

Two-phase Web Service Discovery based on Rich Functional Descriptions

Michael Stollberg, Uwe Keller, Holger Lausen, and Stijn Heymans

Digital Enterprise Research Institute Innsbruck (DERI Austria),
Institute for Computer Science, University of Innsbruck,
Technikerstrasse 21a, A-6020 Innsbruck, Austria
Email: {firstname.lastname}@deri.org

Abstract. Discovery is a central reasoning task in service-oriented architectures, concerned with detecting Web services that are usable for solving a given request. This paper presents two extensions in continuation of previous works towards goal-based Web service discovery with sophisticated semantic matchmaking. At first, we distinguish goal templates as generic objective descriptions and goal instances that denote concrete requests as an instantiation of a goal template. Secondly, we formally describe requested and provided functionalities on the level of state transitions that denote executions of Web services, respectively solutions for goals. Upon this, we specify a two-phase discovery procedure along with semantic matchmaking techniques that allow to accurately determine the usability of a Web service. The techniques are defined in the Abstract State Space model that supports several languages for describing Web services.

1 Introduction

Discovery is concerned with detecting usable Web services for solving a given request. This is the first central reasoning task in the context of Semantic Web services, followed by contracting and behavioral conformance tests [17]. Several research works present discovery techniques by semantic matchmaking of requested and provided functionalities, e.g. [16,13,2,7,11]. However, due to deficiencies in the expressiveness and the formal semantics of functional descriptions most existing approaches lack in the achievable quality of the matchmaking results for Web service discovery.

In this respect, we present the advancements towards a goal-based approach for semantically enabled Web service discovery with sophisticated matchmaking. Initially presented in [9], the requester and the provider perspective are separated by formally describing client objectives as goals; a Web service is understood to provide access to several services by its invocation with concrete input values. We extend this approach by differentiating two notions of goals. A *goal template* is a generic objective description that is defined at design time, and a *goal instance* denotes a concrete client request that is created at runtime by instantiating a goal template with concrete input values. Apart from better supporting goal formulation by clients, this allows to realize an efficient two-phase Web service discovery. Usable Web services for goal templates are determined at design time and kept in the system. At runtime, the discovery for goal

instances only needs to investigate those Web services that are usable for the corresponding goal template, so that the number of matchmaking operations necessary at runtime can be reduced. This paper specifies the semantic matchmaking techniques for this framework.

In order to properly describe provided and requested functionalities, we consider a state-based model of the world. Therein, a particular execution of a Web service denotes a sequence of state transitions; such a sequence is also a solution for a goal if the client objective is solved in the end-state. The functionality provided by a Web service is a set of all its possible executions, and a goal template as well as a goal instance describes a set of possible solutions. We formally describe possible executions and solutions with respect to the start- and end-states in *Abstract State Spaces*, a language independent model that defines precise formal semantics for such functional descriptions [10].

On top of this, we specify semantic matchmaking techniques that allow to precisely determine the usability of a Web service for solving a goal. In particular, we (1) revise the definition of previously identified matching degrees and use these to differentiate the usability of a Web service on the goal template level, (2) present a novel approach for semantic matchmaking on the goal instance level, and (3) finally integrate the matchmaking techniques for the goal template and the goal instance level. We specify the techniques in a first-order logic framework and illustrate the definitions by a running example throughout the paper: a goal specifies the objective of finding the best restaurant in a city, and a Web service provides a search facility for the best French restaurant in a city. As we shall discuss, this Web service is only usable for specific goal instances – namely those that specify a city wherein the best restaurant in French.

The paper is structured as follows. Section 2 introduces the concepts of our two-phase discovery approach, and Section 3 defines the formal functional descriptions for Web services and goals. Section 4 specifies the integrated semantic matchmaking techniques for Web service discovery, and Section 5 demonstrates this in the running example. Section 6 discusses related work and positions our approach therein. Finally, Section 7 concludes the paper. A detailed report on this work is provided in [20].

2 Concepts and Approach

The specification of semantic matchmaking techniques for Web service discovery is strongly dependent on the underlying conception and the formal description of Web services and goals. This section introduces the relevant concepts and then outlines the two-phase Web service discovery by discussing the meaning of a match.

2.1 Web Services and Goals

In accordance to the common understanding, we consider a Web service as a computational facility that is invocable over the Internet via an interface [1]. As an abstraction that is sufficient for our purpose, we define a Web service as a pair $W = (IF, \iota)$ such that $IF = (i_1, \dots, i_n)$ is a finite set of names that denotes all inputs required for invoking W , and ι is the implementation of W that is executed when W is invoked.

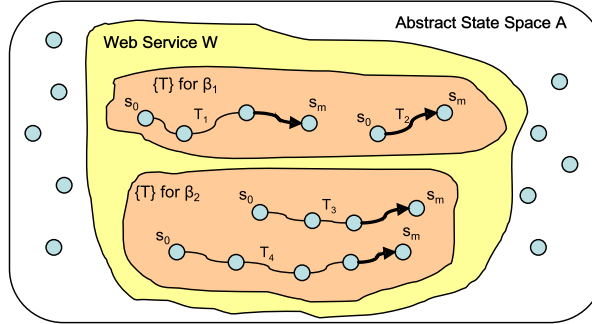


Fig. 1. Web Service, Executions, Input Bindings

In the Abstract State Space model (ASS, [10]), a particular execution of W denotes a finite sequence of state transitions $\tau = (s_0, \dots, s_m)$, i.e. a change of the world from a start state s_0 to an end state s_m . Such a τ is triggered by invoking W with concrete input values; we refer to this as an input binding $\beta : \{i_1, \dots, i_n\} \rightarrow \mathcal{U}$, i.e. a total function that assigns objects of some universe \mathcal{U} to the IF -names. In dependence of the start state, there can be different executions of W for the same input binding. Relevant for the context of discovery, we understand the *overall functionality* provided by W as the set of all its possible executions, denoted by $\{\tau\}_W$. As illustrated in Figure 1, this can be further differentiated into the distinct sets of possible executions of W for each valid input binding, such that $\{\tau\}_W = \bigcup \{\tau\}_{W(\beta)}$ with $W(\beta)$ denoting the set of possible executions of W when invoked with a particular input binding β .¹

Goals in our approach are formally described client objectives. In accordance to related AI research (e.g. [3,15]), we understand a goal as the formal description of the desire of the client to get from the current state of the world into a state wherein the objective is satisfied. This abstracts from technical details irrelevant to the client objective. As promoted by the WSMO framework [12], the overall aim is to enable problem-oriented Web service usage: the client merely specifies the objective to be achieved as a goal, and the system detects and executes suitable Web services for solving this.

