book both a hotel stay and a flight and another a flight and a car rental, would not be much appreciated if its outcome is two flights booked for the same trip, together with the hotel and the car.
In order to deal with the execution information, we extend the Web service model and define:

Request: a pair $<\mathcal{D}, \mathcal{P}_O>$, where $\mathcal{D}$ is the description of the requested service and $\mathcal{P}_O$ are the preconditions provided with the request. For example, if the request is made by a personal user agent, then it is able to provide some initial information to be used as preconditions for a service execution, e.g., a valid credit card.

Web service[6]**:** a triple $<\mathcal{WS}_\mathcal{D}, \mathcal{P}, \mathcal{E}>$, where $\mathcal{WS}_\mathcal{D}$ represents provided service description, $\mathcal{P}$ the preconditions and $\mathcal{E}$ the effects. $\mathcal{WS}_\mathcal{D}$ is a description of the offered service[7]. Using a language endowed with a well-defined syntax and semantics, it models what the service offers. $\mathcal{P}$ and $\mathcal{E}$ are respectively the preconditions and the effects specification.

[6] A Web service implemented by combining other Web services is referred as composite "... to distinguish it from the ones implemented through conventional programming languages and invoking conventional services which are called basic" (Alonso et al., 2003). Since a service being basic or composite is transparent to the client, in the following we address them as Web service for simplicity.

[7] What we call here $\mathcal{WS}_\mathcal{D}$ –Web service Description– can also be interpreted as a description of the effects provided by the Web service itself.

In our setting, $WS_{\mathcal{D}}$ and $\mathcal{D}$ are formulated as DLs concepts w.r.t. the domain ontology $\mathcal{T}d$. For the sake of simplicity, here $\mathcal{P}_O$, $\mathcal{P}$ and $\mathcal{E}$ are modeled as conjunction of atomic concepts represented in the Precondition/Effect ontology $\mathcal{T}p/e$. The following definition and formalization refers to such $\mathcal{T}p/e$ ontology but they can be extended considering other logical formulations for both $\mathcal{P}$ and $\mathcal{E}$. In other words, whatever the logical framework, it has to allow effects duplication detection and the precondition satisfaction by means of other Web service effects.

A simple covering solution, as the one proposed above, cannot deal with the $\mathcal{P}$, $\mathcal{E}$ specifications of the services. To compose Web services dealing with their $\mathcal{P}/\mathcal{E}$, we introduce an informal definition of Web service flow with respect to some initial preconditions $\mathcal{P}_O$, from now on $\mathcal{WSF}(\mathcal{P}_O)$.

$\mathcal{WSF}(\mathcal{P}o)$ is a finite sequence of Web services (ws_1, ws_2, ...ws_i, ..., ws_n), where for each Web service ws_i belonging to the flow, all the following conditions hold:

1. The preconditions of the first Web service ws_1 executed in a composite Web service are satisfied by the information $\mathcal{P}_O$provided with the initial request –see Figure 1(a).
2. In order to execute a Web service ws_i, the available information for preconditions satisfaction comes from the effects of all the Web services ws_j , with j < i, previously executed – see Figure 1(a).
3. Effects duplication after the execution of the composite Web service has to be avoided – see Figure 1(b).

The available information for ws_i is the conjunction of both the effects produced by $ws_j \in \mathcal{WSF}(\mathcal{P}o)$, with j < i, along the execution flow and the initial preconditions $\mathcal{P}_O$.

In a more formal way, indicating with $\mathcal{AI}_i$ the **A**vailable **I**nformation for ws_i and with $\mathcal{E}_j$ the effects produced by ws_j, with j < i, the following relation ensues:

$\mathcal{AI}_i = \mathcal{P}_O \sqcap \mathcal{E}_1 \sqcap \mathcal{E}_2 \sqcap ... \sqcap \mathcal{E}_{i-1}$

Since, for simplicity, we illustrate the approach using a concept taxonomy to represent preconditions and effects, both $\mathcal{P}_i$ and $\mathcal{E}_i$ are expressed as conjunction of concept names $\mathcal{CN}$ introduced in $\mathcal{T}p/e$; $\mathcal{P}_i = \sqcap_j \mathcal{CN}_j$ and $\mathcal{E}_i = \sqcap_k \mathcal{CN}_k$. Now we can formally define a Web service flow.

Definition 6 *A* ***Web service flow*** *with respect to some initial preconditions* $\mathcal{P}_O$ *is a finite sequence of Web services* $\mathcal{WSF}(\mathcal{P}_O) = (ws_1, ws_2, ...ws_i, ..., ws_n)$, *with* i = 1..n, *where for each Web service* $ws_i \in \mathcal{WSF}(\mathcal{P}_O)$ *all the following conditions hold:*

1. for ws_1, $\mathcal{P}_O \sqsubseteq \mathcal{P}_1$.

2. for ws_i, *with* i > 1, $\mathcal{AI}_i \sqsubseteq \mathcal{P}_i$.

3. for ws_i*, with* i > 1*, for each concept name* CN *occurring in* $\mathcal{E}_i$*,* $\mathcal{AI}_i \not\sqsubseteq CN$.

We indicate with $D_{WSF} = \{WS_{\mathcal{D}i} \mid ws_i \in WSF(\mathcal{P}_0)\}$*, the set of Web service descriptions in* $WSF(\mathcal{P}_0)$.

Based on the definition of Web service flow, here we define a composite Web service with respect to a request.

Definition 7 *Let* $\mathcal{R} = \{ WS_{\mathcal{D}i}, \mathcal{P}_i, \mathcal{E}_i \}$*, with* i=1..k*, be a set of Web services* ws_i*, and* $<\mathcal{D}, \mathcal{P}_0>$ *be a request, such that both* $\mathcal{D}$ *and* $WS_{\mathcal{D}i}$ *are modeled as concept descriptions satisfiable w.r.t. a domain ontology* $\mathcal{T}d$*, and* $\mathcal{P}_0$*,* $\mathcal{P}_i$ *and* $\mathcal{E}_i$ *modeled as conjunction of concept names satisfiable w.r.t. a Precondition/Effect ontology* $\mathcal{T}p/e$.
A composite Web service for $<\mathcal{D}, \mathcal{P}_0>$ *with respect to* $\mathcal{R}$*,* $CWS(<\mathcal{D}, \mathcal{P}_0, \mathcal{R}>)$*, is a Web service flow such that for each* ws_j *in the execution flow,* $\mathcal{D}_{CWS(<\mathcal{D},\mathcal{P}0,\mathcal{R}>)} = \{ WS_{\mathcal{D}j} \mid ws_j \in CWS(<\mathcal{D}, \mathcal{P}_0,\mathcal{R}>)\}$*,* ***covers*** $\mathcal{D}$.

A composite Web service is an execution flow such that it can be started using some information provided by the requester ($\mathcal{P}_0$) as initial preconditions, and the provided composite service covers the user request description ($\mathcal{D}$).
In this definition of composite Web Service, we consider a user perspective while handling preconditions. In fact, we suppose the only initial information the user is willing to provide are the ones in $\mathcal{P}_0$.

