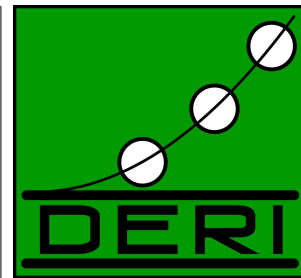


DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE



SEMANTIC WEB SERVICE DISCOVERY

Michael Stollberg Uwe Keller

DERI TECHNICAL REPORT 2006-10-20
OCTOBER 2006

DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE

DERI Galway
University Road
Galway, Ireland
www.deri.ie

DERI Innsbruck
Technikerstrasse 21a
Innsbruck, Austria
www.deri.at

DERI Korea
Yeonggun-Dong, Chongno-Gu
Seoul, Korea
korea.deri.org

DERI Stanford
Serra Mall
Stanford, USA
www.deri.us

MATCHMAKING FOR GOAL TEMPLATES AND GOAL INSTANCES ON RICH FUNCTIONAL DESCRIPTIONS

Michael Stollberg¹ Uwe Keller¹

Abstract. As a central reasoning task in service-oriented architectures, discovery is concerned with detecting Web services that are usable for solving a given request. The emerging concept of Semantic Web services strives towards automation of the complete Web service usage process. Existing approaches for Web service discovery in this field work on confined descriptions of Web services and requests. This limits the achievable accuracy of discovery results. Exploiting the full potential of semantically enabled Web service discovery, this paper presents a discovery framework that works on sufficiently rich descriptions of the functionality provided by Web services and requested by clients in a state-based model of the world. We differentiate two elements for formally specifying requests for Web services: goal templates as generic objective descriptions and goal instances that denote concrete requests by instantiating a goal template. Upon this, we specify a two-step discovery procedure along with semantic matchmaking techniques that allow to accurately determine the usability of a Web service. The presented framework is defined in a language independent manner so that it is applicable to several languages for semantically describing Web services.

Keywords: Discovery, Goal Templates, Goal Instances, Abstract State Spaces, Functional Descriptions, Matchmaking

¹Digital Enterprise Research Institute (DERI) Innsbruck, University of Innsbruck, Technikerstraße 21a, A-6020 Innsbruck, Austria. eMail: {michael.stollberg, uwe.keller}@deri.org.

Acknowledgements: 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 **RW**² (FFG 809250).

The authors like to thank Holger Lausen and Stijn Heymans for constructive feedback and discussion on this work.

Copyright © 2006 by the authors

Contents

1	Introduction	1
2	Concepts and Approach	2
2.1	Web Services and Goals	2
2.2	The Meaning of a Match in Web Service Discovery	3
3	Formal Functional Descriptions	6
3.1	Definition and Semantics	6
3.2	Simulation as Logical Formulae	7
4	Semantic Matchmaking	11
4.1	Goal Template Level Discovery	11
4.2	Goal Instance Level Discovery	12
4.2.1	Requirements Analysis	12
4.2.2	Input Bindings	13
4.2.3	Matchmaking For Goal Instances	14
4.3	Integration of Matchmaking Techniques	15
5	Illustrative Example	16
5.1	Goal and Web Service Description	16
5.2	Semantic Web Service Discovery	17
5.2.1	Goal Template Level	19
5.2.2	Goal Instance Level	20
5.3	Web Service Discovery for Goal Instances under other Matching Degrees	21
6	Related Work	22
7	Conclusions	24
A	Appendix	27
A.1	Overview of Matchmaking Degrees For $\mathcal{D}_G, \mathcal{D}_W$	27
A.2	Proof for Theorem 1: Simulation of Functional Description as FOL Structure	28
A.3	Proof for Theorem 2: Integrated Matchmaking for 2-Phased Web Service Discovery	29

1 Introduction

Web services are considered as the base technology for service-oriented architectures (SOA). Providing access to computational facilities over the Internet, the aim is to dynamically detect and execute appropriate Web services for individual client requests [7]. The initial technology stack around WSDL, SOAP, and UDDI limits the usage of Web services to manual inspection, hence fails to provide sufficient support for realizing the SOA vision. The emerging concept of Semantic Web services aims at overcoming this: working on ontology-based descriptions, inference-based techniques shall enable automated discovery, composition, mediation, and execution of Web services (see [8] for a recent overview).

One of the central tasks in SOA-systems is *discovery*. As the first preprocessing step for solving a given request, this is concerned with detecting usable Web services with respect to the provided and requested functionalities [11]. The major requirement for semantically enabled Web service discovery is *accuracy*: the discovery result should encompass all those Web service that are definitely able to solve the given request. Only if this is given, the subsequent processing steps in the system can rely on the appropriateness of the discovery component.

To ensure accuracy of semantically enabled Web service discovery, an appropriate formal description model along with adequate matchmaking techniques is needed. Most existing approaches that have been presented for semantically discovery of Web services lack in this respect. Commonly, the underlying model of the world and the formal descriptions do not allow to precisely describe requested and provided functionalities, which results in inaccurate matchmaking techniques. In order to overcome these deficiencies, we present a framework for Web service discovery with sufficiently rich functional descriptions and semantic matchmaking techniques that allow to precisely determine the usability of a Web service.

Our approach considers a state-based model of the world that Web services act in. The functionality provided by a Web service is formally described in terms of pre- and post-state constraints of possible executions. We differentiate two notions for formally specifying requests: *goal templates* are generic objective descriptions whereof *goal instances* are created by instantiation. The former are kept in the system while the latter are defined during runtime. Due to their formal interrelation, we define a two-phase discovery procedure: the discovery results for goal templates serve as a pre-filter for Web service discovery for goal instances. In this paper, we explain the underlying model and the formal description of provided and requested functionalities, and upon this specify accurate matchmaking techniques for both the goal template and the goal instance level.

The presented approach is independent of the description language for Web services and goals. Therefore, we aim at a generic specification for accurate Web service discovery that can be adopted to several Semantic Web service frameworks. In this paper, we use classical first-order logic for illustration and demonstration. Although we introduce the framework in the context of Web services, our approach is not limited to this but can be exploited to several other technologies that are proposed for the next generation of the Internet and deal with formally described provided and requested functionalities.

The paper is structured as follows. At first, Section 2 explains the concepts of Web services and goals, and introduces our two-phase discovery framework. Then, Section 3 defines the formal description model for requested and provided functionalities. On the basis of this, Section 4 specifies the semantic matchmaking techniques for discovery on the goal template and on the goal instance level. Section 5 illustrates the specification of functional descriptions and demonstrates the discovery techniques within an illustrative example. Section 6 examines related work on semantic Web service discovery and positions our approach therein. Finally, Section 7 concludes the paper.

2 Concepts and Approach

The specification of semantic matchmaking techniques for discovery is strongly dependent on the underlying understanding and the formal description of Web services and requests for these. This section first explains our understanding of Web services and goals, i.e. formalized client objectives, and then introduces our approach discussing the meaning of a match.

In the context of discovery, our main focus is on formalized provided and requested functionalities and the logical relationships between them that denote the usability of a Web service for solving a given request. Prominent frameworks for Semantic Web services lack of precisely defined semantics for functional descriptions (see Section 6 for an extensive discussion). Hence, we apply so-called *Abstract State Spaces* (short: ASS) as the underlying formal model. Presented in [13], this defines a state-based model for Web services and the world they act in with rigorous formal definitions. As discussed later, this allows to define semantic matchmaking on the level of executions of Web services, and therewith is sufficiently expressive for our purposes.