We have refined the initial WSMO goal model based on experiences in realizing respective technologies [21]. The extension relevant in the context of discovery is the differentiation of *goal templates* as generic, reusable objective descriptions, and *goal instances* that denote concrete client requests as instantiations of a goal template. Inspired by related system implementations such as IRS [4] and SWF [22], this allows to support goal formulation by client via graphical user interfaces. Instead of requiring the client to specify potentially complex logical formulae for goal formulation, merely pre-defined templates are instantiated with concrete inputs. Figure 2 illustrates this.

¹ We consider the functionalities provided by Web services to satisfy two properties: (1) *deterministic*, i.e. all outputs and effects of an execution are completely dependent on the provided inputs and the start state; non-deterministic functionalities violate the composability of Web services [17]; (2) *non-adaptive*, meaning that in contrast to intelligent software agents a Web service does not itself change the provided functionality [6].

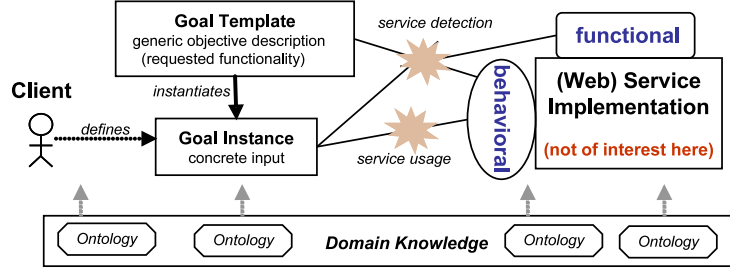


Fig. 2. Goal Templates, Goal Instances, and Web Services

While we shall specify their formal description in the next section, a goal template \mathcal{G} defines generic constraints on the initial state and the desired final state to be achieved. In our restaurant search example, the goal template \mathcal{G} defines that the best restaurant shall be found in a city that is provided as an input by the client. Its meaning in the ASS model is that \mathcal{G} specifies a set of sequences of state transitions $\{\tau\}_{\mathcal{G}}$ as its possible solutions. For each $\tau = (s_0, \dots, s_m) \in \{\tau\}_{\mathcal{G}}$, the start-state s_0 satisfies the constraints on the initial state, and the end-state s_m satisfies the constraints on the desired state of the world. At runtime, a client creates a goal instance $GI(\mathcal{G})$ by defining concrete values for the inputs specified in \mathcal{G} . In the example, this is the concrete city in which the best restaurant shall be found. We refer to this as an input binding β for \mathcal{G} ; this also constitutes the input binding for invoking a Web service to solve $GI(\mathcal{G})$ as discussed above. Because of this instantiation, the possible solutions for $GI(\mathcal{G})$ are a subset of those for \mathcal{G} , so that $\{\tau\}_{GI(\mathcal{G})} \subset \{\tau\}_{\mathcal{G}}$.

2.2 The Meaning of a Match for Web Service Discovery

We now turn towards Web service discovery. With respect to the conception of Web services and goals explained above, the aim is to find a Web service that can provide a τ that is a solution for the goal. Hence, we define the meaning of a match as follows.

Definition 1. Let W be a Web service, \mathcal{G} a goal template, and $GI(\mathcal{G})$ a goal instance that instantiates \mathcal{G} with an input binding β . Let $\tau = (s_0, \dots, s_m)$ be a sequence of states in an Abstract State Space \mathcal{A} . We define the following sets:

$$\begin{aligned} \{\tau\}_{\mathcal{G}} &:= \text{possible solutions for } \mathcal{G} \\ \{\tau\}_W &:= \text{possible executions of } W \\ \{\tau\}_{GI(\mathcal{G})} \subset \{\tau\}_{\mathcal{G}} &:= \text{possible solutions for } GI(\mathcal{G}) \text{ that defines } \beta \\ \{\tau\}_{W(\beta)} \subset \{\tau\}_W &:= \text{possible executions of } W \text{ when invoked with } \beta \end{aligned}$$

We define the usability of a Web service for solving a goal as:

$$\begin{aligned} (i) \text{ match}(\mathcal{G}, W) &: \exists \tau. \tau \in (\{\tau\}_{\mathcal{G}} \cap \{\tau\}_W) \\ (ii) \text{ match}(GI(\mathcal{G}), W) &: \exists \tau. \tau \in (\{\tau\}_{GI(\mathcal{G})} \cap \{\tau\}_{W(\beta)}) \end{aligned}$$

This defines the basic matching conditions for Web Service discovery. Clause (i) states that a Web service W is usable for solving a goal template \mathcal{G} if there exists at

least one execution of W that is a possible solution for \mathcal{G} . Clause (ii) defines that W is usable for solving a goal instance $GI(\mathcal{G})$ if there is at least one execution of W that is also a solution for $GI(\mathcal{G})$ when W is invoked with the inputs defined in $GI(\mathcal{G})$.

Because of $\{\tau\}_{GI(\mathcal{G})} \subset \{\tau\}_{\mathcal{G}}$ it holds that a Web service that is usable for solving a goal instance is also usable for the corresponding goal template. If W can provide a $\tau \in \{\tau\}_{GI(\mathcal{G})}$, then this τ also is also an element of $\{\tau\}_{\mathcal{G}}$. Formally, we can express this as $match(GI(\mathcal{G}), W) \Rightarrow match(\mathcal{G}, W)$. As the logical complement, it also holds that $\neg match(\mathcal{G}, W) \Rightarrow \neg match(GI(\mathcal{G}), W)$, i.e. that a Web service that is not usable for a goal template is also not usable for any of its goal instances.

This constitutes the foundation of our two-phase discovery. Usable Web services for goal templates \mathcal{G} can be determined at design, i.e. when a new goal template is defined. Web service discovery for concrete goal instances $GI(\mathcal{G})$ is performed at runtime. Because of $\neg match(\mathcal{G}, W) \Rightarrow \neg match(GI(\mathcal{G}), W)$, this merely needs to consider the set of Web services that are usable for the corresponding goal template \mathcal{G} . While the achievable efficiency increase is discussed elsewhere [22], this paper specifies the semantic matchmaking techniques for evaluating the matching conditions on the basis of formal descriptions. Without such techniques, we would need to perform test runs of W in order to determine its usability for solving a goal.

3 Formal Functional Descriptions

The following defines functional descriptions for Web services and goals that serve as the basis for semantic matchmaking techniques for Web service discovery. To properly describe requested and provided functionalities on the level of state transitions, we apply functional descriptions as defined in the ASS model mentioned above. This section specifies their structure and formal meaning in a first-order logic framework, and illustrates the definitions in our running example.