Computing a Composite Web Service

We now extend *GREEDYsolveCCoP* to cope with Web service preconditions and effects, and determine an algorithm to automatically compute a composite Web service.
For such purpose we also define an *executable Web service* and an *executable set*.

Definition 8 *Given a Web service flow* $WSF(\mathcal{P}_0) = (ws_1, ws_2, ...ws_i, ..., ws_n)$*, we say that a Web service is an* ***executable Web service*** ws^{ex} *for* $WSF(\mathcal{P}_0)$ *iff*

1. $ws^{ex} \notin WSF(\mathcal{P}_0)$

2. $WSF'(\mathcal{P}_0) = (ws_1, ws_2, ...ws_i, ..., ws_n, ws^{ex})$ *is a Web service flow*.

An executable Web service ws^{ex} for $WSF(\mathcal{P}_0)$ is a Web service which can be invoked after the execution of $WSF(\mathcal{P}_0)$, i.e., its preconditions are satisfied after the execution of $WSF(\mathcal{P}_0)$ and such that its effects are not already provided by $WSF(\mathcal{P}_0)$, and its description is compatible with the service descriptions in $\mathcal{D}_{CWS(<\mathcal{D},\mathcal{P}0,\mathcal{R}>)}$.

Definition 9 *Given a Web service flow* $WSF(P_O)$ *and a set of Web services* $\mathcal{R} = \{ ws_i \}$ *we call* ***executable set*** *of* $WSF(P_O)$, *the set of all the* $ws_i \in \mathcal{R}$ *such that* ws_i *is an executable service for* $WSF(P_O)$ }. $EX_{WSF(P_O)} = \{ ws_i^{ex} \mid ws_i^{ex}$ *is an executable service for* $WSF(P_O)\}$

The executable set is hence the set of all the services that can be invoked after the execution of a Web service flow. Based on the above introduced definitions we present an algorithm able to compute a composite Web service $CWS(<D, P_O, \mathcal{R}>)$.

Algorithm *serviceComposer*$(\mathcal{R}, <D, P_O>, T')$

input a set of services $\mathcal{R} = \{ ws_i = < WS_{Di}, P_i, E_i>\}$, a request $<D, P_O>$, where D and WS_D are satisfiable in T'

output $<CWS, H>$

begin algorithm

1: $CWS\ (<D, P_O, \mathcal{R}>) = \emptyset$;

2: $Du = D$;

3: $Hmin = D$;

4: **do**

5: **compute** $EX_{CWS(<D, P0, \mathcal{R}>)}$;

6: $WS_{DMAX} = \top$;

7: **for each** $ws_i \in EX_{CWS(<D, P0, \mathcal{R}>)}$

8: **if** $D_{CWS(<D,P0,\mathcal{R}>)} \cup \{ WS_{Di} \}$ covers Du **then** /* ⊗ */

9: $H = solveCAP\ (<L, WS_{Di}, Du, T'>)$; /* ⊙ */

10: **if** $H \prec_{T'} Hmin$ **then** /* ⊘ */

11: $WS_{DMAX} = WS_{Di}$;

12: $Hmin = H$;

13: **end if**

14: **end if**

15: **end for each**

16: **if** $WS_{DMAX} \not\equiv \top$ **then**

17: $\mathcal{R} = \mathcal{R} \setminus \{ ws_i \}$;

18: $CWS\ (<D, P_O, \mathcal{R}>) = (CWS\ (<D, P_O, \mathcal{R}>), ws_i)$;

19: $Du = Hmin$;

20: **end if**

21: **while**$(WS_{Dmin} \not\equiv \top)$;

22: **return** $< CWS\ (<D, P_O, \mathcal{R}>), Duncovered >$;

end algorithm

serviceComposer returns the composite Web service CWS (<$\mathcal{D}$, $\mathcal{P}_O$, $\mathcal{R}$>) and the part of the request description $\mathcal{D}$ that has remained –in case– uncovered, i.e. $\mathcal{D}uncovered$. Notice that, since *serviceComposer* uses a greedy approach, CWS (<$\mathcal{D}$, $\mathcal{P}_O$, $\mathcal{R}$>) represents the quickest solution to the orchestration problem, and since *Duncovered* is computed solving a CAP, it is not ***the*** explanation for the uncovered part of the request, it is ***an*** explanation for that, which depends also on the minimality criterion adopted to solve the CAPs during the algorithm execution. Differently from *GREEDYsolveCCoP*, in *serviceComposer* we have a new source for complexity due to the computation of $\mathcal{EX}_{CWS(<\mathcal{D}, \mathcal{P}_0, \mathcal{R}>)}$ in row 5. The complexity of such a computation is strongly dependent on the language used to model both $\mathcal{P}$ and $\mathcal{E}$; if $\mathcal{EX}_{CWS(<\mathcal{D}, \mathcal{P}_0, \mathcal{R}>)}$ is computed polynomially, *serviceComposer* remains polynomial in time for $\mathcal{ALN}$.

Kayak ⊑ EcoTour ; EcoTour ⊑ Tour ; ThemeParksTour ⊑ ParksTour ; ParksTour ⊑ Tour

SwimmingPool ⊑ FitnessFacilities ; FitnessCenter ⊑ FitnessFacilities

SmokingAllowed ⊑ RoomFacilities ; NoSmoking ⊑RoomFacilities ; disj(SmokingAllowed, NoSmoking) ;
PetsAllowed ⊑ RoomFacilities ; NoPetsAllowed ⊑ RoomFacilities ; disj(PetsAllowed, NoPetsAllowed)

FitnessFacilities ⊑ HotelFacilities ; RoomFacilities ⊑ HotelFacilities ;

AirportTransfer ⊑ FlightFacilities

GPS ⊑ CarRentalFacilities ; EmergencyAssistance ⊑ CarRentalFacilities ;
ExpressReturn ⊑ CarRentalFacilities

HotelFacilities ⊑ Facilities ; FlightFacilities ⊑ Facilities ;

CarRentalFacilities ⊑ Facilities ; disj(HotelFacilities, FlightFacilities,CarRentalFacilities)

BedRoom ⊑ ∃hasBeds

DoubleRoom ≡ BedRoom ⊓ (= 2 hasBeds)

SingleRoom ≡ BedRoom ⊓ (= 1 hasBeds)

WeddingRoom ≡ DoubleRoom ⊓ ∃hasBeds.DoubleBed

Figure 2: The reference ontology for the example

AN ILLUSTRATIVE EXAMPLE

In order to describe the *serviceComposer* behavior we model a simple travel reservation scenario. The request description $\mathcal{D}$ and the Web service descriptions $\mathcal{WS}_{\mathcal{D}i}$ are modeled with respect to the toy ontology $\mathcal{T}$ pictured in Figure 2.

Our system (see next Section) implements *findIrred* (Di Noia et al., 2003) to solve Concept Abduction Problems (⊙ in *serviceComposer*) and *rankPotential* (Di Noia et al., 2004) to rank concepts (⊘ in *serviceComposer*), the results presented below refer to such algorithms.

