Agent-based semantic composition of Web services using distributed description logics

Mourad Ouziri, Damien Pellier

Agent-based semantic composition of Web services using distributed description logics

Mourad Ouziri and Damien Pellier

LIPADE, Université Paris Descartes
45 rue des Saints Pères, 75006 Paris, France
{mourad.ouziri,damien.pellier}@parisdescartes.fr

Abstract. An important research challenge consists in composing web services in an automatic and distributed manner on a large scale. Indeed, most queries can not be satisfiable by one service and must be processed by composing several services. Each web service is often written by different designers and is described using the terms of their own ontology. Therefore, the composition process needs to deal with a variety of heterogeneous ontologies. In order to tackle this challenge, we propose an approach using Distributed Description Logics (DDL) to achieve the semantic composition of web services. DDL allows one to make semantic connections between ontologies and thus web services, as well as to reason to get a semantic composition of web services.

1 Introduction

The advent of Web services is an inevitable consequence of Web technology and its dissemination on a large scale, poses the problem of their automatic composition. The interoperability of Web services is guaranteed by three key XML-based standards. These standards have been defined to develop and deploy Web services: (1) SOAP (*Simple Object Access Protocol*) defines a communication protocol for Web services; (2) UDDI (*Universal Description Discovery and Integration*) is a registry service allowing the discovery of Web services and (3) WSDL (*Web Services Description Language*) is a language used to describe Web services which provides concepts to describe Web services from a syntactic point of view. Unfortunately, composing Web services requires more than the description of each service. In particular, it must be able to understand the other services and to learn how to interact with them. Thus, the lack of semantic tags in WSDL restricts their interoperability.

The concept of ontology is the key to improve Web services with semantics and interoperability. Ontologies enrich Web services with expressive and computer interpretable languages. They capture the semantics of Web services based on a formal representation of a set of concepts within a domain and the connections between those concepts and them, may be used to reason and compose Web services. Integrating ontologies into Web services could not only enhance the quality and the robustness of service discovery and invocation, but also pave the way for automated composition and seamless interoperation. Unfortunately, guaranteeing the interoperability and the automatic composition of Web services is not enough. This approach assumes that all the concepts are based on the same ontology. In practice, designers of Web services use their own ontologies to describe their services. Therefore, we have to deal with the heterogeneous ontologies. For instance, how can one connect the terms "trip" and "journey" and indicate that they refer to the same concept ? Dealing with a variety of different ontology-based descriptions of web services is still an open challenge.

In order to remove this obstacle, we propose a new approach based on distributed description logics. Distributed description logics is used to establish semantic connections between heterogeneous Web services. This approach has two main advantages:

1. To increase the interoperability between Web services by composing heterogeneous Web services. Our approach makes semantic composition of heterogeneous Web services. Even if the Web services are described using different and heterogeneous ontologies, our approach connects these ontologies using semantic connections between the terms of the ontologies. Then, we can use these connections to infer composable Web services automatically.
2. To reduce the complexity of the composition process by limiting it to only composable Web services. Indeed, traditional composition processes use planning techniques to compose Web services. The complexity of the composition process is limited by the number of services to be composed. This approach allows one to consider semantically composable services only as oppose to all available services.

The rest of the paper is organized as follows: section 2 proposes a synthesis of the related work, section 3 proposes a primary example, section 4 presents an overview of the distributed logic description and finally section 5 introduces our contribution.

2 Related works

Over the previous decade, Web services have been the focus of a lot of research. The published literature concerns automatic discovery [17] and composition of web services [3]. Many appraoches [12] [14] and languages [15], e.g., XLANG (*XML Business Process Language*), BPML (*Business Process Modeling Language*), WSFL (*Web Service Choreography Interface*), etc., were proposed to describe how web services can interact with each other with messages (taking to account the business logic and execution order of the interactions) and track the sequence of messages that may involve multiple parties and multiple sources (including customers, suppliers, and partners). In the rest of this paper, we are focused on the use of description logics [1] for web services discovering and composition:

Web services discovering: Matching is the process of searching the space of possible matches between supply and demand, finding the best available ones. Most of the works using description logics process for matching problems between a service provider and a service requester using standard satisfiability reasoning. Based on CLASSIC [7] structural subsumption algorithm, the best matches finding algorithm is proposed in [9]. The work proposed in [11] deals with the problems which occur in the matchmaking of incomplete service description because of the open-world assumption. In [5], proposed matchmaker architecture performs semantic matching of Web Services on the basis of input and output descriptions of semantic Web Services. In [4] the service discovery is processed as a new instance of the problem of rewriting concepts using terminologies and calls the best covering problem. A hyper-graph-based algorithm to compute the best covers is proposed.