3.1 Definition and Semantics

The ASS model describes functionalities in terms of preconditions and effects along with explicitly defining in- and outputs. Focussing on the formal meaning of functional descriptions, they are defined independent of the language used for specifying preconditions and effects. The following recalls the definitions, referring to [10] for details.

An Abstract State Space \mathcal{A} is defined over a signature Σ and some domain knowledge Ω . A functional description is described as a 5-tuple $(\Sigma, \Omega, IF, \phi^{pre}, \phi^{eff})$. The signature Σ differentiates *static symbols* Σ_S that are not changed, *dynamic symbols* Σ_D that are changed by execution of a Web service, and Σ_D^{pre} that denote the interpretation of a dynamic symbol in the start state. Preconditions ϕ^{pre} and effects ϕ^{eff} are defined as statements in a logic $\mathcal{L}(\Sigma)$. $IF = (i_1, \dots, i_n)$ is a set of variables that denote all required inputs. To explicitly specify the deterministic dependency between the start- and end-states with respect to input values, they can occur as the only free variables in ϕ^{pre} and ϕ^{eff} . An input binding $\beta : \{i_1, \dots, i_n\} \rightarrow \mathcal{U}_{\mathcal{A}}$ is a total function that assigns objects of the universe of \mathcal{A} to each IF -variable. Finally, the symbol *out* denotes the computational outputs that are constrained by ϕ^{eff} .

The meaning of a functional description is defined with respect to the start- and the end-state of a sequence of state transitions. Formally, a $\tau = (s_0, \dots, s_m)$ in \mathcal{A} is considered to satisfy the described functionality if and only if it holds that if $s_0 \models_{\mathcal{L}(\Sigma)} \phi^{pre}$ then $s_m \models_{\mathcal{L}(\Sigma)} \phi^{eff}$. Here, $s \models_{\mathcal{L}(\Sigma)} \phi$ expresses that the formula ϕ is satisfied by the universe $\mathcal{U}_{\mathcal{A}}$ in a state s under the logic $\mathcal{L}(\Sigma)$. We refer to this as *implication semantics*: if the precondition is satisfied in s_0 , then s_m will satisfy the effect; otherwise, we can not make any statement about the behavior of the described functionality. Because the *IF*-variables occur as free variables in both the precondition ϕ^{pre} and the effect ϕ^{eff} , the end-state s_m is completely dependent on the start-state s_0 . This reflects the deterministic nature of functionalities provided by Web services.

While functional descriptions in the ASS model are defined independent of the specification language for preconditions and effects, we use classical first-order logic (FOL, [19]) for illustration throughout this work. In order to ease the handling of functional descriptions, we describe them as a first-order logic structure that maintains the formal semantics as defined in the ASS model.

Definition 2. A functional description is a 4-tuple $\mathcal{D} = (\Sigma, \Omega, IF, \phi^{\mathcal{D}})$ such that:

- (i) Σ is a signature consisting of Σ_S (static symbols), Σ_D (dynamic symbols), and Σ_D^{pre} (pre-variants of dynamic symbols)
- (ii) $\Omega \subseteq \mathcal{L}(\Sigma)$ defines consistent domain knowledge
- (iii) IF is a set of variables i_1, \dots, i_n that denote all required input values; an input binding $\beta : \{i_1, \dots, i_n\} \rightarrow \mathcal{U}_{\mathcal{A}}$ is a total function that assigns objects of the universe of \mathcal{A} to each *IF*-variable
- (iv) $\phi^{\mathcal{D}}$ is a FOL formula of the form $[\phi^{pre}]_{\Sigma_D^{pre} \rightarrow \Sigma_D} \Rightarrow \phi^{eff}$ such that
 - ϕ^{pre} is the precondition with *IF* as the only free variables
 - ϕ^{eff} is the effect with *IF* as the only free variables and the outputs are denoted by the predicate *out*
 - $[\phi]_{\Sigma_D^{pre} \rightarrow \Sigma_D}$ is the formula ϕ' derived from ϕ by replacing every dynamic symbol $\alpha \in \Sigma_D$ by its corresponding pre-variant $\alpha_{pre} \in \Sigma_D^{pre}$.

Essentially, $\phi^{\mathcal{D}}$ defines a logical implication between the precondition and the effect formulae. The rewriting function for the precondition handles dynamic symbols. For example, consider a functionality for a bank account withdrawal with $\phi^{pre} : account(a) \wedge balance(a) \geq x$, $\phi^{eff} : account(a) \wedge balance(a) = balance_{pre}(a) - x$, and $\Sigma_D = balance(a)$. We obtain $\phi^{\mathcal{D}} = (account(a) \wedge balance_{pre}(a) \geq x) \Rightarrow (account(a) \wedge balance(a) = balance_{pre}(a) - x)$, so that the relationship between the start- and end-state is specified explicitly. The following specifies the meaning of such a functional description that formally describes the overall functionality provided by a Web service.

Definition 3. Let W be a Web service with $\{\tau\}_W$ as the set of its possible executions in an Abstract State Space \mathcal{A} . Let $\mathcal{D} = (\Sigma, \Omega, IF, \phi^{\mathcal{D}})$ be a functional description. Let $\Omega_{\mathcal{A}} = \Omega \cup [\Omega]_{\Sigma_D^{pre} \rightarrow \Sigma_D}$ be the domain knowledge extended with $\alpha_{pre} \in \Sigma_D^{pre}$.

W provides the functionality described by \mathcal{D} , denoted by $W \models_{\mathcal{A}} \mathcal{D}$, if and only if:

- (i) every Σ -interpretation I with $I \models \Omega_{\mathcal{A}}$ and $I, \beta \models \phi^{\mathcal{D}}$ under every input binding $\beta : IF \rightarrow \mathcal{U}_{\mathcal{A}}$ represents a $\tau \in \{\tau\}_W$, and
- (ii) every $\tau \in \{\tau\}_W$ is represented by a Σ -interpretation I with $I, \beta \models \phi^{\mathcal{D}}$ and $I \models \Omega_{\mathcal{A}}$ under every input binding $\beta : IF \rightarrow \mathcal{U}_{\mathcal{A}}$.