From now on we write H_{WSname} to express that H is a solution to the CAP <$\mathcal{L}$, WS_{nameD}, $\mathcal{D}u$, $\mathcal{T}$>, computed using findIrred and $|H_{WSname}|$ to denote the solution computed by *rankPotential*(WS_{nameD}, $\mathcal{D}u$).

Imagine you wish to plan your holiday in Florida, particularly in the Orlando neighborhood, you want to book a flight and reserve a room in a hotel where pets are allowed and as you would like to move around in Florida, you also want to rent a car with satellite navigation system. Furthermore you wish to try some attraction like Park Tour and Eco Tour. You provide a VALID_CREDIT_CARD, as information for Web service accessing.

Your request can be formalized as follows:

$< \mathcal{D}$ = ∃flightBooking ⊓ ∃hotelReservation ⊓ ∀hotelReservation.(∃withFacilities⊓ ∀withFacilities.PetsAllowed) ⊓ ∃carRenting ⊓ ∀carRenting.(∃withFacilities⊓∀withFacilities.GPS) ⊓ ∃attraction ⊓ ∀attraction.(ParksTour ⊓ EcoTour),

$\mathcal{P}o$ = VALID_CREDIT_CARD >

Let us suppose the registry $\mathcal{R}$ stores the following Web services descriptions, modeled in accordance with the ontology in Figure 2.

$\mathcal{R}$ = {*FloridaTour, SuperCar, HotelBooker, NiceHolidays, MyCar*}

FloridaTour is a Web service which includes, among its options, hotel reservation, flight booking with airport transfer and tour around theme parks;

<FloridaTour = ∃hotelReservation⊓∃flightBooking⊓∀flightBooking.(∃withFacilities⊓∀withFacilities.AirportTransfer) ⊓ ∃attraction ⊓ ∀attraction.ThemeParksTour,

$\mathcal{P}$ = VALID_CREDIT_CARD,

$\mathcal{E}$ = FLIGHT_RESERVATION ⊓ HOTEL_RESERVATION ⊓ ATTRACTION_TICKET>

SuperCar is a car rental service that includes a satellite navigation system and ensures emergency roadside assistance;

<SuperCar=∃carRenting⊓∀carRenting.(∃withFacilities⊓ ∀withFacilities.(GPS⊓EmergencyAssistance)),

$\mathcal{P}$ = VALID_CREDIT_CARD,

$\mathcal{E}$ = CAR_RESERVATION>

HotelBooker is a Web service that allows to reserve a three diamonds hotel, but pets are absolutely not allowed;

<HotelBooker = ∃hotelReservation ⊓ ∀hotelReservation.(= 3 diamonds)⊓ (∃withFacilities ⊓ ∀withFacilities.NoPetsAllowed),

$\mathcal{P}$ = VALID_CREDIT_CARD,

$\mathcal{E}$ = HOTEL_RESERVATION >

NiceHoliday reserves rooms in hotels at least three diamonds, organizing a kayak tour;

<NiceHoliday = ∃hotelReservation ⊓ ∀hotelReservation.≥ 3 diamonds⊓ ⊓∃attraction ⊓ ∀attraction.Kayak,

$\mathcal{P}$ = VALID_CREDIT_CARD,

E = HOTEL_RESERVATION ⊓ ATTRACTION_TICKET>

MyCar is a car rental service offering also express return and emergency roadside assistance;

<MyCar =∃carRenting⊓∀carRenting.(∃withFacilities⊓∀withFacilities.(ExpressReturn⊓ EmergencyAssistance)),
P = VALID_CREDIT_CARD,
E = CAR_RESERVATION>

In order to compute a CWS (<$\mathcal{D}$, $\mathcal{P}_O$, $\mathcal{R}$>) the first step is to evaluate $\mathcal{EX}_{CWS(<\mathcal{D}, \mathcal{P}_O, \mathcal{R}>)}$ with respect to an empty Web service flow CWS (<$\mathcal{D}$, $\mathcal{P}_O$, $\mathcal{R}$>) = ∅ ($\mathcal{EX}_0$ for short). As $\mathcal{P}_O$ = VALID_CREDIT_CARD then all the services will be in the executable set $\mathcal{EX}_0$= {FloridaTour, SuperCar, HotelBooker, NiceHolidays, MyCar}.
At this initial step $\mathcal{D}uncovered$ = $\mathcal{D}$ then:

$H_{FloridaTour}$=∀hotelReservation.(∃withFacilities⊓∀withFacilities.PetsAllowed)⊓∃carRenting⊓∀carRenting.(∃withFacilities⊓∀withFacilities.GPS)⊓∀attraction.EcoTour

| $H_{FloridaTour}$ | = 6

$H_{SuperCar}$=∃flightBooking⊓∃hotelReservation⊓∀hotelReservation.(∃withFacilities ⊓∀withFacilities.PetsAllowed)⊓ ∃attraction⊓ ∀attraction. (ParksTour ⊓ EcoTour)

| $H_{SuperCar}$ |= 7

$H_{HotelBooker}$ =NOT_COMPUTED

$H_{hotelBooker}$ is not computed because $\mathcal{D}uncovered$ and $HotelBooker_{\mathcal{D}}$ are not consistent with each other, i.e. $\mathcal{D}uncovered \sqcap HotelBooker_{\mathcal{D}} \equiv \bot$, and consequently this Web service cannot belong to a covering solution; in fact we are looking for a hotel were pets are allowed, while HotelBooker does not admit pets.

$H_{niceHoliday}$ = ∃flightBooking⊓∀hotelReservation.(∃withFacilities⊓∀withFacilities.PetsAllowed) ⊓∃carRenting⊓ ∀carRenting.(∃withFacilities ⊓∀withFacilities.GPS) ⊓∀attraction.ParksTour

| $H_{niceHoliday}$ |= 7

H_{MyCar} = ∃flightBooking⊓∃hotelReservation⊓∀hotelReservation.(∃withFacilities⊓∀withFacilities.PetsAllowed) ⊓ ∃attraction ⊓ ∀attraction.(ParksTour ⊓ EcoTour)⊓∀carRenting.(∀withFacilities.GPS)

| H_{MyCar} |= 8

Among the services in $\mathcal{EX}_0$, *FloridaTour* will be chosen because of the smallest length of $\mathcal{H}$. After the execution of the first iteration in the algorithm [rows 7-16] we have:

- *CWS* (<$\mathcal{D}$, $\mathcal{P}_O$, $\mathcal{R}$>) = (*FloridaTour*)
- $\mathcal{AI}_1$=FLIGHT_RESERVATION⊓HOTEL_RESERVATION⊓ATTRACTION_TICKET⊓VALID_CREDIT_CARD
- $\mathcal{D}uncovered$=∀hotelReservation.(∃withFacilities⊓∀withFacilities.PetsAllowed)⊓∃carRenting⊓∀carRenting.(∃withFacilities⊓∀withFacilities.GPS)⊓ ∀attraction.EcoTour

With respect to $\mathcal{AI}_1$, the new executable set $\mathcal{EX}_1$ is now:

$\mathcal{EX}_1$= {*MyCar, SuperCar*}

Notice that *NiceHoliday* and *HotelBooker* are not in $\mathcal{EX}_1$ because the effects they provide are already in $\mathcal{AI}_1$.