2.1 Web Services and Goals

In accordance to the commonly accepted conception, we understand a Web service as a computational facility that is invocable over the Internet via an interface [4, 2]. Following the approach of Semantic Web services, a Web service is associated with a comprehensive, ontology-based description that covers all aspects relevant for automated detection and usage for solving client requests. The description element of central interest in our context is the **functional description** that formally specifies the *overall functionality* provided by the Web services (in accordance to WSMO capabilities [18] and OWL-S Service Profiles [20]).

As the counterpart to Web services in service-oriented architectures, goals represent formally described client objectives. Integrating works around WSMO – the only Semantic Web service framework that defines goals as a top level notion – we distinguish two notions: **Goal Templates** formally specify client objectives in a generic manner along with all information required for automated Web service usage, and **Goal Instances** define concrete client requests by instantiation of a goal template with concrete inputs. As a simple example, consider the objective of travelling from Innsbruck to Vienna. Here, the goal template specifies the origin and destination to be cities located in Austria as required inputs; the goal instance instantiates them with 'Innsbruck' and 'Vienna' as concrete objects in the domain that satisfy the goal template specification.

Figure 1 shows the relation of Web services and goals on a conceptual level with further explanations below.

An important restriction is that we only consider Web services that provide a *deterministic functionality*. Adopted from general computational theory [25], this means that all outputs and post-execution effects that result from the execution of a Web service are completely dependent of the inputs provided for its invocation and consumption. Without this restriction, a Web service could create arbitrary objects in the world that are not related to a usage request. This would contradict the composability of Web services, which is a central prerequisite for their usage in service-oriented architectures [7].

In consequence, goals define requests for such functionalities. Thereby, goal templates are pre-defined schemas, while goal instances are defined at runtime. This distinction eases the formulation of requests by clients, and it allows to allocate expensive operations for Web service detection at the level of goal templates [29, 6]. As the element of central interest for Web service discovery, a goal template specifies a generic objective description as a *requested functionality*. A goal instance defines a variable assignment for the input values specified therein. These concrete values are used for invoking and consuming the Web

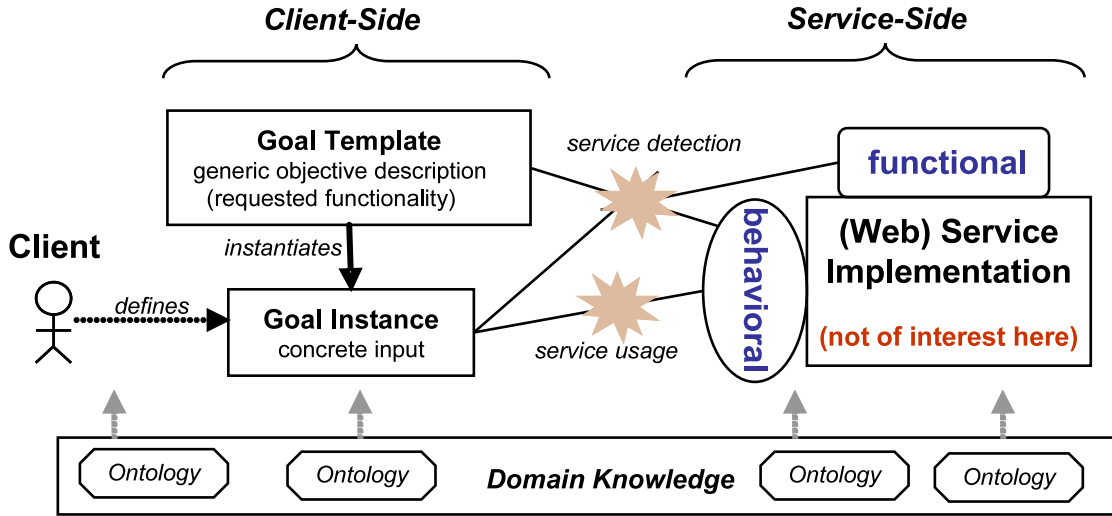


Figure 1: Web services, Goals, Goal Instances

service that has been discovered.

2.2 The Meaning of a Match in Web Service Discovery

We now turn towards the usability of a Web service for solving a request. In particular, we discuss the notion of a *match*, i.e. when a Web service is considered to be usable for solving a goal template, respectively a goal instance. We therefore subsequently introduce Abstract State Spaces (short: ASS) as defined in [13].

An Abstract State Space \mathcal{A} is defined over a signature Σ with respect to consistent background domain knowledge Ω (most commonly defined as an ontology). A state $s \in \mathcal{A}$ is a static snapshot of the world. State transitions are achieved by executions of Web services as well as by other acting entities. Each state is accessible via an observation function $\omega(s)$, a total surjective mapping that assigns Σ -interpretations over the universe $\mathcal{U}_{\mathcal{A}}$, a non-empty set of objects that satisfy Ω .

Sufficient for our context, the ASS model defines a Web service as a pair $W = (IF, \iota)$ such that $IF = (i_1, \dots, i_n)$ is a finite sequence of names denoting all required input values for W , and ι is the implementation of W . An execution of W is triggered by an input binding $\beta : \{i_1, \dots, i_n\} \rightarrow \mathcal{U}_{\mathcal{A}}$, i.e. a total function that assigns objects of $\mathcal{U}_{\mathcal{A}}$ to the IF -names. A particular execution of W for a given β and a specific start state s_0 denotes a finite sequence of state transitions $\tau = (s_0, \dots, s_m)$ such that all $s_0, s_j, s_m \in \mathcal{A}$. Due to the deterministic behavior of ι , for each τ we can infer the termination state s_m from the start state s_0 .

Hence, the functionality provided by a Web service W denotes a set of possible finite sequences of state transitions, denoted by $\{\tau\}_W$. As illustrated in Figure 2, 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 invocation of W with a particular input binding β .

Let us now consider an objective that a client wants to achieve by using Web services. In accordance to related AI research (e.g. [5, 22]), we understand this as the desire of the client to change the world from its current state into a state wherein the objective is satisfied. The purpose of Web service discovery is to find a Web service that can perform this change of the world.