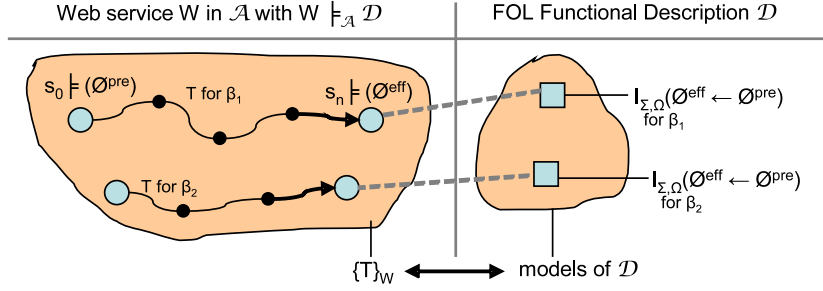


Fig. 3. Illustration of $W \models_{\mathcal{A}} D$

This defines that a Web service W provides the functionality described by D if and only if every Σ -interpretation I, β that is a model of ϕ^D describes a $\tau = (s_0, \dots, s_m) \in \{\tau\}_W$. Such a Σ -interpretation describes the objects that exist in the end-state s_m if W is executed for a particular input binding β in a specific start state s_0 . For the implication semantics from clause (iv) in Definition 2, it holds that $I, \beta \models \phi^D$ if $I, \beta \models \phi^{pre}$ and $I, \beta \models \phi^{eff}$; if $I \not\models \phi^{pre}$, we can not make any statement about the end-state of a τ . Hence, if a $\tau \in \{\tau\}_W$ can be described by a Σ -interpretation I with $I, \beta \models \phi^D$, then it satisfies the described functionality; if there is a $\tau \in \{\tau\}_W$ that cannot be described by such a Σ -interpretation, then W does not provide the described functionality. Figure 3 illustrates this, while we refer to [20] for the formal explanation of this definition and its relationship to the ASS model.

The meaning of a functional description D_G of a goal template \mathcal{G} is analogous. Here, $\{\tau\}_G$ is the set of sequences of state transitions that are solutions for \mathcal{G} such that every $\tau \in \{\tau\}_G$ corresponds to a Σ -interpretation that is a model of D_G . To precisely evaluate the usability of a Web service, in some cases we need to consider the concrete value assignments for the IF -variables. These are provided by the creation of a goal instance $GI(\mathcal{G})$ that defines an input binding β for the IF -variables in D_G of the corresponding goal template \mathcal{G} . Subsequently, this β constitutes the inputs for invoking a Web service in order to solve $GI(\mathcal{G})$. We shall discuss this in more detail in the context of discovery on the goal instance level (Section 4.2).

3.2 Illustration in Running Example

In order to illustrate the above definitions, Table 1 shows the formal functional descriptions of the goal template \mathcal{G} and the Web service W in our restaurant search example.

The goal describes the objective of finding the best restaurant in a city. The specific city is an input required for instantiation. Hence, D_G specifies one IF -variable that is constrained in the precondition ϕ^{pre} to be a *city*. The effect ϕ^{eff} describes the desired state of the world to be given if and only if the received output is a restaurant in the city such that there does not exist any better restaurant in the city. Analogously, D_W describes the functionality provided by the Web service W . The mere difference occurs in the effect: the output of W is a French restaurant in the city that is provided as input such that there does not exist any better French restaurant in the city.

We use classical first-order logic (FOL, [19]) as the specification language. The signature Σ for both \mathcal{D}_G and \mathcal{D}_W defines the respective symbols. Here, $? \langle name \rangle$ denotes a variable. The domain knowledge Ω is defined in the *best restaurant ontology*. This contains axioms specifying that the predicate $better(\cdot, \cdot)$ denotes a partial order, that any restaurant has exactly one type and that the restaurant types *italian* and *french* are distinct from each other, and that restaurants are located in cities. We omit the complete ontology specification due to space limitations. The table shows the functional descriptions with precondition and effects and the corresponding ϕ^D in accordance to Definition 2.

Table 1. Functional Descriptions \mathcal{D}_G , \mathcal{D}_W in Running Example

Goal	Web Service
“find best restaurant in a city” Ω : best restaurant ontology IF : $\{?x\}$ ϕ^{pre} : $city(?x)$ ϕ^{eff} : $\forall ?y. out(?y) \Leftrightarrow ($ $restaurant(?y)$ $\wedge in(?y, ?x)$ $\wedge \neg \exists ?z. (restaurant(?z)$ $\wedge in(?z, ?x)$ $\wedge better(?z, ?y))$.	“provide best French restaurant in a city” Ω : best restaurant ontology IF : $\{?x\}$ ϕ^{pre} : $city(?x)$ ϕ^{eff} : $\forall ?y. out(?y) \Leftrightarrow ($ $restaurant(?y)$ $\wedge in(?y, ?x) \wedge type(?y, french)$ $\wedge \neg \exists ?z. (restaurant(?z)$ $\wedge in(?z, ?x) \wedge type(?z, french)$ $\wedge better(?z, ?y))$.
$\phi^{\mathcal{D}_G}$: $city(?x) \Rightarrow ($ $\forall ?y. out(?y) \Leftrightarrow ($ $restaurant(?y)$ $\wedge in(?y, ?x)$ $\wedge \neg \exists ?z. (restaurant(?z)$ $\wedge in(?z, ?x)$ $\wedge better(?z, ?y))$.	$\phi^{\mathcal{D}_W}$: $city(?x) \Rightarrow ($ $\forall ?y. out(?y) \Leftrightarrow ($ $restaurant(?y)$ $\wedge in(?y, ?x) \wedge type(?y, french)$ $\wedge \neg \exists ?z. (restaurant(?z)$ $\wedge in(?z, ?x) \wedge type(?z, french)$ $\wedge better(?z, ?y))$.

4 Semantic Matchmaking for Web Service Discovery

On the basis of the formal descriptions we now specify the semantic matchmaking techniques for the two-phased Web service discovery introduced in Section 2.2. The aim is to provide semantic means that allow to precisely determine the usability of a Web service with respect to the matching conditions on the goal template and the goal instance level from Definition 1. We therefore define matchmaking on functional descriptions and input bindings as specified above. These provide sufficiently rich descriptions of possible Web service executions and possible solution for goals. The following first specifies semantic matchmaking on the goal template level, then on the goal instance level, and finally integrates the techniques for both levels. We shall demonstrate the techniques in our running example in Section 5.

4.1 Goal Template Level

We express the usability of a Web service W for solving a goal template \mathcal{G} in terms of matching degrees. Adopting the concept and denotation of the degrees from several previous works on Web service discovery (e.g. [16,13,8]), we define them over the functional descriptions of goals and Web services as defined in Section 3.1.

The distinct degrees denote specific relationships between the possible executions $\{\tau\}_W$ of W and possible solutions $\{\tau\}_\mathcal{G}$ for \mathcal{G} . Four degrees – *exact*, *plugin*, *subsume*, *intersect* – denote different situations wherein the matching condition in clause (i) of Definition 1 is satisfied; the *disjoint* degree denotes that this is not given. In our two-phase discovery, these matching degrees serve as a pre-filter for determining the usability of a Web service W for solving a goal instance $GI(\mathcal{G})$ that instantiates the goal template \mathcal{G} . We shall discuss this in more detail in Section 4.3.