$H_{SuperCar}$=∀hotelReservation.(∃withFacilities⊓∀withFacilities.PetsAllowed) ⊓∀attraction.EcoTour
| $H_{SuperCar}$ |= 3

H_{MyCar}=∀hotelReservation.(∃withFacilities⊓∀withFacilities.PetsAllowed) ⊓∀carRenting.(∀withFacilities.GPS) ⊓ ∀attraction.EcoTour
| H_{MyCar} |= 4

SuperCar will be added to *CWS* (<$\mathcal{D}$, $\mathcal{P}_O$, $\mathcal{R}$>) because of | $H_{SuperCar}$ |<| H_{MyCar} |, thus obtaining:

- *CWS* (<$\mathcal{D}$, $\mathcal{P}_O$, $\mathcal{R}$>) = (*FloridaTour, SuperCar*)
- $\mathcal{AI}_2$=FLIGHT_RESERVATION⊓HOTEL_RESERVATION⊓ATTRACTION_TICKET⊓CAR_RESERVATION⊓VALID_CREDIT_CARD
- $\mathcal{D}uncovered$=∀hotelReservation.(∃withFacilities⊓∀withFacilities.PetsAllowed)⊓∀attraction.EcoTour

The next executable set ($\mathcal{EX}_2$) is:

$\mathcal{EX}_2$= ∅

The final result pair is then:

<(*FloridaTour,SuperCar*),∀hotelReservation.(∃withFacilities⊓∀withFacilities.PetsAllowed) ⊓∀attraction.EcoTour>

That is, with respect to the information provided by the user, the system is able to compute the candidate composite Web service (*FloridaTour, SuperCar*), but it cannot know if FloridaTour service is able both to reserve hotels where pets are allowed and enjoy an eco tour. Nothing is

specified about that in the description of selected Web services. Notice that missing information can be used to initiate a dialogue between the user and the orchestrator.

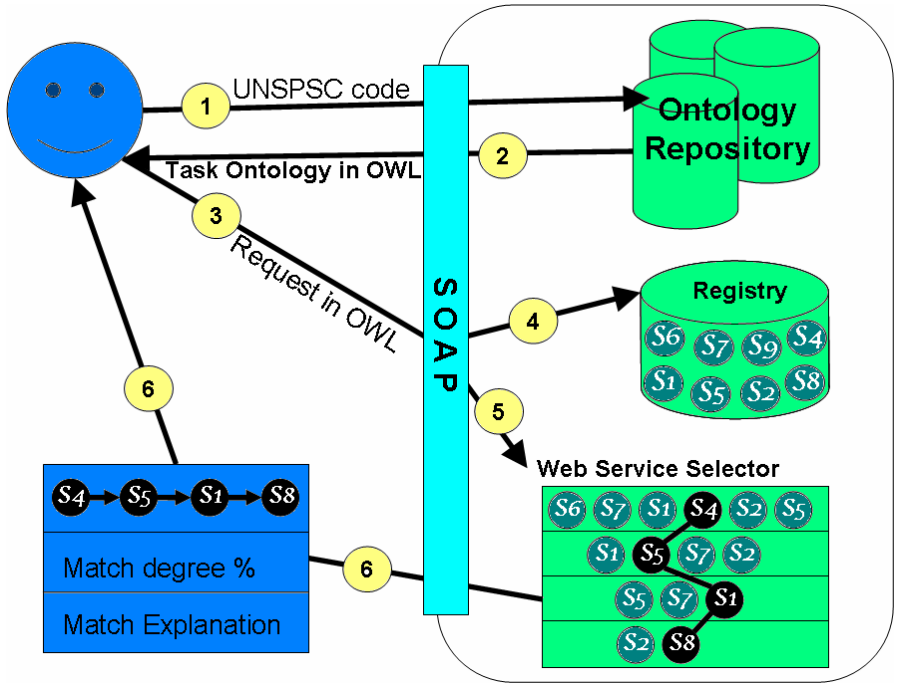

Figure 3: Schematic of the Web services Orchestrator Architecture

SYSTEM AND EXPERIMENTS

We implemented a testbed prototype, whose architecture is sketched in Figure 3. The framework includes `jUDDI+`, an extended version of the open source UDDI implementation by the Apache Software Foundation – jUDDI[8]– able to cope with OWL based annotation of Web services. The implementation allowed to validate the approach, test algorithms behavior, and carry out large scale experiments. We model each Web service specification as an OWL-S 1.1 Profile instance, although as previously hinted, the approach could be implemented in WSDL-S[9] with minor modifications. In order to deal with the Web service semantic description $\mathcal{D}$ we extended the `Result` class in the OWL-S 1.1 Process by means of a new `<owl:ObjectProperty rdf:ID="effectsDescription"/>`[10].

```
<owl:ObjectProperty rdf:ID="effectsDescription">
<rdfs:label>effectsDescription</rdfs:label>
<rdfs:domain rdfs:resource="#Result"/>
<rdfs:range rdfs:resource="owl:Thing"/>
</owl:ObjectProperty>
```

Figure 4: <owl:ObjectProperty rdf:ID="effectsDescription"> definition.

[8] `http://ws.apache.org/juddi/`
[9] `http://www.w3.org/Submission/WSDL-S/`
[10] This property was introduced in OWL-S 0.9 but disappeared in its subsequent 1.0 version.

The property has OWL-S `Result` class as domain and `<owl:Thing/>` as range; the cardinality is restricted to 1, because (in this initial prototype) its specification refers to a description w.r.t. a task ontology identified by a unique UNSPSC code. The code in Figure 4 is the only modification to the standard OWL-S 1.1 definition and code.

```
<profile:serviceName>
        Service Name
</profile:serviceName>
<profile:textDescription>
        A human-understandable description
        of the service.
</profile:textDescription>
<profile:serviceCategory>
        <addParam:UNSPSC rdf:ID="UNSPSC-category">
            <profile:value>
                UNSPSC value
            </profile:value>
            <profile:code>
                UNSPSC code
            </profile:code>
        </addParam:UNSPSC>
</profile:serviceCategory>
<profile:hasPrecondition>
        A logic expression referring to concepts
        in the  TP/Eontology.
</profile:hasPrecondition>
<profile:hasResult>
        <process:Result rdf:ID="HaveSeatResult">
                <process:hasEffect>
                        A logic expression referring to concepts
                        in the  TP/Eontology.
                </process:hasEffect>
                <process:effectDescription>
                        An OWL-DL expression referring to the
                        concepts in the domain ontology
                        identified by the UNSPSC code.
                </process:effectDescription>
        </process:Result>
</profile:hasResult>
```

Figure 5: An OWL-S Web service Profile instance

An example Profile service description is reported in Figure 5. The main component of jUDDI+ is the Web service Selector module (WSS). It uses `MaMaS-tng`[11], a Description Logics based reasoner that exposes non-standard inference services able to cope with incomplete information.