Web services composition: The web service composition problem consists in selecting a finite parallel or sequence of Web services to match a request. In [6], logical reasoning of description logics is used to perform e-Services composition. To do it, authors propose to re-express situation calculus action theories as a description logics knowledge base. In [13], description logics and AI planning are both used to compose services. This work does not deal with heterogeneous service descriptions. That is, the approach can not be composed if the services are described using multiple heterogeneous ontologies. Finally, the work presented in [18] uses description logics only to represent actions, plans and goals and to infer the subsumption connection between actions, plans and goals during plan generation, plan recognition, or plan evaluation. But, this work does not deal with service composition.

3 Desciption Logics Foundation

Description Logics (DL) [1] is a family of logics developed to represent complex hierarchical structures and to make reasoning facilities over these structures. A description logics knowledge base is composed of two parts: abstract knowledge (TBox) and concrete knowledge (ABox). Concrete knowledge ABox represents a set of facts, which are expressed by assertions on individuals of a real world. Abstract knowledge TBox is a set of concept and role descriptions. Concepts are unary predicates and roles are binary predicates. Semantics in DL is given by means of an interpretation function $I = (\Delta^I, .^I)$, where Δ^I is a set which represents the individuals of concrete knowledge and $.^I$ is an interpretation function defined as:

- $.^I(C) = C^I \subseteq \Delta^I$ for each concept C;
- $.^I(R) = R^I \subseteq \Delta^I \times \Delta^I$ for each role R;

Finally, a concept description is expressed using constructors (see [10]) for examples).

Distributed description logics extend standard description logics to create descriptions that link concepts of multiple knowledge bases. Inspired by distributed first order logic [10], Distributed Description Logics (DDL) extends standard description logics as follows [8]:

1. Distributed ABox $DAB = (\{A_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I})$: consists of a set of A-boxes and a set of individual correspondences $r_{ij} \subseteq \Delta_i \times \Delta_j$, where Δ_i and Δ_j are interpretation domains for A_i and A_j respectively.
2. Distributed TBox $DTB = (\{T_i\}_{i \in I}, \{B_{ij}\}_{i \neq j \in I})$: consists of a set of ordinary T-boxes and a set of so-called bridge rules, which express intentional assertions about connections. B_{ij} is a set of directional bridge rules from $KB_i(T_i, A_i)$ to $KB_j(T_j, A_j)$. A bridge rule that connects KB_i to KB_j is an axiom (in KB_j) of the following two forms:
   - Into-rules $i : C \xrightarrow{\sqsubseteq} j : D$, i.e., in the knowledge base KB_j, the concept $j : D$ of KB_j subsumes the imported concept $i : C$ of KB_i. In the rest of the paper, we use the simple syntax $i : C \sqsubseteq j : D$ to express into-rules.

- Onto-rules $i : C \xrightarrow{\sqsupseteq} j : D$, i.e., in the knowledge base KB_j, the concept $j : D$ of KB_j is subsumed by the imported concept $i : C$ of KB_i. In the rest of the paper, we use the simple syntax $i : C \sqsupseteq j : D$ to express onto-rules.

3. Distributed interpretation $DI = (\{I_i\}_{i \in I}, \{r_{ij}\}_{i \neq j \in I})$ consists of a set of ordinary interpretations of DTB T-Boxes and domain relations that interprets bridge rules as follows:
   - Into-rule: $i : C \xrightarrow{\sqsubseteq} j : D$, if $r_{ij}(C^{mj}) \subseteq D^{mj}$
   - Onto-rule: $i : C \xrightarrow{\sqsupseteq} j : D$, if $r_{ij}(C^{mj}) \supseteq D^{mj}$

4 Agent-Based Semantic Composition of Web Services

Composing Web services requires the description of each service so that other services can understand its features. Unfortunately, semantic descriptions of services are not enough to allow automatic communication between services. That is, many terminologies can be used to describe services capabilities. Thus, we need to connect these terminologies to establish semantic and efficient communication between services. Our work focuses on semantic composition of services based on their functional aspects and no on their quality of services.