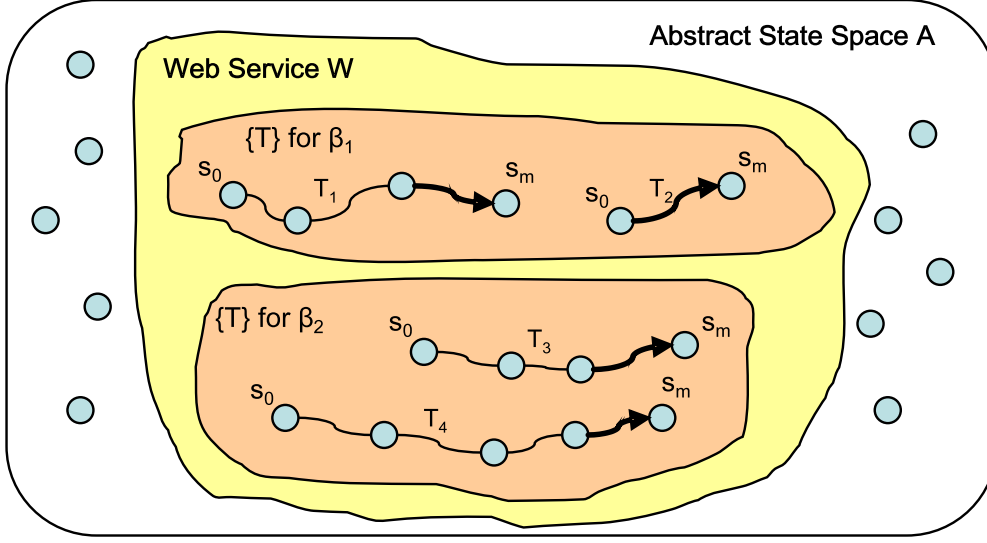


Figure 2: Web Service, Executions, Input Bindings in ASS

In our approach, a goal template \mathcal{G} describes such an objective as restrictions on the initial state and the desired final state to be achieved. In the ASS model, this means that \mathcal{G} specifies possible sequences of state transitions $\{\tau\}_{\mathcal{G}}$ such that 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. In consequence, we consider a Web service W to be usable for achieving an objective described by \mathcal{G} if there exists at least one execution of W that is a possible solution for \mathcal{G} , i.e.: $\exists \tau. \tau \in (\{\tau\}_{\mathcal{G}} \cap \{\tau\}_W)$.

For specifying a concrete client request, a goal instance $GI(\mathcal{G})$ defines concrete values for the inputs specified in \mathcal{G} . Essentially, such a goal instantiation is defined over an input binding β such that $GI(\mathcal{G}) = \mathcal{G}(\beta)$; these concrete values constitute the input binding for invoking the Web service as discussed above. Such a goal instantiation restricts the possible solutions for $GI(\mathcal{G})$ to a subset of those for \mathcal{G} , so that $\{\tau\}_{GI(\mathcal{G})} \subset \{\tau\}_{\mathcal{G}}$. Hence, for a Web service W to be usable for solving a goal instance there must be at least one execution of W for this particular input binding that is also a solution for $GI(\mathcal{G})$. Hence, the matching condition for goal instances is $\exists \tau. \tau \in (\{\tau\}_{GI(\mathcal{G})} \cap \{\tau\}_{W(\beta)})$.

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 that constitutes the input binding for invoking W . Let $\{\tau\}$ denote a set of sequences of state transitions in an Abstract State Space \mathcal{A} such that:

$$\begin{aligned} \{\tau\}_W &:= \text{the set of possible executions of } W \\ \{\tau\}_{\mathcal{G}} &:= \text{the set of } \tau \text{ that can solve } \mathcal{G} \\ \{\tau\}_{W(\beta)} \subset \{\tau\}_W &:= \text{the set of possible executions of } W \text{ for } \beta \\ \{\tau\}_{GI(\mathcal{G})} \subset \{\tau\}_{\mathcal{G}} &:= \text{the set of } \tau \text{ that can solve } GI(\mathcal{G}) \end{aligned}$$

We define a **match** as the basic condition for the usability of a Web service as:

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

It holds that:

- (iii) $match(GI(\mathcal{G}), W) \Rightarrow match(\mathcal{G}, W)$, and hence
- (iv) $\neg match(\mathcal{G}, W) \Rightarrow \neg match(GI(\mathcal{G}), W)$

Definition 1 summarizes the above discussion in a concise manner. We make two observations constitute the foundation of our framework for Web service discovery. At first, evaluating the basic matching condition in clauses (i) and (ii) requires matchmaking on formal descriptions of the functionalities provided by Web services and requested by goals – without this, we would need to perform test runs of W for determining its usability. We specify accurate semantic matchmaking techniques for the goal template level in Section 4.1 and for the goal instance level in Section 4.2.

Secondly, there is an invariable correlation of the matches on the goal template and the goal instance level. Clause (iii) states that a Web service that is usable for solving a goal instance is also usable for the corresponding goal template. This trivially holds because $\{\tau\}_{GI(\mathcal{G})} \subset \{\tau\}_{\mathcal{G}}$, so that if there is a Web service W with $match(GI(\mathcal{G}), W)$ then also $match(\mathcal{G}, W)$. Clause (iv) states that a Web service that is not usable for a goal template is also not usable for any of its goal instances. This also trivially holds as the logical complement of clause (iii), as $(a \Rightarrow b) \Leftrightarrow (\neg b \Rightarrow \neg a)$.

Because of this, we define a two-phase discovery framework. At first, usable Web services for goal templates \mathcal{G} are determined. This is performed at design time, i.e. when a new goal template is defined. At runtime, Web service discovery for the concrete goal instance $GI(\mathcal{G})$ is performed, whereby merely the set of Web services that are usable for the corresponding goal template \mathcal{G} need to be taken into consideration. As shown in [27], this allows to improve the runtime efficiency of discovery engines. However, in the remainder of this paper we concentrate on the formal description of requested and provided functionalities as well as semantic matchmaking techniques for precisely determining the usability of a Web service on both the goal template and goal instance level.

3 Formal Functional Descriptions

The following specifies the means for formally describing the functionality provided by a Web services and requested by a goal. As the basis for automatically determining the usability of a Web service for solving the goal by semantic matchmaking, this need to properly describe the possible executions of a Web service as well as the possible solutions of a goal. We first explain the definition and meaning, and then present their simulation by conventional logical formulae that eases handling of functional descriptions.

3.1 Definition and Semantics

Deterministic functionalities that we consider here can most properly be described in terms of preconditions and effects [10]. Such a description denotes a black box description of a functionality that neglects intermediate states that are traversed during the execution as well as technical aspects on the invocation and consumption of a Web service.

The essential requirements for the appropriateness of functional descriptions is that they allow to explicitly specify the dependency of the effect on the precondition and the changes of the world that result from the execution of a Web service. While prominent frameworks for Semantic Web services lack in this respect (see Section 6), we apply functional descriptions in the ASS model as defined in [13].

Recalling from above, an Abstract State Space \mathcal{A} is defined over a signature Σ and some domain knowledge Ω . Preconditions and effects are defined as statements in a logic $\mathcal{L}(\Sigma)$. To properly specify the changes of the world that result from a Web service execution, the ASS model defines extensions to the signature. At first, Σ_S denotes *static symbols* that are not changed by the execution of a Web service such that for all $s, s' \in \mathcal{A}$ and $\alpha \in \Sigma_S : \alpha(s) = \alpha(s')$. Secondly, Σ_D denotes *dynamic symbols* that are explicitly changed by the execution. The extension α_{pre} allows to explicate the dependency of a symbol in the effect on its value in the start-state, i.e. for all $s_0, s' \in \mathcal{A}$ and $\alpha \in \Sigma_D : \alpha(s_0) = \alpha_{pre}(s')$. Finally, the symbol *out* denotes the computational outputs of a Web service execution.

To explicitly specify the dependency between the post- and the pre-state, the ASS model defines so-called input variables $IF = (i_1, \dots, i_n)$ as the formal description of the inputs required by a Web service. Their scope is the complete functional description, and they occur as free variables in the precondition ϕ^{pre} and the effect ϕ^{eff} . Concrete inputs for invoking a Web service – respectively for instantiating a goal – are defined as an input binding $\beta : \{i_1, \dots, i_n\} \rightarrow \mathcal{U}_A$, a total function that assigns objects of the universe of \mathcal{A} to the IF -variables.