[11] `http://sisinflab.poliba.it/MAMAS-tng/`

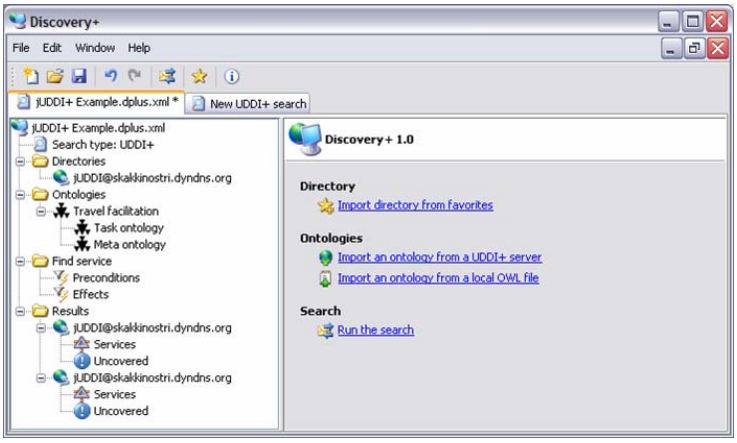

Figure 6: A snapshot of the GUI for service querying to jUDDI+

In our implementation, a user request is a triple < $\mathcal{P}_O$, $\mathcal{E}$, σ>, where $\mathcal{P}_O$ represents the initial preconditions/inputs provided by the user, e.g., `VALID_CREDIT_CARD`, $\mathcal{E}$ the desired effects/outpus after Web service(s) execution, e.g., *flight reservation and double room booking in a hotel with a swimming pool and SPA*, σ is a two-values variable in the set {`discovery`, `composition`} with the obvious meanings.

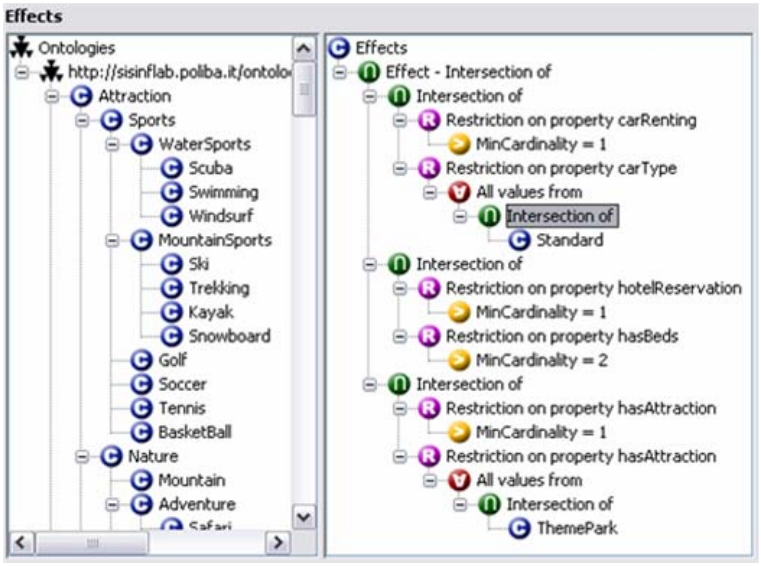

Figure 7: A snapshot of the effects composition interface

In the OWL-S compliant version of jUDDI+, the composition process is performed, using the friendly graphical interface pictured in Figure 6, in the following steps:

1. In the initial step, the agent determines the task ontology that will be used (or the relative URL) selecting it via the UNSPSC code.
2. The selected ontology (or the relative URL) is sent back to the agent.
3. Using the selected ontology, the user formulates a request representing her desired effects E, using its OWL Classes and Properties, see the snapshot in Figure 7. Finally the user

selects if she is interested in a composite service or in discovering a single Web service.The request is then transmitted to the system, together with the initial preconditions $\mathcal{P}_O$ the user is able to provide. From the information provided by the agent, the system selects two types of information: (a) UNSPSC code identifying the ontology and (b) OWL request description and initial preconditions.

4. Using (a) in step 3), the system retrieves the corresponding Web services instances and sends them to WSS together with the corresponding task ontology.
5. The request description and the initial preconditions selected in (b) part of step 3), are sent to WSS.
6. The information is converted to DIG syntax and sent to MaMaS-tng.
7. If the user asked for a composite service able to satisfy her request, then WSS computes: the composite service, the matching degree and, when an approximate solution occurs, an explanation of which part of the request is not provided in the composite service. All these information are returned to the requester. The semantic match degree is computed as: $Match\_degree = 100*(1 - rankPotential(\sqcap i\ WS_{\mathcal{D}i}, \mathcal{D})/depth(\mathcal{D}))$

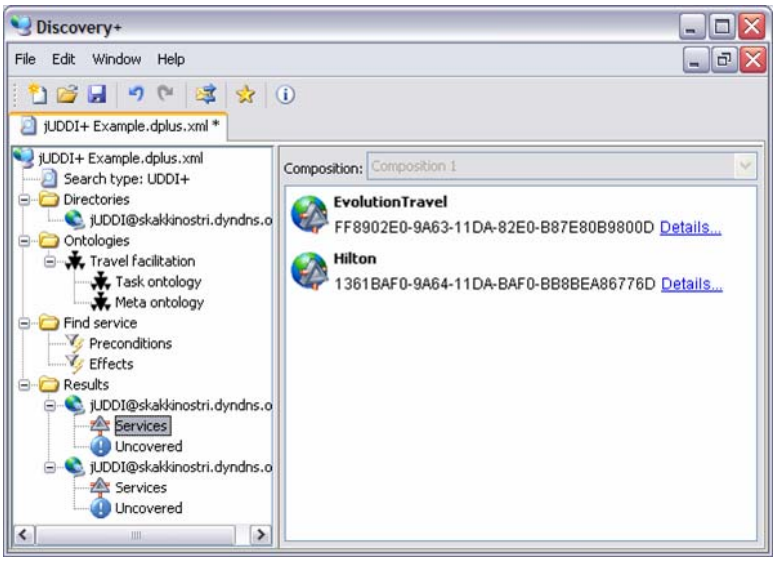

Figure 8: A snapshot of results obtained from jUDDI+.

8. If the user asked for single services able to satisfy her request, then WSS computes, for each WS referring to the same ontology, a match degree with the request. In particular, for each service, a subsumption relation between the user provided preconditions/inputs and WS preconditions/inputs is checked. If the relation holds, the WS is selected as a candidate to satisfy the user request. For each candidate a subsumption relation between the WS preconditions/inputs and user provided preconditions/inputs is checked. If the subsumption holds, then the corresponding WS is ranked as the the best one, else, via MaMaS-tng, the effects/outputs semantic-based covering degree and explanation are computed as a solution to a Concept Abduction Problem. Hence, the set of candidate WSs is ranked with respect to the covering degree and a semantic-based explanation of the match result is also provided to the user. The result of a query for composite service using our graphical inteface is shown in Figure 8.