4.1 Primary example

Let us consider an e-tourism application example where three agents A_1, A_2 and A_3 provide hotel booking service in New-York and Washington, airplane transport service between France and the USA and restaurant service respectively. Suppose a person submits the query: *"I am in Paris and I would like to visit New-York for one week in July. I want to eat in a restaurant."* The three agents A_1, A_2 and A_3 must collaborate to process the query because none of them can solve the request alone. The communications between the agents to compose their services can be illustrated by the following informal dialogue:

A1.1: agent A_1 says: *"I can book a hotel from July 1st to July 7th. But someone else should propose a corresponding trip and restaurant with a complete menu".*
A2.1: agent A_2 says: *"I can offer a flight. But there is no flight available on July 1st. I can offer flights on July 3rd and July 9th".*
A1.2: agent A_1 says: *"OK, I can book a hotel from July 3rd to July 9th".*
A3.1: agent A_3 says: *"I can propose different restaurants with a full menu between July 3rd to July 9th".*

The three agents A_1, A_2 and A_3 use different, incompatible terminologies to communicate. Thus, the above scenario of communication is not successful. That is, in $A1.1$, agent A_1 asks for a trip whereas in $A2.1$ agent A_2 offers a flight. Automatic agents do not make a semantic connection between the terms "trip" and "flight". When agent A_2 receives the request *"I need a trip"* from agent A_1, it replies in $A2.1$ by *"I cannot offer trip"* (as it does not make the semantic connection between "trip" and "flight").

This small dialogue shows that the agents must be connected using semantic connections. We do so in two stages. First, we make a semantic description of the agents,

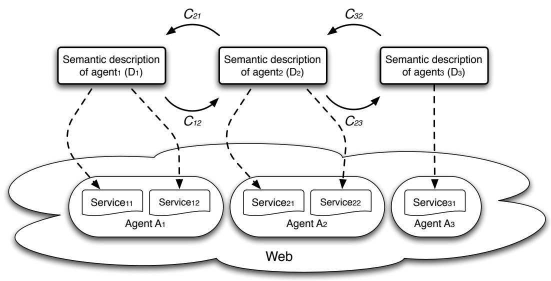

Fig. 1. Ontologies based annotation of Web services

especially of their provided services (each agent is described using a specific terminology). Secondly, we connect the agents between themselves using semantic connections between the agent descriptions. The proposed approach is shown in Fig. 1.

Services provided by agents are described using terms (concepts of ontology). For example, *Hotel*, *Trip*, *Flight*, *Date*, *Restaurant*, *NewYork*, *Menu*, *FullMenu*, etc. are some terms that can be used to describe agents A_1, A_2 and A_3 of the above example. Relations C_{ij} between descriptions (see Fig 1) are semantic connections between the description D_i of the agent A_i and the description D_j of agent A_j from the point of view of A_j. These connections are directional and expressed from a particular agent's viewpoint.

The semantic connections C_{ij} connect terms of descriptions D_i to terms of D_j from the point of view of the agent A_j. This is done by importing terms of D_i in D_j using a set of assertions. In the above example, agents A_1 and A_2 can be connected using the assertion: $2 :$ *Flight* is a sub-concept of $1 :$ *Trip*.

The service description in DL can be automatically generated from WSDL. However, the semantic connections are expressed manually in each peer then they are used by reasoning algorithms to discover automatically all the other implicit connections. We use only subsumption and disjunction relationships to connect concepts of different knowledge bases. Overlapping relationship is not considered because it may be expressed using subsumption and disjunction by refining concepts.