Definition 2. In an Abstract State Space \mathcal{A} , a **Functional Description** \mathcal{D} is defined as a 5-tuple $\mathcal{D} = (\Sigma_{\mathcal{A}}, \Omega, IF, \phi^{pre}, \phi^{eff})$ such that

- (i) $\Sigma_{\mathcal{A}}$ is a signature Σ explicitly extended with Σ_S (static symbols) and Σ_D (dynamic symbols)
- (ii) $\Omega \subseteq \mathcal{L}(\Sigma)$ defines consistent domain knowledge
- (iii) IF is a set of variables i_1, \dots, i_n whose scope is \mathcal{D} ; this denotes all required input values wherefore an input binding $\beta : \{i_1, \dots, i_n\} \rightarrow \mathcal{U}_A$ assigns objects of the universe \mathcal{U}_A
- (iv) ϕ^{pre} is a statement in $\mathcal{L}(\Sigma)$ that constraints the initial state s_0 wherein the only free variables are a subset of IF
- (v) ϕ^{eff} is a statement in $\mathcal{L}(\Sigma)$ that constraints the final state s_m wherein only IF can occur as free variables and *out* denotes the output.

The mathematical structure of a functional description is identical for provided and requested functionalities. A Web service W is considered to provide the functionality described in \mathcal{D} if for all possible executions $\tau = (s_0, \dots, s_m) \in \{\tau\}_W$ it holds that the start-state s_0 satisfies the precondition ϕ^{pre} and the termination state s_m satisfies the effect ϕ^{eff} . For a goal template, \mathcal{D} formally describes all possible solutions $\tau \in \{\tau\}_{\mathcal{G}}$ with respect to their initial and final states.

While we discuss examples in Section 5, the following defines the formal semantics of functional descriptions. Here, the observation function $\omega(s)$ assigns a Σ -interpretation over the universe $\mathcal{U}_{\mathcal{A}}$ to each abstract state $s \in \mathcal{A}$, and $I, \beta \models_{\mathcal{L}(\Sigma)} \phi$ expresses that a formula ϕ is satisfied under a Σ -interpretation I and an input binding β that defines a variable assignment of the IF -variables.

Definition 3. Let $W = (IF, \iota)$ be a Web service in an Abstract State Space \mathcal{A} , and $\mathcal{D} = (\Sigma_{\mathcal{A}}, \Omega, IF_{\mathcal{D}}, \phi^{pre}, \phi^{eff})$ be a functional description with an input binding β for the $IF_{\mathcal{D}}$ -variables.

We say that W satisfies \mathcal{D} , denoted by $W \models_{\mathcal{A}} \mathcal{D}$, if and only if:

- (i) there is a bijection $\pi : IF \rightarrow IF_{\mathcal{D}}$ such that for each input $i_j \in IF$ required by W there is a corresponding input variable $i'_j \in IF_{\mathcal{D}}$
- (ii) for all possible executions $\tau = (s_0, \dots, s_m) \in \tau_W$ it holds that if $\omega(s_0), \beta \models_{\mathcal{L}(\Sigma)} \phi^{pre}$ then $\omega(s_m), \beta \models_{\mathcal{L}(\Sigma)} \phi^{eff}$.

3.2 Simulation as Logical Formulae

While the ASS model and hence functional descriptions therein are language-independent, we use classical first-order logic (FOL) as the knowledge definition language $\mathcal{L}(\Sigma)$ in the following. In order to reduce the complexity of dealing with functional descriptions, we define a first-order logic structure that properly simulates the semantics of a functional description. Essentially, this structure defines the precondition and effect as FOL formulae and simulate the formal semantics of functional descriptions from Definition 3 as a logical implication. As we will see below, this allows to apply standard notions from model-theoretic semantics like entailment and logical equivalence for specifying logical relationships and operations on functional descriptions [26].

The problem of representing a functional description \mathcal{D} that is defined in accordance to Definition 2 as a conventional formula $\phi_{\mathcal{L}}^{\mathcal{D}}$ in a static language \mathcal{L} is that we need to deal with different logical frameworks. While the former representation is concerned with states and transitions between, the latter is concerned with models of formulae and does not provide means for presenting dynamics. The following discusses this, using classical first-order logic as an expressive language for static knowledge specification with model-theoretic semantics. We commence with discussing the meaning of functional descriptions, and derive a formal substantiation for representing functional descriptions as conventional FOL formulae.

Above, a functional description \mathcal{D} restricts the possible sequences of state transitions $\tau = (s_0, \dots, s_n)$ in \mathcal{A} with respect to their pre-state s_0 and their post-state s_n ; they elide all intermediate states that are traversed during execution of a Web service. We can omit the dynamic aspects of states and transitions between them, and thus represent a functional description \mathcal{D} by an FOL structure $sim(\mathcal{D}) = (IF, \phi^{\mathcal{D}})$ that is defined over the same signature as \mathcal{D} and with respect to the same formalized domain knowledge Ω . Therein, the formula $\phi^{\mathcal{D}}$ defines a logical implication of the post-state constraint ϕ^{eff} by the pre-state constraint ϕ^{pre} .

To properly capture the correlation and dependence of the pre- and post-state constraints, we define $\phi^{\mathcal{D}}$ as $[\phi^{pre}]_{\Sigma_D^{pre} \rightarrow \Sigma_D} \Rightarrow \phi^{eff}$ with the following correspondence to Definition 2: $IF = (i_1, \dots, i_n)$ correspond to the IF -variables that occur as free variables in the state constraints, ϕ^{pre} corresponds to the precondition, and ϕ^{eff} to the effect. Defined in [13], $[\phi]_{\Sigma_D^{pre} \rightarrow \Sigma_D}$ denotes the formula ϕ' that is derived from ϕ by replacing

any dynamic symbol $\alpha \in \Sigma_D$ by its corresponding pre-variant $\alpha_{pre} \in \Sigma_D^{pre}$. This allows handling of dynamic symbols that are changed by the execution of a Web service, because each symbol $\alpha_{pre} \in \Sigma_D^{pre}$ that occurs in ϕ^{eff} is denoted by the same symbol in the re-written precondition $[\phi^{pre}]_{\Sigma_D^{pre} \rightarrow \Sigma_D}$.

A Σ -interpretation I that is a model of $sim(\mathcal{D})$ corresponds to the termination state s_n of the execution of a Web service W with $W \models_{\mathcal{A}} \mathcal{D}$ for a specific input binding β for the IF -variables i_1, \dots, i_n . The reason is that the (error-free) execution of W for a specific input binding results in provision of objects whose properties are described by the effect constraint. This is dependent of the precondition and the free input variables, which are bound to concrete objects by β . Hence, we can describe the information space $\omega_{\beta}(s_n)$ of the termination state of a specific execution of a Web service by an interpretation. As $sim(\mathcal{D})$ is defined over the same signature and models the intended relationship between the precondition and effect in \mathcal{D} , it holds that every interpretation I that simulates $\omega_{\beta}(s_n)$ is a model of $sim(\mathcal{D})$. Hence, we say that $sim(\mathcal{D})$ semantically simulates \mathcal{D} , denoted as $sim(\mathcal{D}) \simeq \mathcal{D}$. Figure 3 illustrates this correlation that we formally substantiate in the following.