We define the criteria for each degree over $\mathcal{D}_\mathcal{G}$ and \mathcal{D}_W from Definition 2, along with an explicit quantification of input bindings β . As the condition for the *exact* degree, $\Omega_{\mathcal{A}} \models \forall\beta. \phi^{\mathcal{D}_\mathcal{G}} \Leftrightarrow \phi^{\mathcal{D}_W}$ defines that every possible execution of W is a solution for \mathcal{G} and vice versa. We assume that all functional descriptions \mathcal{D} are consistent, i.e. that there exists a Σ -interpretation I under a β that is a model of $\phi^{\mathcal{D}}$. Representing a refinement of the matching degree definitions from [8], we therewith obtain a precise means for differentiating the usability of a Web service on the goal template level. Table 2 provides a concise compilation of the matchmaking degree definitions.

Table 2. Definition of Matching Degrees for $\mathcal{D}_\mathcal{G}$, \mathcal{D}_W

Denotation $\mathcal{D}_\mathcal{G} = (\Sigma, \Omega, IF, \phi^{\mathcal{D}_\mathcal{G}})$ $\mathcal{D}_W = (\Sigma, \Omega, IF, \phi^{\mathcal{D}_W})$	Definition $\beta : IF \rightarrow \mathcal{U}_{\mathcal{A}}$ $\phi^{\mathcal{D}} = [\phi^{pre}]_{\Sigma_D^{pre} \rightarrow \Sigma_D} \Rightarrow \phi^{eff}$ $\Omega_{\mathcal{A}} = \Omega \cup [\Omega]_{\Sigma_D^{pre} \rightarrow \Sigma_D}$	Meaning for $\{\tau\}_\mathcal{G}$, $\{\tau\}_W$ with $W \models_{\mathcal{A}} \mathcal{D}_W$
exact ($\mathcal{D}_\mathcal{G}$, \mathcal{D}_W)	$\Omega_{\mathcal{A}} \models \forall\beta. \phi^{\mathcal{D}_\mathcal{G}} \Leftrightarrow \phi^{\mathcal{D}_W}$	if and only if $\tau \in \{\tau\}_\mathcal{G}$ then $\tau \in \{\tau\}_W$
plugin ($\mathcal{D}_\mathcal{G}$, \mathcal{D}_W)	$\Omega_{\mathcal{A}} \models \forall\beta. \phi^{\mathcal{D}_\mathcal{G}} \Rightarrow \phi^{\mathcal{D}_W}$	if $\tau \in \{\tau\}_\mathcal{G}$ then $\tau \in \{\tau\}_W$
subsume ($\mathcal{D}_\mathcal{G}$, \mathcal{D}_W)	$\Omega_{\mathcal{A}} \models \forall\beta. \phi^{\mathcal{D}_\mathcal{G}} \Leftarrow \phi^{\mathcal{D}_W}$	if $\tau \in \{\tau\}_W$ then $\tau \in \{\tau\}_\mathcal{G}$
intersect ($\mathcal{D}_\mathcal{G}$, \mathcal{D}_W)	$\Omega_{\mathcal{A}} \models \exists\beta. \phi^{\mathcal{D}_\mathcal{G}} \wedge \phi^{\mathcal{D}_W}$	there is a τ such that $\tau \in \{\tau\}_\mathcal{G}$ and $\tau \in \{\tau\}_W$
disjoint ($\mathcal{D}_\mathcal{G}$, \mathcal{D}_W)	$\Omega_{\mathcal{A}} \models \neg\exists\beta. \phi^{\mathcal{D}_\mathcal{G}} \wedge \phi^{\mathcal{D}_W}$	there is no τ such that $\tau \in \{\tau\}_\mathcal{G}$ and $\tau \in \{\tau\}_W$

4.2 Goal Instance Level

A goal instance $GI(\mathcal{G})$ is created by defining an input binding β for the *IF*-variables in the functional description $\mathcal{D}_\mathcal{G}$ of the corresponding goal template \mathcal{G} . Recalling from Definition 1, a match on the goal instance level is given if there exists a $\tau = (s_0, \dots, s_m)$ in \mathcal{A} that is a solution for $GI(\mathcal{G})$ and can be provided by a Web service W when it is

invoked with the concrete input values defined in $GI(\mathcal{G})$. The following specifies a general technique for determining this on the basis of the available descriptions, independent of the matching degree between \mathcal{D}_G and \mathcal{D}_W .

Formally, an input binding $\beta : \{i_1, \dots, i_n\} \rightarrow \mathcal{U}_A$ is a total function that defines a variable assignment over the universe \mathcal{U}_A for the input variables IF defined in a functional description \mathcal{D} (cf. Definition 2). We therewith obtain an assignment of concrete values v for all inputs required in \mathcal{D} , i.e. $\beta = \{i_1|v_1, \dots, i_n|v_n\}$. Given such a β , we can instantiate \mathcal{D} by substituting all IF -variables that occur as free variables in ϕ^{pre} and ϕ^{eff} by the concrete values defined in β . We obtain $[\mathcal{D}]_\beta$ as the functional description that is instantiated for the context of β ; this can be evaluated because it does no longer contain any free variables. By instantiating the functional descriptions \mathcal{D}_G of the corresponding goal template \mathcal{G} and \mathcal{D}_W of the Web service W with the input binding β defined in $GI(\mathcal{G})$, we obtain $[\mathcal{D}_G]_\beta$ as the functionality requested by $GI(\mathcal{G})$ and $[\mathcal{D}_W]_\beta$ as the functionality that can be provided by W when it is invoked with β .

For W to be usable for solving $GI(\mathcal{G})$, there must be a τ such that $\tau \in \{\tau\}_{GI(\mathcal{G})}$ and $\tau \in \{\tau\}_{W(\beta)}$ (cf. clause (ii) from Definition 1). To determine this on the basis of the given descriptions, it must hold that – with respect to the domain knowledge – there exists a Σ -interpretation I that is a common model for $\phi^{\mathcal{D}_G}$ and $\phi^{\mathcal{D}_W}$ when both functional descriptions are instantiated with the input binding β defined in $GI(\mathcal{G})$. Formally, this means that the union of the formulae $\Omega_A \cup \{[\phi^{\mathcal{D}_G}]_\beta, [\phi^{\mathcal{D}_W}]_\beta\}$ must be satisfiable, i.e. that there exists a Σ -interpretation that is a model for the extended domain knowledge Ω_A and for the instantiated goal description $[\phi^{\mathcal{D}_G}]_\beta$ and for the instantiated Web service description $[\phi^{\mathcal{D}_W}]_\beta$. In accordance to Definition 3, this I represents a τ that is a solution for $GI(\mathcal{G})$ and can be provided by W if it is invoked with β .