4.2 Service composition model

Given a set of Web services to compose, we propose the Web services composition model described as follows:

Definition 1 (Service description). *A service is described by the tuple $\langle D, P \rangle$ with D is a precise description of the task achieved by the service and P is a set of preconditions required by the service to achieve its task. Both elements are represented in a standard description logics.*

Definition 2 (Distributed directed knowledge base). *A distributed directed knowledge base $dKB < S_1, S_2, C_{12} >$ from service S_1 to service S_2 is defined by adding the* *following axioms : (i) Axioms that define service S_1, (ii) Axioms that define service S_2 and (iii) Axioms C_{12} that connect the terms of service S_1 to those of service S_2.*

Now, we define the concept of composable web services as follows:

Definition 3 (Service Composition Problem). *The service composition model is described by the tuple $\langle S, C\rangle$ where S is a set of services described and C is a set of semantic connections between these services. We use distributed description logics to represent this element.*

Given two services $S_i \langle D_i, P_i\rangle$ and $S_j \langle D_j, P_j\rangle$ in CS. The service S_i is composable with the service S_j, denoted by $S_i \circ S_j$, if the service S_j satisfies (subsumes) the preconditions P_i of S_j. That is, $dKB < S_i, S_j, C_{ij} > \models P_i \sqsubseteq D_j$.

4.3 Problem formalization

Our formalization is based on description logics and their extensions to distributed description logics (see section 3). As shown in figure Fig 1, the system is a collection of inter-related knowledge bases. Each agent is represented by description of provided services and semantic connections with the descriptions of the other agents. Formally, an agent is represented in a standard description logics TBox. Following the example of section 4.1, we describe services provided by agents A_1, A_2 and A_3 as follows:

Service S_1 of A_1:
Hotel $\sqcap$ $\exists$*location.NewYork* $\sqcap$ $\exists$*arrival.Date* $\sqcap$ $\exists$*departure.Date*
Preconditions: $\exists$*in.NewYork* $\sqcap$ (*Trip* $\sqcap$ $\exists$*hasDestination.NewYork* $\sqcap$ ≤ 1 *hasDestination*) *Restaurant* $\sqcap$ $\exists$*hasMenu.Complete*
Effects: HotelReservation

Service S_2 of A_2:
Flight $\sqcap$ $\exists$*departure.Date* $\sqcap$ $\exists$*departureAirport.FrenchAirport* $\sqcap$ $\exists$*arrivalAirport.USAirport*
Preconditions: $\exists$*oldLocation.France* $\sqcap$ ≤ 1 *oldLocation*
Effects: $\exists$*newLocation.France* $\sqcap$ ≤ 1 *newLocation*

Service S_3 of A_3:
Restaurant $\sqcap$ $\exists$*propose.Menu* $\sqcap$ $\exists$*location.City* $\sqcap$ ≤ 1 *location*
Preconditions: Restaurant $\sqsubseteq$ $\exists$*menuDate.Date* $\sqcap$ $\exists$*menuType*.(*Light* $\sqcap$ *Full* $\sqcap$ *Vegetarian*) $\sqcap$ ≤ 1 *menuType*
Effects: $\exists$*toBeIn.NY* $\sqcap$ ≤ 1 *toBeIn*

The query: "*I am in Paris and I will visit New-York in July. I want to eat a complete menu in a restaurant.*" may be represented in description logics as:

– *Hotel* $\sqcap$ $\exists$*location.NewYork* $\sqcap$ $\exists$*arrival.July* $\sqcap$ $\exists$*departure.July* $\quad (1 : q)$
– $\exists$*in.Paris* $\sqcap$ ≤ 1 *in* $\quad (1 : f)$

Using the subsumption reasoning of description logics, we have $1 : q \sqsubseteq S_1$ as *July* $\sqsubseteq$ *Date*. Consequently, agent A_1 is able to execute the query. However, the preconditions of service S_1 implies that a trip is needed. Indeed, $\exists in.Paris \sqcap \leq 1 in \sqcap (\exists in.NewYork \sqcap (Trip \sqcap \exists hasDestination.NewYork \sqcap \leq 1 hasDestination))$ is equivalent to $\exists in.Paris \sqcap \exists in.NewYork \sqcap \leq 1 in$ or $\exists in.Paris \sqcap \leq 1 in \sqcap (Trip \sqcap \exists hasDestination.NewYork \sqcap \leq 1 hasDestination)$. We have $\exists in.Paris \sqcap \exists in.NewYork \sqcap \leq 1 in \sqsubseteq \perp$ as $Paris \sqcap NewYork \sqsubseteq \perp$. Then, $\exists in.Paris \sqcap (Trip \sqcap \exists hasDestination.NewYork \sqcap \leq 1 hasDestination)$ must be satisfied. This means that agent A_1 requires a trip. Agent A_1 must submit the description of the required trip to agents A_2 and A_3. Agent A_2 should be able to satisfy the submitted requirement. However, as agents A_1 and A_2 use heterogeneous terminologies, *Trip* and *Flight* respectively, the reasoning services of description logics do not infer the connection between the requirement of A_1 and the offer of A_2, expressed respectively by the descriptions: $\exists in.Paris \sqcap (Trip \sqcap \exists hasDestination.NewYork \sqcap \leq 1 hasDestination)$ and $Flight \sqcap \exists departure.Date \sqcap \exists departureAirport.FrenchAirport \sqcap \exists arrivalAirport.USAirport$ although terms *Trip* and *Flight*, *in* and *departureAirport*, *hasDestination* and *arrivalAirport* have the same meaning.