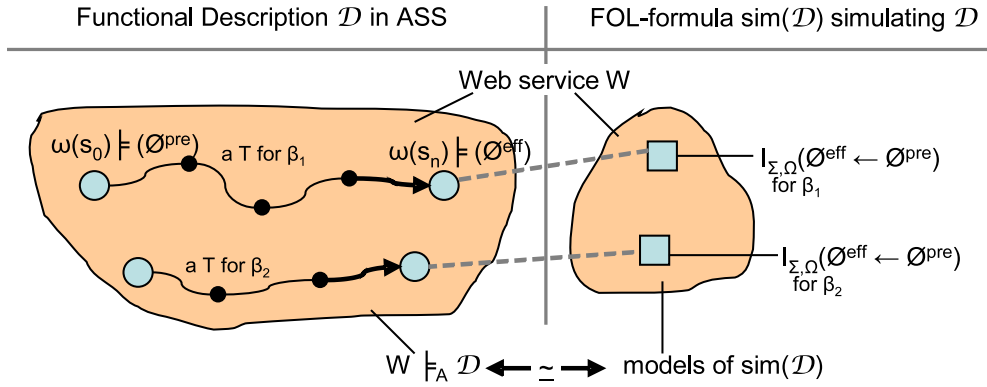


Figure 3: Correlation of a Functional Description \mathcal{D} and $sim(\mathcal{D})$

Definition 4. Let a Web service W be a pair $W = (IF, \iota)$ with a set of input variables $IF = (i_1, \dots, i_n)$ and an implementation ι . Let \mathcal{A} be an Abstract State Space with a signature $\Sigma_{\mathcal{A}} = (\Sigma_S \uplus \Sigma_D \uplus \Sigma_D^{pre} \uplus out)$ and the universe $\mathcal{U}_{\mathcal{A}}$ as a non-empty set of objects. Let a sequence of state transitions $\tau_W = (s_0, \dots, s_n)$ denote an execution of W in \mathcal{A} over $\Sigma_{\mathcal{A}}$. For the execution $\tau_W(\beta)$ of W for a specific input binding $\beta : (i_1, \dots, i_n) \rightarrow \mathcal{U}_{\mathcal{A}}$ let a $\Sigma_{\mathcal{A}}$ -interpretation $\mathcal{I}_{\tau_W}(\beta) = (\mathcal{U}_{\mathcal{A}}, I_{\tau_W}(\beta))$ denote the mapping of symbols in $I_{\tau_W}(\beta)$ to $\mathcal{U}_{\mathcal{A}}$ in the termination state s_n of $\tau_W(\beta)$. Let a Σ -interpretation $\mathcal{I}_W(\beta) = (\mathcal{U}_{\mathcal{A}}, I_W(\beta))$ denote the mapping of symbols $I_W(\beta)$ to the objects in the universe $\mathcal{U}_{\mathcal{A}}$.

We define the simulation of a Web service execution $\tau_W(\beta)$ by a $\Sigma_{\mathcal{A}}$ -interpretation $\mathcal{I}_W(\beta)$ as

$$\tau_W(\beta) \simeq \mathcal{I}_W(\beta) \quad \text{iff.} \quad \text{for } \tau_W(\beta) = \iota(s_0, \beta) = (s_0, \dots, s_n) \text{ and } s_n = (\mathcal{U}_{\mathcal{A}}, I_{\tau_W}(\beta)) \text{ holds}$$

- (i) for all predicates $\alpha \in \Sigma$ with the arity m holds

$$\forall x_1, \dots, x_m \in \mathcal{U}_{\mathcal{A}}. (x_1, \dots, x_m) \in I_{\tau_W}(\beta)(\alpha) \Leftrightarrow (x_1, \dots, x_m) \in I_W(\beta)(\alpha).$$
- (ii) for all functions $f \in \Sigma$ with the arity m holds

$$\forall x_1, \dots, x_m \in \mathcal{U}_{\mathcal{A}}. I_{\tau_W}(\beta)(f)(x_1, \dots, x_m) = x_0 \Leftrightarrow I_W(\beta)(f)(x_1, \dots, x_m) = x_0.$$

This definition states that the execution of a Web service W for a particular input binding β can be semantically simulated by an interpretation $\mathcal{I}_W(\beta)$ that is defined over the same signature as W . Recall that the

signature Σ in an Abstract State Space is extended with static symbols Σ_S , dynamic symbols Σ_D and their pre-variants Σ_D^{pre} , and the *out* symbol (see Definition 2). Thus, the world in the termination state $\omega(s_n)$ of an execution of W for a particular input binding β covers all relevant aspects: the mapping to objects that exists in s_n described by ϕ^{eff} in dependence of the pre-state constraints ϕ^{pre} – which is explicitly defined via the signature extensions Σ_S , Σ_D , and Σ_D^{pre} – as well as the output objects that are explicitly denoted by *out*. This can be represented by a particular Σ -interpretation over Σ_A such that all predicate and function symbols have the same meaning as in $\omega(s_n)$.

Definition 5. Let \mathcal{D} be a functional description in an Abstract State Space \mathcal{A} with a precondition ϕ^{pre} and an effect ϕ^{eff} defined in first-order logic over Σ_A and with respect to formalized domain knowledge Ω . Let i_1, \dots, i_n be the input variables whose scope is \mathcal{D} and that occur as free variables in ϕ^{pre} and ϕ^{eff} . Let $W \models_{\mathcal{A}} \mathcal{D}$ denote that W is a capability model of \mathcal{D} such that all possible executions $\tau_W = (s_0, \dots, s_n)$ of W satisfy \mathcal{D} . The set of all input bindings for IF in \mathcal{A} is denoted by $In_{\mathcal{A}}(IF)$.

We define the simulation of a functional description \mathcal{D} by a first-order logic formula ϕ as

- $$\mathcal{D} \simeq \phi \text{ iff. } \begin{array}{l} \text{for all } s_0 \in \mathcal{A} \text{ and for all } \beta \in In_{\mathcal{A}}(IF) \text{ holds that} \\ \text{(i) for each execution } \tau_W(\beta) \text{ of each Web service } W \text{ with } W \models_{\mathcal{A}} \mathcal{D} \text{ holds that} \\ \text{for all } \Sigma\text{-interpretations } \mathcal{I} \text{ such that } \tau_W(\beta) \simeq \mathcal{I} \text{ holds that } \mathcal{I} \models \phi. \\ \text{(ii) for all } \Sigma\text{-interpretations } \mathcal{I} \text{ such that } \mathcal{I} \models \phi \text{ holds that} \\ \text{for all Web services } W \text{ such that for each execution } \tau_W(\beta) \text{ of } W \\ \text{holds } \tau_W(\beta) \simeq \mathcal{I} \text{ holds that } W \models_{\mathcal{A}} \mathcal{D}. \end{array}$$

This definition states that a functional description \mathcal{D} that is specified in accordance to Definition 2 can be simulated by a FOL-formula ϕ that is defined over the same signature as \mathcal{D} . It therefore has to hold that each Σ -interpretation \mathcal{I} that is a model of ϕ simulates an execution of a Web service W that provides the functionality described by \mathcal{D} such that the models $\mathcal{M}(\phi)$ as the set of such interpretations covers each possible execution of W . In combination with Definition 4, this provides the correctness criterion of a first-order structure that represents a functional description \mathcal{D} by maintaining the formal semantics.