Definition 4. Let $\mathcal{D}_G = (\Sigma, \Omega, IF_G, \phi^{\mathcal{D}_G})$ be a functional description of a goal template \mathcal{G} . Let $GI(\mathcal{G})$ be a goal instance that instantiates \mathcal{G} with the input binding $\beta : IF_G \rightarrow \mathcal{U}_A$. Let $\mathcal{D}_W = (\Sigma, \Omega, IF_W, \phi^{\mathcal{D}_W})$ be a functional description, and let $W = (IF, \iota)$ be a Web service with $W \models_A \mathcal{D}_W$.

$match(GI(\mathcal{G}), W)$ is given if there exists a Σ -interpretation I such that:

$$I \models \Omega_A \quad \text{and} \quad I \models [\phi^{\mathcal{D}_G}]_\beta \quad \text{and} \quad I \models [\phi^{\mathcal{D}_W}]_\beta.$$

Another requirement for W to be usable for solving $GI(\mathcal{G})$ is that the β defined in $GI(\mathcal{G})$ provides concrete values for all inputs that are required to invoke W . This is given if there is a bijection $\pi : IF_{\mathcal{D}_G} \rightarrow IF_{\mathcal{D}_W}$ such that for every input variable in \mathcal{D}_W there is a corresponding input variable in \mathcal{D}_G , and each $i \in IF_{\mathcal{D}_G}$ is assigned with the concrete value from β . Subsequently, if there is a second bijection $\pi_2 : IF_{\mathcal{D}_W} \rightarrow IF_W$ such that for each input name required by W there is a corresponding input variable in \mathcal{D}_W , then there is a concrete value assignment for each input required by W .²

² We are aware of that this requirement is not trivial to realize in practice, as it requires a semantic mapping between the input variables of functional descriptions and the Web service. Moreover, this may require mediation between incompatible ontologies used by the requester and provider [5]. However, to invoke a Web service there must be concrete values for all required inputs – the two bijections denote the basic requirement therefore. [20] discusses ways to weaken the requirements for the necessary compatibility, e.g. by creating existentially quantified ontology instances for input values that are not explicitly defined by the client.

4.3 Integration of Matchmaking Techniques

We complete this section with combing the semantic matchmaking techniques for the goal template and the goal instance level in order to attain an integrated matchmaking framework for our two-phase Web service discovery. We therefore extend matchmaking degrees from Table 2 with the matchmaking condition for the goal instance level. Due to their definition, we can simplify the matching condition from Definition 4 for the distinct matchmaking degrees as follows.

Theorem 1. *Let \mathcal{D}_G describe the requested functionality in a goal template \mathcal{G} . Let $GI(\mathcal{G})$ be a goal instance of \mathcal{G} that defines an input binding β . Let W be a Web service, and let \mathcal{D}_W be a functional description such that $W \models_A \mathcal{D}_W$.*

W is usable for solving $GI(\mathcal{G})$ if and only if:

- (i) *exact*($\mathcal{D}_G, \mathcal{D}_W$) *or*
- (ii) *plugin*($\mathcal{D}_G, \mathcal{D}_W$) *or*
- (iii) *subsume*($\mathcal{D}_G, \mathcal{D}_W$) *and* $\bigwedge \Omega_A \wedge [\phi^{\mathcal{D}_W}]_\beta$ *is satisfiable, or*
- (iv) *intersect*($\mathcal{D}_G, \mathcal{D}_W$) *and* $\bigwedge \Omega_A \wedge [\phi^{\mathcal{D}_G}]_\beta \wedge [\phi^{\mathcal{D}_W}]_\beta$ *is satisfiable.*

This specifies the minimal matchmaking conditions for determining the usability of a Web service for solving a concrete client request that is described by a goal instance. Under both the *exact* and the *plugin* degree, W can be used for solving any goal instance $GI(\mathcal{G})$ because $\{\tau\}_{GI(\mathcal{G})} \subset \{\tau\}_{\mathcal{G}} \subseteq \{\tau\}_W$ and $\tau \in \{\tau\}_{GI(\mathcal{G})} \Leftrightarrow \tau \in \{\tau\}_{W(\beta)}$. Under the *subsume* degree it holds that $\{\tau\}_{\mathcal{G}} \supseteq \{\tau\}_W$, i.e. every execution of W can solve \mathcal{G} but there can be solutions of \mathcal{G} that cannot be provided by W . Hence, W is only usable for solving $GI(\mathcal{G})$ if the input binding β defined in $GI(\mathcal{G})$ allows to invoke W . This is given if there is a Σ -interpretation that is a model for $[\phi^{\mathcal{D}_W}]_\beta$ and the conjunction of the axioms in Ω_A . Under *intersect* as the weakest degree, the complete matchmaking condition for the goal instance level must hold because there can be solutions for \mathcal{G} that can not be provided by W and vice versa. The *disjoint* degree denotes that W is not usable for solving the goal template and thus neither for any of its instantiations. We refer to [20] for the formal proof of this theorem.

5 Evaluation

In order to demonstrate the precision for Web service discovery that is achievable with the presented matchmaking techniques, this section discusses them for our restaurant search example. We have implemented and verified the matchmaking techniques in VAMPIRE [18], a resolution-based theorem prover for classical first-order logic with equality that allows to realize matchmaking exactly as we have specified above. Due to space limitations, we here content ourselves with condensed explanations on the matchmaking techniques for the goal and the Web service as introduced in Section 3.2. A more detailed documentation as well as further examples for discovery under other matchmaking degrees is provided in [20].³

³ The VAMPIRE implementation along with installation instructions and the proof obligations for the best restaurant search example are available at: <http://members.deri.at/~michaels/software/best-restaurant-example.zip>

The following discusses the matchmaking techniques for the goal of finding the best restaurant in a city and a Web service that provides the best French restaurant in a city (*cf.* functional descriptions in Table 1). This is an example for the *intersect* degree and hence requires the full range of the extended matchmaking for the goal instance level.

For illustration, it is sufficient to consider city A wherein the best restaurant is French and city B wherein the best restaurant is not French. We define two input bindings, $\beta_1 = \{?x|A\}$ and $\beta_2 = \{?x|B\}$, and examine the solutions for \mathcal{G} and the executions of W for each. Table 3 provides a concise overview of the information relevant for our discussion. The first part shows the description of the three best restaurants in A and B as background ontologies $\Omega_1, \Omega_2 \subseteq \Omega$. The second part shows the goal instances, i.e. when $\mathcal{D}_{\mathcal{G}}$ is instantiated with the concrete values defined in the distinct β as explained in Section 4.2. Analogously, the third part shows the only possible instantiations for W . Finally, the fourth part identifies common Σ -interpretations that serve as a witness for a semantic match between the goal instances and the described Web Services.