The solution we propose consists in connecting agent terminologies using DDL (Distributed Description Logics). Connecting agents A_i to A_j consists in making semantic connections between the preconditions of the connected agent A_i and description of the connecting agent A_j. For our example, we establish a semantic connection between agents A_1 and A_2 using the following distributed assertions added to the knowledge base of agent A_1: (i) $A_1 : Trip \sqsupseteq A_2 : Flight$, (ii) $A_1 : NewYork \sqsubseteq A_2 : USAirport$, (iii) $A_1 : in \sqsupseteq A_2 : departureAirport$ and (iv) $A_1 : hasDestination \sqsupseteq A_2 : arrivalAirport$.

4.4 Distributed composition algorithm

The distributed composition algorithm we propose is based on the distributed satisfiability reasoning proposed in [16]. It is based on *standard tableau algorithms* [2] and uses the message-based communication between local tableau algorithms. The distributed composition algorithm works at two levels: intra-agent level and inter-agent level. At the intra-agent level, the composition algorithm checks whether the agent supports the query. This is done using standard satisfiability reasoning (propagation rules) of description logics. If an agent supports the query, the algorithm verifies whether the facts of the query satisfy the preconditions of the agent. If the agent preconditions are satisfied, the algorithm ends. Otherwise, the algorithm follows at the inter-agents level to search agents that are able to satisfy the preconditions.

The proposed distributed reasoning algorithm, called $DComp_{A_i}(i : Q < i : q, i : f >)$ is based on the distributed satisfiability reasoning $DSat(C)$ proposed in [1]. The automatic composition works as follows:

In: a query $i : Q$ and initial fact $i : f$ expressed over agent A_i.
Out: set of composable services CS.
$DComp_{A_i}(i : Q < i : q, i : f >)$

1. Call $Sat_{A_i}(i : q \sqcap S_i)$ to check whether the query is supported by service S_i of agent A_i. This generates a constraint system (see figure 2), which is a set of assertions: $x : C$ and xRy where x and y are individuals, C is a concept description and R is a role description.