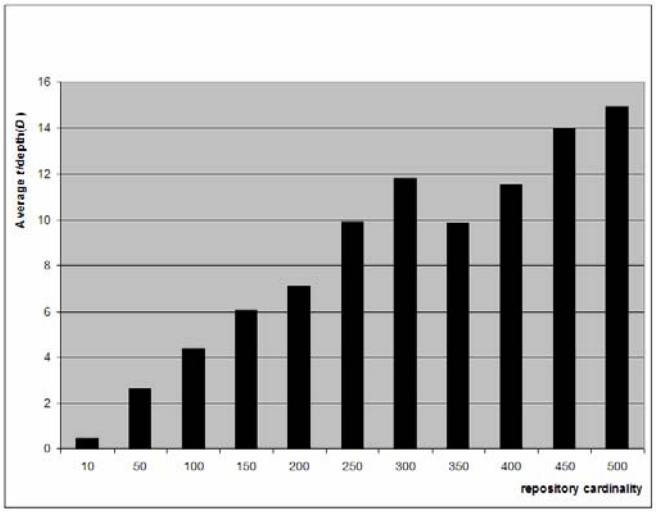

Figure 9: Concept Covering numerical evaluation:t/depth(D) vs. registry cardinality

Experiments have been carried out on the system, to test correctness, performances and scalability, thus numerically validating theoretical complexity results and practical feasibility of our approach.

We randomly generated a set of 7000 OWL-S service descriptions $WS_{\mathcal{D}i}$, where depth($WS_{\mathcal{D}i}$) of the descriptions follows a Gauss distribution with $\mu = 10$ and $\sigma = 20$.

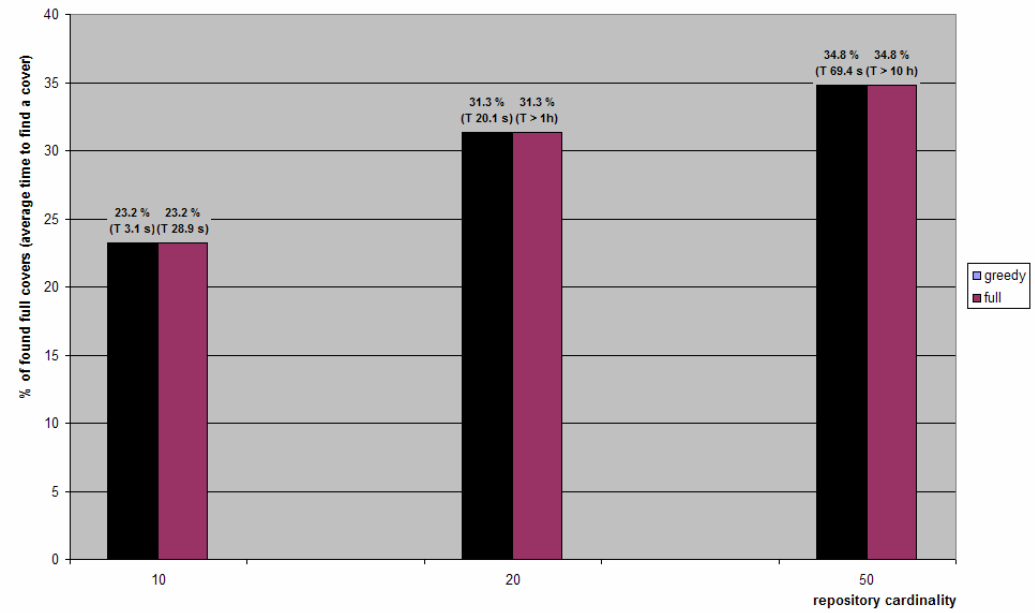

Figure 10: Comparison between the proposed greedy concept covering algorithm and a full covering one. (T: time; s: secs; h: hours)

Tests were carried out using requests having depth 5, 11, 15, 24, and sets of 10, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500 service descriptions, all randomly generated for each iteration.

In Figure 9 simulation results are plotted, showing how average t/depth(D) (time in seconds per request depth) changes w.r.t. the number of services in the registry, while performing a Covering

of a request. Tests were also conducted to experimentally verify the ability of our algorithms to actually find covering services whenever such a cover exists. These were executed adopting as comparison a full covering algorithm, and results showed that, for the set we considered, whenever a cover existed our algorithm would find it with 100 % probability, see Figure 10.
It is noteworthy to recall that we were unable —for practical execution time reasons— to run tests using the full cover algorithm with a dataset larger than fifty services. We anyway admit that for large service sets with cardinality over 1000 also the greedy algorithm takes times that are well over one minute and may become prohibitive in a practical usage of the approach. We believe that to circumvent this issue a feasible solution is a pre-classification on DBMS of services instances along thelines of e.g. (Horrocks et al., 2004), which would definitively releave the reasoning engine from much of the processing.

CONCLUSION AND FUTURE WORK

In this work we proposed a tractable greedy algorithm to perform an automatic semantic Web services composition and we presented motivations, examples and jUDDI+, a framework implementing the presented approach in OWL-S. We extended Concept Covering definition to a more general setting and to more expressive logics and re-defined it in terms of Concept Abduction. We showed how a semantic specification of the service, based not only on the Inputs Outputs Preconditions Effects model but also endowed with a semantic description of the provided functionalities can be helpful in the composition, particularly in the discovery process. Such an approach is also more user-oriented as it makes simpler to devise a request as description of the service actually searched, rather than a specification based exclusively on invocation information. Current and future work are devoted to define suitable pre-classification techniques to handle efficiently large web service sets, to the investigation of an extension to negotiable and strict constraints of the precondition specifications and request (Colucci et al., 2005c), and to studying how to model the composite Web service, computed by serviceComposer, with respect to a BPEL4WS specification. A rule-based system is also under development able to handle rules for Inputs/Outputs/Preconditions/Effects specifications, in order to consider also the concrete Web service during a second step of discovery and composition.

ACKNOWLEGMENT

We acknowledge support of EU FP-6 IST STREP TOWL co. 026896. Some of the authors also acknowledge partial support of projects PE_043 (ACAB-C2) and PS_092 (DIPIS), funded by Apulia Region. We are also grateful to M. Paolucci for useful discussions.

REFERENCES

Alonso, G., Casati, F., Kuno, H., and Machiraju, V. (2003). *Web services: Concepts, Architectures and Applications*. Springer Verlag.
Baader, F., Calvanese, D., Mc Guinness, D., Nardi, D., and Patel-Schneider, P. (2002). *The Description Logic Handbook*. Cambridge University Press.
Benatallah, B., Hacid, M.-S., Leger, A., Rey, C., and Toumani, F. (2004). *On automating Web services discovery*. VLDB Journal, 14(1):84–96.
Benatallah, B., Hacid, M.-S., Rey, C., and Toumani, F. (2003). *Request Rewriting-Based Web Service Discovery*. In volume 2870 of Lecture Notes in Computer Science, pages 242–257. Springer.

Berardi, D., Calvanese, D., Giacomo, G. D., Lenzerini, M., and Mecella, M. (2003). *Automatic composition of E-services that export their behavior*. In volume 2910 of Lecture Notes in Computer Science, pages 43–58. Springer.