Table 3. Relevant Information for Matchmaking Illustration

City A: $\Omega_1 \subseteq \Omega$	City B: $\Omega_2 \subseteq \Omega$
$\Omega_1 = \{city(A)$ $restaurant(r1A)$ $in(r1A, A), type(r1A, french)$ $restaurant(r2A)$ $in(r2A, A), type(r2A, italian)$ $restaurant(r3A)$ $in(r3A, A), type(r3A, french)$ $better(r1A, r2A)$ $better(r2A, r3A)\}$	$\Omega_2 = \{city(B)$ $restaurant(r1B)$ $in(r1B, B), type(r1B, italian)$ $restaurant(r2B)$ $in(r2B, B), type(r2B, french)$ $restaurant(r3B)$ $in(r3B, B), type(r3B, french)$ $better(r1B, r2B)$ $better(r2B, r3B)\}$
$[\phi^{\mathcal{D}_{\mathcal{G}}}]_{\beta_1}$ with $\beta_1 = \{x A\}$	$[\phi^{\mathcal{D}_{\mathcal{G}}}]_{\beta_2}$ with $\beta_2 = \{x B\}$
$city(A) \Rightarrow ($ $\forall ?y.(out(?y) \Leftrightarrow ($ $restaurant(?y) \wedge in(?y, A)$ $\wedge \neg \exists ?z.(restaurant(?z)$ $\wedge in(?z, A)$ $\wedge better(?z, ?y))))$	$city(B) \Rightarrow ($ $\forall ?y.(out(?y) \Leftrightarrow ($ $restaurant(?y) \wedge in(?y, B)$ $\wedge \neg \exists ?z.(restaurant(?z)$ $\wedge in(?z, B)$ $\wedge better(?z, ?y))))$
$[\phi^{\mathcal{D}_W}]_{\beta_1}$ with $\beta_1 = \{x A\}$	$[\phi^{\mathcal{D}_W}]_{\beta_2}$ with $\beta_2 = \{x B\}$
$city(A) \Rightarrow ($ $\forall ?y.(out(?y) \Leftrightarrow ($ $restaurant(?y)$ $\wedge in(?y, A) \wedge type(?y, french)$ $\wedge \neg \exists ?z.(restaurant(?z)$ $\wedge in(?z, A) \wedge type(?z, french)$ $\wedge better(?z, ?y))))$	$city(B) \Rightarrow ($ $\forall ?y.(out(?y) \Leftrightarrow ($ $restaurant(?y)$ $\wedge in(?y, B) \wedge type(?y, french)$ $\wedge \neg \exists ?z.(restaurant(?z)$ $\wedge in(?z, B) \wedge type(?z, french)$ $\wedge better(?z, ?y))))$
I_1 with $I_1 \models \Omega \cup \{[\phi^{\mathcal{D}_{\mathcal{G}}}]_{\beta_1}, [\phi^{\mathcal{D}_W}]_{\beta_1}\}$	I_2 with $I_2 \models \Omega \cup \{[\phi^{\mathcal{D}_{\mathcal{G}}}]_{\beta_2}, [\phi^{\mathcal{D}_W}]_{\beta_2}\}$
$\Omega_1 \cup \Omega_2 \cup \{out(r1A),$ $better(r1A, r3A), better(r1B, r3B)\}$	No such I_2 can exist!

We can observe that for the input binding β_1 , there is a Σ -interpretation I_1 that is consistent with the background ontology Ω and satisfies both the instantiation of the goal template $[\phi^{\mathcal{D}_G}]_{\beta_1}$ as well as the instantiation of the Web service $[\phi^{\mathcal{D}_W}]_{\beta_1}$. The witnessing execution τ corresponds to the pair (I_1, β_1) . Hence, the condition for the *intersect* match is satisfied (cf. Table 2). Furthermore, we observe that for the input binding β_2 there can not exist such a common interpretation. Hence, neither the condition for the *subsumes* nor for the *plugin* is satisfied; thus also not the one for the *exact* degree. Assume that there would be such a common interpretation I_2 , i.e. a Σ -interpretation that satisfies Ω , $[\phi^{\mathcal{D}_G}]_{\beta_2}$ and $[\phi^{\mathcal{D}_W}]_{\beta_2}$. From the second column of Table 3 we can conclude that any object $?y$ that is the best restaurant in city B is a french restaurant. However, this is not consistent with the background ontology Ω as described above, since then restaurant $r1B$ must be at the same time an italian as well as a french restaurant.

Because of the *intersect* degree on the goal template level, clause (iv) of Theorem 1 must hold for W to be usable for solving a goal instance $GI(\mathcal{G})$ that instantiates \mathcal{G} . This requires that there must be a Σ -interpretation that is (a) consistent with the background ontology Ω and (b) a common model for $[\phi^{\mathcal{D}_G}]_{\beta}$ and $[\phi^{\mathcal{D}_W}]_{\beta}$ (cf. Definition 4). Let us consider $GI(\mathcal{G})_1$ as the goal instance that instantiates \mathcal{G} with β_1 , and $GI(\mathcal{G})_2$ as the goal instance that defines β_2 . Analyzing the possible solutions and executions in Table 3 reveals the intuitively expected discovery results: the Σ -interpretation I_1 serves as a witness for a $\tau \in \{\tau\}_{GI(\mathcal{G})_1}$ and $\tau \in \{\tau\}_{W_{\beta_1}}$. Hence, W is usable for solving $GI(\mathcal{G})_1$. On the other hand, as discussed above, there can not exist such a witness for $GI(\mathcal{G})_2$; therefore W can not be used to solve $GI(\mathcal{G})_2$.

6 Related Work

Due to its relevance for service-oriented architectures, Web service discovery is subject to several research efforts. We here discuss directly related works with respect to the quality of matchmaking techniques and the modelling client objectives, referring to more exhaustive overviews, e.g. in [9,10,20].

As early works, [16] presents matchmaking of in- and outputs in OWL-S, and [13] defines matchmaking of requested and provided results in a DL framework. Both define the matching degrees in terms of concept subsumption, and work on OWL-S service advertisements and requests described by inputs, outputs, preconditions, and effects [14]. Although using OWL as an expressive specification language, this description neither explicates the dependency pre- and post execution descriptions nor defines formal semantics for functional descriptions. Hence, the matchmaking algorithms merely allow to detect ontological relationships between corresponding description elements – but not to determine whether the invocation of a Web service in a particular state of the world will satisfy a client request. We can observe the same deficiencies in [2].