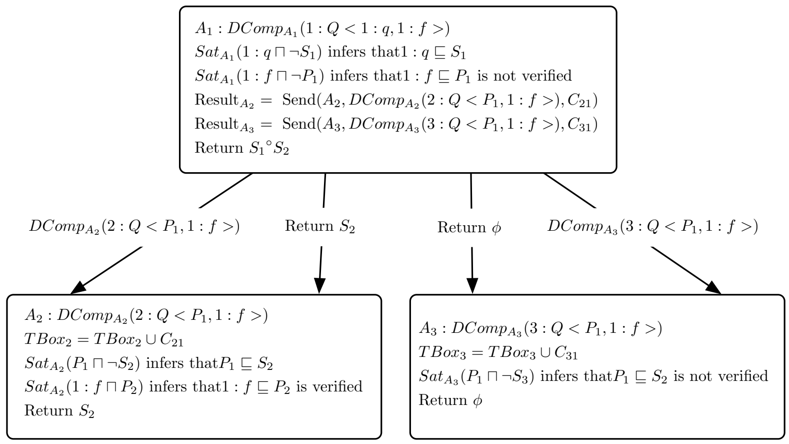

Fig. 2. Beginning of the algorithm viewed by agent A_1

2. If $i : q \sqcap S_i \sqsubseteq \perp$, query is not supported by service S_i. Return $NULL$.
3. Call $Sat_{A_i}(P_i \sqcap \neg i : f)$ to check the subsumption $P_i \sqsubseteq i : f$:
   (a) If $P_i \sqsubseteq i : f$, the preconditions of the query are satisfied by the facts given by the query. Service S_i does not require to be composed, it is able to process on its own query Q. The algorithm terminates and returns $CS = \{S_i\}$.
   (b) Otherwise, service S_i requires to be composed with services that are able to provide preconditions P_i.
4. For each agent A_j connected to A_i by connections C_{ji}, we verify whether A_j satisfies preconditions P_i of A_i:
   (a) Add axioms of C_{ji}.
   (b) Send $CS_j = DComp_{A_j}(j : Q < i : P_i, i : f >)$ to agent A_j with axioms of C_{ji}.
   (c) If $CS_j \neq NULL$ then return $CS = \{S_i{}^\circ CS_j\}$, else return $NULL$.

Let us illustrate this algorithm using the example of section 4.1. The query $1 : Q < 1 : q, 1 : f >$ such that $1 : q = Hotel \sqcap \exists location.NewYork \sqcap \exists arrival.July \sqcap \exists departure.July$ and $1 : f = \exists in.Paris$ is expressed by agent A_1. The algorithm starts with agent A_1 as shown in figure 2.

From figure 3, the query is submitted to agent A_1, which applies standard satisfiability reasoning to decide whether the service provided by agent A_1 is able to process the query. The satisfiability reasoning is performed using propagation rules. Concept $1 : q \sqcap \neg S_1$ is satisfiable because all reasoning possibilities leads to *clash*. Then, query $1 : q$ is subsumed by service S_1.

5 Conclusion

We propose in this paper a formal solution to compose heterogeneous web services. The proposed solution consists in describing services and preconditions provided by agents

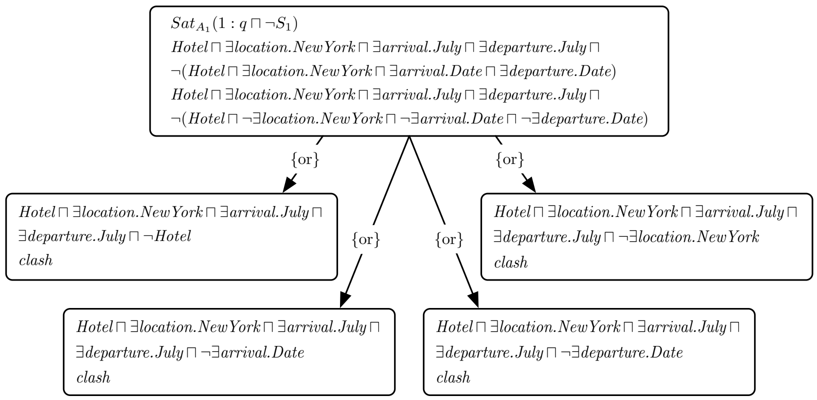

Fig. 3. Satisfiability reasoning and propagation rules

using description logics and making semantic connections between these descriptions. These inter-agents connections are formalized using distributed description logics. We propose a distributed reasoning algorithm that composes web services at a conceptual level with respect to agent connections. This algorithm uses the standard satisfiability algorithm of description logics. The use of distributed description logics allows to make more complete and consistent connections between the agents. That is, logical reasoning uses explicit connections to infer implicit ones since the number of agents to be connected in the semantic Web may be huge. Practicality, approaches based on logics and those based on planning are limited to few agents.

As future works, we plan to propose more reasoning facilities into one main direction. How to propose a complete model that integrates Web services composition at a conceptual level and practical composition at a planning level. Indeed, the Web services description used in our approach is very similar to the planning language such as PDDL (Planning Domain Description Language).