Berardi, D., Calvanese, D., Giacomo, G. D., Lenzerini, M., and Mecella, M. (2004). *Synthesis of Underspecified Composite e-Services based on Automated Reasoning.* In Proc. of the 2nd Int. Conf. on Service Oriented Computing (ICSOC), New York City, NY, USA, November 15-18, 2004, pages 105–114.

Cabral, L., Domingue, J., Motta, E., Payne, T. R., and Hakimpour, F. (2004). *Approaches to semantic web services: an overview and comparisons*. In Bussler, C., Davies, J., Fensel, D., and Studer, R., editors, The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004, Heraklion, Crete, Greece, May 10-12, 2004, Proceedings, volume 3053 of Lecture Notes in Computer Science, pages 225–239. Springer.

Calì, A., Calvanese, D., Giacomo, G. D., and M.Lenzerini (2004). *Data integration under integrity constraints*. Information Systems, 29(2):147–163.

Calvanese, D., De Giacomo, G., and Lenzerini, M. (1998). *On the Decidability of Query Containment under Constraints*. In Proceedings of the Seventeenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98), Seattle, Washington, United States June 01 - 04, 1998, pages 149–158.

Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F., and Ragone, A. (2005a). *Semantic-based automated composition of distributed learning objects for personalized e-learning*. In proc. of European Semantic Web Conference, volume 3532, Heraklion, Crete May 29 – June 1, 2005, Lecture Notes in Computer Science, pages 633–648.

Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F., Mongiello, M., Piscitelli, G., and Rossi, G. (2004). *An agency for semantic-based automatic discovery of web-services*. In Artificial Intelligence Applications and Innovations. AIAI- 2004, pages 315–328. Kluwer Academic Publishers.

Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F., and Ragone, A. (2005b). *Fully automated web services orchestration in a resource retrieval scenario.* In Proc. of International Conference on web Services (ICWS '05), July 11-15, 2005 , Orlando , Florida , USA , pages 427–434. IEEE.

Colucci, S., Di Noia, T., Di Sciascio, E.., Donini, F., and Mongiello, M. (2003). *Concept Abduction and Contraction in Description Logics.* In Proceedings of the 16th International Workshop on Description Logics (DL'03), Rome, Italy, September 5-7, 2003 volume 81 of CEUR Workshop Proceedings.

Colucci, S., Di Noia, T., Di Sciascio, E. , Donini, F., and Mongiello, M. (2005c). *Concept Abduction and Contraction for Semantic-based Discovery of Matches and Negotiation Spaces in an E-Marketplace*. Electronic Commerce Research and Applications, 4(4):345–361.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1990). *Introduction to Algorithms*. MIT.

Di Noia, T., Di Sciascio, E., Donini, F. (2007). *Semantic Matchmaking as Non-Monotonic Reasoning: A Description Logic Approach.* The Journal of Artificial Intelligence Research, JAIR, 28:.

Di Noia, T., Di Sciascio, E., Donini, F., and Mongiello, M. (2003). *Abductive matchmaking using description logics.* In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), pages 337–342, Acapulco, Messico. Morgan Kaufmann, Los Altos.

Di Noia, T., Di Sciascio, E., and Donini, F. M. (2004). *Extending Semantic-Based Matchmaking via Concept Abduction and Contraction*. In Engineering Knowledge in the Age of the Semantic Web (EKAW '04), volume 3257 of LNAI, 5-8th October 2004 - Whittlebury Hall, Northamptonshire, UK, pages 307–320. Springer.

Di Noia, T., E. Di Sciascio, Donini, F., and Mongiello, M. (2004). *A system for principled Matchmaking in an electronic marketplace*. International Journal of Electronic Commerce, 8(4):9–37.

F.Baader (2003). *Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles*. In Proc. of International Joint Conference on Artificial Intelligence (IJCAI), Acapulco, Mexico, August 9-15, 2003, pages 319–324. Morgan Kaufmann.

Fensel, D. and Bussler, C. (2002). *The web service modeling framework WSMF*. Electronic Commerce Research and Applications, 1(2):113–137.

Frauenfelder, M. (2001). *A Smarter Web*. MIT Technology Rev., 104(9):52–58.

Gomez-Perez, A., Gonzalez-Cabero, R., and Lama, M. (2004). *A framework for design and composition of semantic web services.* In Semantic Web Services- 04, AAAI Spring Symposium Series 22 - 24 March, 2004. AAAI Press / The MIT Press.

Gonzales-Castillo, J., Trastour, D., and Bartolini, C. (2001). Description Logics for Matchmaking of Services. In Proceedings of the KI-2001 Workshop on Applications of Description Logics (ADL-2001),Vienna, Austria, September 18, 2001, volume 44. CEUR Workshop Proceedings.

H. Lausen and D. Roman and U. Keller. (2004). *Web service Modeling Ontology - Standard.Working draft*, Digital Enterprise Research Institute (DERI). http.//www.wsmo.org/2004/d2/v0.2/.

Hacid, M.-S., Leger, A., Rey, C., and Toumani, F. (2002). *Computing concept covers: a preliminary report*. In Proc. of the 15th Intl. Workshop on Description Logics (DL'02), Toulouse, France April 19-21, 2002 volume 53 of CEUR Workshop Proceedings.

Haller, A., Gomez, J. M., and Bussler, C. (2005*). Exposing Semantic Web Service principles in SOA to solve EAI scenarios*. In Proceedings of the workshop "Web Service Semantics: Towards Dynamic Business integration".International Conference on the World Wide Web (WWW2005) ,10-14, 2005, Chiba, Japan.

Horrocks, I., Li, L., Turi, D., and Bechhofer, S. (2004). *The instance store: DL reasoning with large numbers of individuals*. In Proc. of the 2004 Description Logic Workshop (DL 2004), Whistler, British Columbia, Canada, June 6-8, 2004, pages 31–40.

J. Domingue and Cabral L. and Hakimpour H. and Sell D. and Motta E. (2004*). IRS-III: A Platform and Infrastructure for CreatingWSMO-based Semantic Web Services*. . In Proceedings of the Workshop on WSMO Implementations (WIW 2004). 29th and 30th September 2004, Frankfurt Germany. CEUR Workshop Proceedings.

J.Peer (2005). *Web Service Composition as AI Planning - a Survey*. Technical Report. University of St.Gallen, http://elektra.mcm.unisg.ch/pbwsc/docs/pfwsc.pdf.

Klusch, M., Fries, B., Khalid, M., and Sycara, K. (2005). *OWLS-MX: Hybrid Semantic Web Service Retrieval*. In Proceedings of 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web Arlington, Virginia, USA 4th - 6th November, 2005.

Korhonen, J., Pajunen, L., and Puustjarvi, J. (2003). *Automatic Composition of Web Service Workflows Using a Semantic Agent*. In Proc. of IEEE/WIC International Conference on Web Intelligence (WI 2003), 13-17 October 2003, Halifax, Canada, pages 566–569.

L. Li and I. Horrocks (2004). A *software framework for matchmaking based on semantic web technology*. Intl. Journal of Electronic Commerce (IJEC), (4):39–60.