Definition 6. Let \mathcal{D} be a functional description defined in an Abstract State Space \mathcal{A} . Let $sim(\mathcal{D})$ be a first-order structure defined over Σ_A as a pair $sim(\mathcal{D}) = (IF, \phi^{\mathcal{D}})$ with $IF = (i_1, \dots, i_n)$ being the set of input variables defined in \mathcal{D} , and $\phi^{\mathcal{D}}$ being a first-order logic formula of the form $[\phi^{pre}]_{\Sigma_D^{pre} \rightarrow \Sigma_D} \Rightarrow \phi^{eff}$ such that:

- (i) ϕ^{pre} is the formula defining the precondition \mathcal{D} wherein i_1, \dots, i_n occur as free variables,
- (ii) ϕ^{eff} is the formula defining the effect of \mathcal{D} wherein i_1, \dots, i_n occur as free variables,
- (iii) and $[\phi]_{\Sigma_D^{pre} \rightarrow \Sigma_D}$ as the formula ϕ' derived from ϕ by replacing every dynamic symbol $\alpha \in \Sigma_D$ by its corresponding pre-variant $\alpha_{pre} \in \Sigma_D^{pre}$.

This defines a specific first-order structure for representing functional descriptions. The formula $\phi^{\mathcal{D}}$ defines an implication between the precondition formula and the effect formula, stating that if the precondition is satisfied then the effect will be satisfied by executing a Web service W with $W \models_{\mathcal{A}} \mathcal{D}$. As in Definition 2, the input variables are kept separate from $\phi^{\mathcal{D}}$ so that the $\phi^{\mathcal{D}}$ can only be evaluated if a concrete input binding $\beta : (i_1, \dots, i_n) \rightarrow \mathcal{U}_A$ is provided. Therewith, $sim(\mathcal{D})$ simulates a functional description \mathcal{D} by omitting the dynamic aspects related to states and transitions between them.

For illustrating the precondition rewriting, let us recall the bank account withdrawal example from above. The input variables are $IF = \{a, x\}$, and the precondition specifies $\phi^{pre} = \text{account}(a) \wedge \text{float}(x) \wedge \text{balance}(a) \geq x$ (the only occurring variables are a, x ; as input variables, these are free variables in ϕ^{pre}).

The only dynamic symbol is *balance*. Applying clause (iii), this is replaced by its pre-variant $balance_{pre}$ in the re-written precondition. Hence, $\phi^{\mathcal{D}} = account(a) \wedge float(x) \wedge balance(a)_{pre} \geq x \Rightarrow account(a) \wedge balance(a) = balance(a)_{pre} - x$. Therewith, the pre-variant of the dynamic symbol occurring in the effect-part of $\phi^{\mathcal{D}}$ is denoted by the same variable in the precondition-part, so that the dependence between the two parts is explicitly specified.

Theorem 1. *Let \mathcal{D} be a functional description in an Abstract State Space \mathcal{A} with a precondition ϕ^{pre} and an effect ϕ^{eff} defined in first-order logic over $\Sigma_{\mathcal{A}}$ and with respect to formalized domain knowledge Ω . Let i_1, \dots, i_n be the input variables whose scope is \mathcal{D} and that occur as free variables in ϕ^{pre} and ϕ^{eff} . Let $sim(\mathcal{D})$ be a first-order structure defined over $\Sigma_{\mathcal{A}}$ as a pair $sim(\mathcal{D}) = (IF, \phi^{\mathcal{D}})$ with \mathcal{D} is a first-order logic formula of the form $[\phi^{pre}]_{\Sigma_D^{pre} \rightarrow \Sigma_D} \Rightarrow \phi^{eff}$.*

Then, $\mathcal{D} \simeq sim(\mathcal{D})$.

$sim(\mathcal{D})$ is defined over the same signature Σ with respect to the same background knowledge Ω , and uses the same precondition ϕ^{pre} and effect ϕ^{eff} as \mathcal{D} . The logical implication of the effect by the re-written precondition simulates the meaning of a functional description as defined in Definition 3: for a particular $\tau = (s_0, \dots, s_m)$, if the start state s_0 satisfies the precondition ϕ^{pre} then the termination state s_m must satisfy the effect ϕ^{eff} . Therewith, a Σ -interpretation that is a model of $sim(\mathcal{D})$ is the same as $\omega(s_m)$ for the particular τ . Moreover, this holds for all $\tau \in \{\tau\}_W$ for each Web service W with $W \models_{\mathcal{A}} \mathcal{D}$; otherwise, $W \models_{\mathcal{A}} \mathcal{D}$ does not hold. We provide a proof of this theorem in Appendix A.2 of this document.

Proposition 1. *Descriptions] Another representation of \mathcal{D} is a first-order logic structure $sim(\mathcal{D})_2 = (IF, \phi_+^{\mathcal{D}})$ with $IF = (i_1, \dots, i_n)$ being the set of input variables defined in \mathcal{D} , and $\phi_+^{\mathcal{D}}$ being a first-order logic formula of the form $[\phi^{pre}]_{\Sigma_D^{pre} \rightarrow \Sigma_D} \wedge \phi^{eff}$.*

It holds that $\phi_+^{\mathcal{D}} \models \phi^{\mathcal{D}}$.

The representation of a functional description \mathcal{D} by $sim(\mathcal{D})_2$ defines a conjunction of the precondition and the effect formulae. For $\mathcal{I}_W(\beta) \models sim(\mathcal{D})_2$, it has to hold that $\beta \models \phi^{pre}$ and $\beta \models \phi^{eff}$. Therewith, this representation of a functional description \mathcal{D} by a first-order logic structure only considers case (1) as discussed above. If the preconditions is not satisfied as in case (2), then the Web service is considered to be not executable – which is a more strict reading of Definition 2. Hence, representing $sim(\mathcal{D})_2$ is a stronger way to represent a functional description that logically entails $sim(\mathcal{D})$. Mainly usable for defining operations and inference rules that are only concerned with the objects retrievable by executing Web services, this is referred to as the *implementation perspective* in literature (e.g. [12]); accordingly, $sim(\mathcal{D})$ is called the *modelling perspective*.

4 Semantic Matchmaking

This section specifies the semantic matchmaking techniques for Web service discovery. With respect to the two-phase discovery framework as introduced in Section 2.2, we first address discovery for the goal template level and then on the goal instance level. Finally, we integrate the semantic matchmaking techniques for both levels.

In order to precisely determine whether a Web service is usable with respect to the basic matching conditions from Definition 1, the matchmaking techniques defined in the following work on formal functional descriptions as defined in Section 3. For the goal template level, we define matchmaking degrees that denote different relationships between the possible solutions of the requested functionality and the possible executions of a Web service. Each of these degrees implies certain conditions on the usability of the Web service for solving a goal instance. We identify these conditions, and present an extension for semantic matchmaking on the goal instance level. While this section provides the definitions, we demonstrate the matchmaking techniques within an illustrative example in Section 5.

4.1 Goal Template Level Discovery

For Web service discovery on the goal template level, we define matchmaking degrees on formal functional descriptions of a goal template \mathcal{G} and a Web service W . Denoting the relationship between possible executions $\{\tau\}_W$ of a Web service and possible solutions $\{\tau\}_\mathcal{G}$ for a goal template, these matchmaking degrees provide a means for evaluating the basic matching condition for the usability of a Web service on the goal template level.