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook. Cambridge University Press (2003)
2. Baader, F., Sattler, U.: An overview of tableau algorithms for desciption logics. Studia Logica 69(1), 5–40 (2001)
3. Benatallah, B., Dumas, M., Sheng, Q.: Facilitating the rapid development and scalable orchestration of composite web services. Journal of Distributed and Parallel Databases 17(1), 5–37 (2005)
4. Benatallah, B., Hacid, M., Léger, A., Rey, C., Toumani, F.: On automating web services discovery. VLDB Journal 14(1), 84–96 (2005)
5. Bener, A., Ozadalia, V., Ilhan, E.: Semantic matchmaker with precondition and effect matching using swrl. Expert Systems with Applications 36(5), 9371–9377 (2009)

6. Berardi, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Mecella, M.: e-service composition by description logics based reasoning. In: Proceedings of the International Workshop on Description Logics. pp. 75–84 (2003)
7. Borgida, A., Patel-Schneide, P.: A semantics and complete algorithm for subsumption in the classic description logic. Journal of Artificial Intelligence Research 1(1), 277–308 (1994)
8. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. Jounal of Data Semantics 1, 153–184 (2003)
9. Colucci, S., Sciascio, T.D., Domini, F., Mongiello, M.: Description logics approach to semantic matching of web services. Journal of Computing and Information Technology 11(3), 217–224 (2003)
10. Ghidini, C., Serafini, L.: Distributed first order logics. In: Frontiers Of Combining Systems 2, Studies in Logic and Computation. pp. 121–140 (1998)
11. Grimm, S., Motik, B., Preist, C.: Matching semantic service descriptions with local closed-world reasoning. In: Proceedings of The Semantic Web: Research and Applications. pp. 575–589 (2006)
12. Levesque, H., Reiter, R., Lespérance, Y., Lin, F., Scherl, R.: GOLOG : a logic programming language for dynamic domains. Journal of Logic Programming 31(1–3), 59–84 (1997)
13. Lin, F., Qiu, L., Huang, H., Yu, Q., Shi, Z.: Description logic based composition of web services. In: Proceedings of the Pacific Rim International Workshop on Agent Computing and Multi-Agent Systems (2006)
14. Peer, J.: A pop-based replanning agent for automatic web service composition. In: Proceedings of the European Conference on Semantic Web. pp. 47–61 (2005)
15. Peltz, C.: Web services orchestration - a review of emerging technologies, tools, and standards. Tech. rep., Hewlett Packard (2003)
16. Serafini, L., Tamilin, A.: Local tableaux for reasoning in distributed description logics. In: Description Logics (2004)
17. Sycara, K., Paolucci, M., Ankolekar, A., Srinivasan, N.: Automated discovery, interaction and composition of semantic web services. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 1(1), 27–46 (2003)
18. Yolanda, G.: Description logics and planning. AI Magazine 26(2), 73–84 (2005)

– **Reviewer:** Why is the contract information represented in WSDL not enough to resolve the interoperability problem among Web services? What are specific examples of the interoperability problem?
Authors: Because WSDL description can be defined by heterogeneous designers, WSDL can use different terms to specify a same concept. For instance, consider the term trip and the term flight that can be used by two planning agents, these two agents cannot plan if they cannot make a semantic connection between these terms. Thus, we need to add a semantic connection layer (see the begin of page 2).

– **Reviewer:** Why does this approach not consider using the ontology mapping to solve the problem of ontology heterogeneity?
Authors: Distributed logic used in our approach is a way to match ontologies (see the sentence of the abstract).

– **Reviewer:** Are heterogeneous ontologies the only problem of realizing semantic composition of Web services?
Authors: No of course, it is possible to use a single ontology to defined agents' capabilities. However, this approach is not realistic because it is hard to design all the concepts of a domain in a single ontology. Thus, in this paper we assume that agents used heterogeneous ontologies (see the begin of page 2).

– **Reviewer:** How does the approach deal with the differences in functional semantics and quality attributes of services?
Authors: Our work focuses on semantic composition of services based on their functional aspects and no on their quality of service. (see page 4)

– **Reviewer:** Will the approach still work when there are a lot of services available? How can the effectiveness of this approach be proved?
Authors: Practicality, approaches based on logics and those based on planning are limited to few agents (see conclusion).

– **Reviewer:** How and by whom can service descriptions in DL be generated?
Authors: The service description in description logics can be automatically generated from WSDL. However, the connection between heterogeneous terms must be done by the service provider (see page 5).

– **Reviewer:** The symbol used in the first formal representation at Section 3 is not visible. Please check the font type used for the symbol.
Authors: done