Laukkanen, M. and Helin, H. (2003). *Composing Workflows of Semantic Web Services*. In Proc. of AAMAS Workshop on Web Services and Agent-Based Engineering. Melbourne, Australia; 14--18 July 2003.

Liang, Q., Chakarapani, L. N., Su, S. Y.W., Chikkamagalur, R. N., and Lam, H. (2004*). A Semi-Automatic Approach to Composite Web Services Discovery, Description and Invocation*. International Journal Web Service Research, 1(4):64–89.

Madhavan, J., Bernstein, P., and Rahm, E. (2001). *Generic schema matching with cupid*. In Proc. of International Conference on Very Large Data Bases (VLDB)11-14 Sept. 2001, Rome, pages 49–58. Morgan Kaufmann.

Mecella, M., Parisi Presicce, F., and Pernici, B. (2002). *Modeling e-Service Orchestration Through Petri Nets.* In Proceedings of the 3rd VLDB International Workshop on Technologies for e-Services (VLDB-TES 2002), Hong Kong, China; August 20-23, 2002. pages 38–47. Springer-Verlag LNCS 2444.

Mecella, M. and Pernici, B. (2002). *Building Flexible and Cooperative Applications Based on e-Services*. Technical report. Università di Roma La Sapienza.

Medjahed, B., Bouguettaya, A., and Elmagarmid, A. K. (2003). *Composing Web services on the Semantic Web.* VLDB Journal, 12(4):333–351.

Motta, E., J.Domingue, Cabral, L., and Gaspari (2003). *IRS-II: A Framework and Infrastructure for Semantic Web Services*. In Proceeding of the 2nd International Semantic Web Conference (ISWC) 20-23 October 2003 Sanibel Island, Florida, USA. LNCS, Vol. 2870, 306-318, Springer-Verlag.

Narayanan, S. and S.McIlraith (2002). *Simulation, verification and automated composition of web services*. In Proceedings of the 11th international conference on World Wide Web (WWW), pages 77–88, Honolulu, Hawaii, USA 7-11 May 2002. ACM Press.

Paolucci, M., Kawamura, T., Payne, T., and Sycara, K. (2002). *Semantic Matching of Web Services Capabilities*. In Proc. of First International Semantic Web Conference (ISWC), Sardinia, Italy, 9-12 June 2002, number 2342 in LNCS, pages 333–347. Springer-Verlag.

Patil, A., Oundhakar, S., Sheth, A., and Verma, K. (2004). *Meteor-s web service annotation framework*. In Proc. International World Wide Web Conference (WWW '04) May 17-22 New York, USA, pages 553–562. ACM.

Preist, C. (2004). *A conceptual architecture for semantic web services*. In Proc. of 3rd. International Semantic Web Conference (ISWC), Hiroshima, Japan, November 7-11, volume 3298 of Lecture Notes in Computer Science, pages 395–409. Springer.
Rao, J. and Su., X. (2004). A *survey of automated web service composition methods.* In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004, volume 3387 of Lecture Notes in Computer Science, pages 43–54. Springer.
S. Agarwal and S. Lamparter (2005). sMart - *A Semantic Matchmaking Portal for Electronic Markets*. In In Proceedings of the 7th Int. IEEE Conf. on E-Commerce Technology (CEC'05), July 19-22, 2005, München, Germany, pages 405–408.
S. Grimm and B. Motik and C. Preist (2004). *Variance in e-Business Service Discovery*. In Semantic Web Services: Preparing to Meet the World of Business Applications, workshop at ISWC-2004, 7-11 November 2004, Hiroshima, Japan. volume 119. CEUR Workshop Proceedings.
Sirin, E., Hendler, J., and B.Parsia (2003). *Semi automatic composition of web services using semantic descriptions*. In Proc. of the ICEIS Workshop on Web Services: Modeling, Archit. and Infrastructure, Angers - France - 23-26 April, 2003, pages 17–24.
Sycara, K. P., Paolucci, M., Ankolekar, A., and Srinivasan, N. (2003). *Automated discovery, interaction and composition of Semantic Web services.* Journal of Web Semantics, 1(1):27–46.
Teege, G. (1994). *Making the difference: A subtraction operation for description logics*. In Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94) Bonn, Germany May 24-27, 1994. Morgan Kaufmann.
U. Keller and R. Lara and H. Lausen and A. Polleres and D. Fensel (2005). *Automatic Location of Services.* In The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, pages 1–16. LNCS.

ABOUT THE AUTHORS

Azzurra Ragone received the laurea degree with honors in management engineering from Politecnico di Bari in 2004. She is currently pursuing her PhD degree in Information Engineering at Politecnico di Bari, Italy. Her research interests are in the area of automatic Web services discovery and composition and automated knowledge-based negotiation.

Tommaso Di Noia is an Assistant Professor in Information Technology Engineering at the Technical University of Bari (Politecnico di Bari), Italy. He got his PhD from the Technical University of Bari. His main scientific interests include description logics – theoretical and practical aspects; resource matchmaking; knowledge representation systems for electronic commerce; automatic Web services discovery and composition; knowledge representation systems and applications for the semantic web. He co-authored papers which received the best paper award at conferences ICEC-2004 and CEC-2006, respectively.

Eugenio Di Sciascio received the laurea degree with honors from the University of Bari, and the PhD from the Politecnico di Bari (Technical University of Bari).
He is currently Full Professor of Information Technology Engineering at the Technical University of Bari, Italy, and chairs the Unified Council of Information Engineering degrees class of Technical University of Bari. Formerly, has has been an Assistant Professor at University of Lecce and Associate Professor at the Technical University of Bari. He is the Scientific Coordinator of *SisInfLab*, the Informtion Systems Laboratory of the Technical University of Bari.
His research interests include SWS discovery and composition, multimedia information retrieval, knowledge representation and e-commerce. He is involved in several national and European research projects related to his research interests. He co-authored papers that received best paper awards at the conferences ICEC'04 and IEEE CEC EEE '06.

Francesco M. Donini is a Full Professor at the Università della Tuscia – Viterbo, Italy. Formerly, he has been Assistant Professor at the Università La Sapienza – Roma, and Associate Professor at Politecnico di Bari. His PhD thesis is on description logics, and since his PhD, he has worked on many aspects of knowledge representation in artificial intelligence, both on theoretical issues and on practical applications.

His research interests span from description logics to non-monotonic reasoning, abduction, algorithms and complexity of reasoning in KR formalisms to their application in a variety of practical contexts. He co-authored papers that won awards at conferences IJCAI-1991, ICEC-2004 and CEC-2006.

Simona Colucci received the laurea degree with honors in management engineering in 2003, and the PhD in Information Engineering in 2006, both from Politecnico di Bari. She currently holds a post-doc position at SisInfLab. Her research interests are in the area of knowledge representation and management, Web services discovery and composition and e-commerce. On these topics she has published papers in international journals and conferences.

Francesco Colasuonno received the laurea degree with honors in Computer Science Engineering from Politecnico di Bari in 2006. He is currently a research assistant at SisInfLab, Politecnico di Bari. His research interests include Semantic Web services discovery and composition and knowledge-based e-commerce.