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. Regarding the usability of a Web service between the goal template and goal instance level, under the *exact* degree W can be used for any goal instance $GI(\mathcal{G})$ that instantiates \mathcal{G} . For the three non-exact matchmaking degrees, this is only given if the input binding β defined for $GI(\mathcal{G})$ triggers an execution of W that results in a termination state that satisfies the goal description. We discuss this in more detail in the following.

The following provides the definition of each matchmaking degree and discusses its meaning for the usability of a Web service for solving a goal instance. Let $\mathcal{D}_\mathcal{G}$ be the description of the functionality requested in a goal template \mathcal{G} , and \mathcal{D}_W as description of the functionality provided by a Web service W with $W \models_{\mathcal{A}} \mathcal{D}_W$. We define criteria for each matching degree over the $sim(\mathcal{D}_\mathcal{G})$ and $sim(\mathcal{D}_W)$ from Theorem 1, along with an explicit quantification of input bindings β . Here, $\Omega \models \forall \beta. \phi^{\mathcal{D}_\mathcal{G}} \Leftrightarrow \phi^{\mathcal{D}_W}$ defines that, under consideration of the domain knowledge Ω , for all Σ -interpretations I under all possible input bindings β holds that $I \models sim(\mathcal{D}_\mathcal{G})$ if and only if $I \models sim(\mathcal{D}_W)$. As the condition for the *exact* degree, this expresses that every possible execution of W is a solution for \mathcal{G} and vice versa. Refining the matching degree definitions for goal-based Web service discovery from [11], we therewith obtain a means for precisely expressing the relationship between $\{\tau\}_\mathcal{G}$ and $\{\tau\}_W$ on the basis of sufficiently rich descriptions.

exact($\mathcal{D}_\mathcal{G}, \mathcal{D}_W$): $\Omega \models \forall \beta. \phi^{\mathcal{D}_\mathcal{G}} \Leftrightarrow \phi^{\mathcal{D}_W}$

This degree denotes that the functionality requested by the goal and the one provided by the Web service are semantically identical, so that $\{\tau\}_\mathcal{G} = \{\tau\}_W$. Here, W can be used for solving any goal instance $GI(\mathcal{G})$, because every input binding β that is a valid instantiation of \mathcal{G} triggers a $\tau \in \{\tau\}_W$ such that $\tau \in \{\tau\}_{GI(\mathcal{G})} \subset \{\tau\}_\mathcal{G}$. However, this degree denotes the most implausible situation occurring in real-world settings.

plugin($\mathcal{D}_G, \mathcal{D}_W$): $\Omega \models \forall \beta. \phi^{\mathcal{D}_G} \Rightarrow \phi^{\mathcal{D}_W}$

This defines that the requested functionality is more specific than the provided functionality, i.e. $\{\tau\}_G \subseteq \{\tau\}_W$. Under this degree, all possible solutions for the goal can be provided by the Web service but there can exist a $\tau \in \{\tau\}_W$ such that $\tau \notin \{\tau\}_G$. Thus, for W to be usable for a goal instance $GI(\mathcal{G})$, it has to hold that the input binding β defined for $GI(\mathcal{G})$ triggers such an execution of W that $\{\tau\}_{W(\beta)} \in \{\tau\}_G$.

subsume($\mathcal{D}_G, \mathcal{D}_W$): $\Omega \models \forall \beta. \phi^{\mathcal{D}_G} \Leftarrow \phi^{\mathcal{D}_W}$

As the opposite to the plugin degree, this denotes that the requested functionality is more general than the provided one, so that $\{\tau\}_G \supseteq \{\tau\}_W$. This means that all executions of W can satisfy \mathcal{G} , but there are possible solutions for \mathcal{G} that cannot be provided by W . In consequence, for W to be usable for a goal instance $GI(\mathcal{G})$, it has to hold that $\{\tau\}_{GI(\mathcal{G})} \subseteq \{\tau\}_W$. This is given if the input binding β defined for $GI(\mathcal{G})$ allows to invoke W .

intersect($\mathcal{D}_G, \mathcal{D}_W$): $\Omega \models \exists \beta. \phi^{\mathcal{D}_G} \wedge \phi^{\mathcal{D}_W}$

As the weakest degree that satisfies the matching condition in clause (i) of Definition 1, this denotes that there exists at least one possible solution for \mathcal{G} that can be provided by W , so that $\{\tau\}_G \cap \{\tau\}_W \neq \emptyset$. However, under this degree there are possible solutions for \mathcal{G} that cannot be provided by W , as well as executions of W that do not solve \mathcal{G} . Hence, for W to be usable for a goal instance $GI(\mathcal{G})$, it has to hold that the input binding β defined for $GI(\mathcal{G})$ instantiates \mathcal{G} in a way such that $\{\tau\}_{GI(\mathcal{G})} \subseteq \{\tau\}_{W(\beta)}$.

disjoint($\mathcal{D}_G, \mathcal{D}_W$): $\Omega \models \neg \exists \beta. \phi^{\mathcal{D}_G} \wedge \phi^{\mathcal{D}_W}$

This degree denotes that an execution of W that can satisfy \mathcal{G} does not exist, i.e. $\{\tau\}_G \cap \{\tau\}_W = \emptyset$. In consequence, under this degree W can not be used for solving \mathcal{G} or any of its instantiations.

Table 3 in Appendix A.1 provides a concise compilation of the above definitions. The existence and denomination of the matchmaking degrees can be considered as a the common result of several research efforts (e.g. [23, 19, 11]). However, the underlying models of the world as well as the formal descriptions in these approaches lack of expressivity for accurately describing requested and provided functionalities. We define the matchmaking degrees on formal functional descriptions as defined in Section 3, which provides a sufficiently rich description model for precise Web service discovery. We discuss related work in more detail in Section 6.

4.2 Goal Instance Level Discovery

We now turn towards the usability of a Web service for a goal instance $GI(\mathcal{G})$ that instantiates a goal template \mathcal{G} . As a novel technique that has not been presented in previous works, the following subsequently determines the requirements for semantic matchmaking on the goal instance level and then specifies a matchmaking technique therefore.

4.2.1 Requirements Analysis

As defined above, $GI(\mathcal{G})$ is created by defining an input binding β for the *IF*-variables in \mathcal{D}_G , i.e. the formally described functionality requested in \mathcal{G} (see Definition 2). Hence, the set of possible solutions for $GI(\mathcal{G})$ is a subset of those for \mathcal{G} , i.e. $\{\tau\}_{GI(\mathcal{G})} \subset \{\tau\}_G$ (see Definition 1).

Recalling the foundations of our two-phase discovery framework from Section 2.2, clause (iv) in Definition 1 states that a Web service W that is not usable for a goal template \mathcal{G} is also not usable for any goal instance of \mathcal{G} . Hence, in terms of the matchmaking degrees for Web service discovery on the goal template level, it holds that under the *disjoint* degree a Web service is not usable for any instantiation of \mathcal{G} . For the other four degrees, it holds that the set of usable Web service for a goal instance $GI(\mathcal{G})$ is always a subset of those usable for its goal template \mathcal{G} – which correlates with clause (iii) of Definition 1. Thus, we can use the matchmaking degrees for semantic Web service discovery on the goal template level as a filtering mechanism for determining usable Web services on the goal instance level.