In WSMO, provided and requested capabilities are described by preconditions, assumptions, postconditions, and effects, along with *shared variables* to define dependencies between the formulae [12]. However, no formal semantics are defined for these complex functional descriptions – which hampers the specification of accurate matchmaking mechanisms. Our functional descriptions overcome this by explicitly describing dependency of preconditions and effects and defining precise formal semantics. [7]

presents a recent approach with a similar focus. Functionalities are described by inputs, outputs, and the relationship between them; a match is given if the requester can provide the input required by the Web service, and the Web service then can provide outputs that satisfy the ones requested. However, this approach is restricted to stateless Web services and hence only covers a subset of the functionalities supported by our approach.

WSMO is the only framework that promotes a goal-based approach for Semantic Web services; most other approaches model client requests as queries for specific Web service descriptions. The differentiation of goal templates and goal instances is a refinement of the WSMO goal model based on experiences in technology realization [21]. A similar two-phased discovery approach is presented in [11]. However, therein goals are described by the desired final state only; the input binding for invoking the discovered Web service is created at runtime. In contrast, we describe the requested functionality in goal templates by preconditions and effects. The reason is that in service-oriented architectures usually the current state of the world is not explicated or is not accessible to the interaction partners. Moreover, defining input bindings on the level of goal instances allows to minimize the client-system interaction as it just needs to be done once.

7 Conclusions

This paper has presented the integrated semantic matchmaking for a two-phased Web service discovery that distinguishes goal templates and goal instances. Continuing previous work, we have defined matchmaking techniques that work on sufficiently rich functional descriptions and can precisely determine the usability of a Web service.

To formally describe client requests on the problem layer, we distinguish goal templates as generic objective descriptions and goal instances that denote a concrete client request as the instantiation of a goal template. We use functional descriptions that precisely describe the start- and end-states of possible executions of Web services as well as of possible solutions for goals. A match is given if a Web service can provide an execution that is a solution for the goal. We have specified semantic matchmaking techniques to evaluate this. On the goal template level, we define matching degrees that differentiate the relationship between possible executions of a Web service and solutions. For a goal instance, a Web service is usable if its execution triggered by the invocation with the concrete inputs is a solution for the instantiated goal description. We therefore have presented a novel matchmaking technique and formally integrated this with the matching degrees on the goal template level. Finally, we have demonstrated that the matchmaking techniques allow to precisely determine the usability of a Web service for solving a concrete client request that is described as a goal instance.

The presented techniques denote the formal foundations for semantic matchmaking in this two-phased discovery approach. We plan to extend this with techniques for efficient management of discovery results, and to continue the integration into frameworks and system implementations for Semantic Web services.

Acknowledgments. This material is based upon works supported by the EU under the **DIP** project (FP6 - 507483) and by the Austrian Federal Ministry for Transport, Innovation, and Technology under the project **RW²** (FFG 809250). The authors like to thank Martin Hepp and Rubén Lara for constructive discussions on the presented work.

References

1. G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer, Berlin, Heidelberg, 2004.
2. B. Benatallah, M.-S. Hacid, A. Leger, C. Rey, and F. Toumani. On Automating Web Services Discovery. *VLDB Journal*, 14(1):84–96, 2005.
3. M. E. Bratman. *Intention, Plans and Practical Reason*. Harvard University Press, Cambridge, MA (USA), 1987.
4. L. Cabral, J. Domingue, S. Galizia, A. Gugliotta, B. Norton, V. Tanasescu, and C. Pedrinaci. IRS-III – A Broker for Semantic Web Services based Applications. In *Proc. of the 5th International Semantic Web Conference (ISWC 2006), Athens(GA), USA*, 2006.
5. E. Cimpian, A. Mocan, and M. Stollberg. Mediation Enabled SemanticWeb Services Usage. In *Proc. of the 1st Asian Semantic Web Conference (ASWC 2006), Beijing, China*, 2006.
6. I. Dickinson and M. Wooldridge. Agents are not (just) Web Services: Considering BDI Agents and Web Services. In *Proc. of the 2005 Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005), Utrecht, The Netherlands*, 2005.
7. D. Hull, E. Zolin, A. Bovykin, I. Horrocks, U. Sattler, and R. Stevens. Deciding Semantic Matching of Stateless Services. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI'2006)*, 2006.
8. U. Keller, R. Lara, H. Lausen, and D. Fensel. Semantic Web Service Discovery in the WSMO Framework. In J. Cardoses, editor, *Semantic Web: Theory, Tools and Applications*. Idea Publishing Group, 2006.
9. U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel. Automatic Location of Services. In *Proc. of the 2nd European Semantic Web Conference (ESWC 2005), Crete, Greece*, 2005.
10. U. Keller, H. Lausen, and M. Stollberg. On the Semantics of Funtional Descriptions of Web Services. In *Proc. of the 3rd European Semantic Web Conference (ESWC 2006), Montenegro*, 2006.
11. R. Lara. Two-phased Web Service Discovery. In *Proc. of AI-Driven Technologies for Services-Oriented Computing Workshop at AAAI-06, Boston, USA*, 2006.
12. H. Lausen, A. Polleres, and D. Roman (eds.). Web Service Modeling Ontology (WSMO). W3C Member Submission 3 June, 2005.
13. L. Li and I. Horrocks. A Software Framework for Matchmaking based on Semantic Web Technology. In *Proc. of the 12th World Wide Web Conference, Budapest, Hungary*, 2003.
14. D. Martin. OWL-S: Semantic Markup for Web Services. W3C Member Submission 22 November, 2004. online: <http://www.w3.org/Submission/OWL-S/>.
15. A. Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA (USA), 1990.
16. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *Proc. of the First International Semantic Web Conference, Springer*, 2002.
17. C. Preist. A Conceptual Architecture for Semantic Web Services. In *Proc. of the 3rd International Semantic Web Conference (ISWC 2004), Hiroshima, Japan*, 2004.
18. A. Riazanov and A. Voronkov. The Design and Implementation of VAMPIRE. *AI Communications*, 15(2):91–110, 2002. Special Issue on CASC.
19. R. M. Smullyan. *First Order Logic*. Springer, 1968.
20. M. Stollberg and U. Keller. Semantic Web Service Discovery. Technical report, DERI, 2006.
21. M. Stollberg and B. Norton. A Refined Goal Model for Semantic Web Services. Proc. of the 2nd International Conference on Internet and Web Applications and Services (ICIW 2007), Mauritius, 2007.
22. M. Stollberg, D. Roman, I. Toma, U. Keller, R. Herzog, P. Zugmann, and D. Fensel. Semantic Web Fred – Automated Goal Resolution on the Semantic Web. In *Proc. of the 38th Hawaii International Conference on System Science (HICSS-38)*, 2005.