In the course of specifying the matchmaking degrees in Section 4.1, we have shown that the usability of a Web service W is dependent on how the β defined in $GI(\mathcal{G})$ instantiates the goal template \mathcal{G} . Analyzing the above discussion reveals the following condition that is common under all matchmaking degrees: a Web service W is usable for solving a goal instance $GI(\mathcal{G})$ if β instantiates \mathcal{G} in such a way that the possible solutions for $GI(\mathcal{G})$ are allocated in the intersection of the possible solutions for \mathcal{G} and possible executions of W that are triggered by β . Naturally, this correlates with the basic matching condition for the goal instance level (clause (ii) of Definition 1).

In order to specify a semantic technique for evaluating these usability conditions on the basis of the available descriptions, the following first examines the properties of input bindings for goals and Web services and then define matchmaking conditions for Web service discovery on the goal instance level.

4.2.2 Input Bindings

An input binding is a variable assignment for the input variables in a functional description. Definition 2 requires $\beta : \{i_1, \dots, i_n\} \rightarrow \mathcal{U}_A$ as a total function that assigns objects over the universe \mathcal{U}_A to the *IF*-variables. With this, we obtain an assignment of concrete values v for all inputs required in a functional description, i.e. $\beta = \{i_1|v_1, \dots, i_n|v_n\}$. This is needed to evaluate the precondition and effect formulae in a functional description, wherein the *IF*-variables can occur as free variables.

Let us now consider the relationship of an input binding that defines a goal instance $GI(\mathcal{G})$ and the one for invoking the Web service W that is to be used. Obviously, both need to define the same input values – otherwise the triggered execution of W will not solve the goal. However, the input bindings do not have to be literally identical but they need to define the same input values. There might be a permutation or renaming of symbols necessary in order to present the concrete input values defined in $GI(\mathcal{G})$ in the form expected by W . Hence, in the following we distinguish two input bindings: $\beta_{\mathcal{G}}$ as the one defined in a goal instance $GI(\mathcal{G})$, and β_W as the one to invoke W for solving $GI(\mathcal{G})$.

Definition 7. Let $\mathcal{D}_{\mathcal{G}} = (\Sigma_{\mathcal{A}}, \Omega, IF_{\mathcal{G}}, \phi^{pre}, \phi^{eff})$ be the functional description of a goal template \mathcal{G} , and let $\mathcal{D}_W = (\Sigma_{\mathcal{A}}, \Omega, IF_W, \phi^{pre}, \phi^{eff})$ describe a provided functionality. Let W be a Web service with $W \models_{\mathcal{A}} \mathcal{D}_W$. Let a goal instance $GI(\mathcal{G})$ be defined by the input binding $\beta_{\mathcal{G}} : IF_{\mathcal{G}} \rightarrow \mathcal{U}_A$.

We define β_W as the **input binding to invoke W for solving $GI(\mathcal{G})$** such that:

- (i) there is a bijection $\pi : IF_{\mathcal{G}} \rightarrow IF_W$
- (ii) $\beta_W(\pi(i)) = \beta_{\mathcal{G}}(i)$ for all $i \in IF_{\mathcal{G}}$

Clause (i) states that for every *IF*-variable in the goal description there is a corresponding *IF*-variable in the Web service description (and vice versa). This ensures on the description level that a goal instance can provide all inputs required for invoking W -as $W \models_{\mathcal{A}} \mathcal{D}$ requires that there exists a similar bijection between the *IF*-variables in \mathcal{D}_W and the input names required by the Web service (see Definition 3). Clause

(ii) defines that the variable assignments in β_W are the same as defined in β_G . Under the assumption that W is usable for solving $GI(\mathcal{G})$, this ensures that the execution of W that is triggered by β_W will solve $GI(\mathcal{G})$.

The requirement on the compatibility is not trivial to realize in practice. It requires requires a semantic mapping between the input variables of functional descriptions and the Web service. Furthermore, maybe also mediation between incompatible ontologies used by the requester and provider. In general, the establishment of the compatibility requires manual intervention in order to define the semantic mapping between and to resolve potentially occurring data level mismatches. We therefore can apply mediators that connect a goal templates and a Web services and define the necessary mappings. The WSMO framework provides the concept of WG mediators for this purpose [21].

Another aspect is that there must be concrete values for all required inputs in order to invoke Web service. The two bijections - the one between the input variables of the goal and the Web service description in clause (i) of Definition 7, and the one between the Web service description and the Web service in clause (i) of Definition 3 – denote the basic requirement therefore. However, requiring the client to provide concrete values for all inputs is a very restrictive requirement. In some cases, the client may not be able or not willing to provide specific data (e.g. details on banking information); in other cases, the background ontology Ω may define attributes or relations that are not relevant for formulating the client request. To handle both cases, we specify so-called *generic instances* for each required input value that is not explicitly specified by the client. A generic instance defines existence of an instance of a concept with universally quantified variables [27]: if there is a required input i for which β_G does not provide a concrete value assignment, then we create a generic instance of the form $\exists x.\phi_{pre}(i)$ where $\phi_{pre}(i)$ denotes the conditions defined in the precondition of the goal template description. Therewith, we can ensure that there is a concrete value assignment for each input required by W .

4.2.3 Matchmaking For Goal Instances

We now can define the semantic discovery technique for determining the usability of a Web service for solving a goal instance. We therefore need to specific a matchmaking condition that allows to evaluate the basic matching condition for the goal instance level on basis of the available description elements.

Clause (ii) in Definition 1 defines $match(GI(\mathcal{G}), W)$ to be given if $\exists \tau. \tau \in (\{\tau\}_{GI(\mathcal{G})} \cap \{\tau\}_{W(\beta)})$. In terms of the introduced notions, this is satisfied if β_G instantiates \mathcal{G} such that there is at least one possible solution for $GI(\mathcal{G})$ that can be provided by an execution of W that is triggered by β_W .

The approach for determining this on basis of the given description elements is as follows. Formally, an input binding $\beta : \{i_1, \dots, i_n\} \rightarrow \mathcal{U}_{\mathcal{A}}$ is a total function that defines a variable assignment over the universe $\mathcal{U}_{\mathcal{A}}$ 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 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 $\Omega_{\mathcal{A}} \cup \{[\phi^{\mathcal{D}_G}]_{\beta}, [\phi^{\mathcal{D}_W}]_{\beta}\}$ must be satisfiable, i.e. that there exists a Σ -interpretation that is a model for the extended domain knowledge $\Omega_{\mathcal{A}}$ 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 Theorem 1, 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 8. Let $\mathcal{D}_G = (\Sigma_{\mathcal{A}}, \Omega, IF_G, \phi_G^{pre}, \phi_G^{eff})$ be the functional description of a goal template \mathcal{G} that is simulated by $sim(\mathcal{D}_G)$. Let $\mathcal{D}_W = (\Sigma_{\mathcal{A}}, \Omega, IF_W, \phi_W^{pre}, \phi_W^{eff})$ be a functional description simulated by $sim(\mathcal{D}_W)$, and let W be a Web service such that $W \models_{\mathcal{A}} \mathcal{D}_W$. Let the goal instance $GI(\mathcal{G})$ be defined by the input binding β_G , and let β_W be the input binding to invoke W for solving $GI(\mathcal{G})$. Let $\Omega_{\mathcal{A}} = \Omega \cup [\Omega]_{\Sigma_D^{pre} \rightarrow \Sigma_D}$ be the background knowledge extended with the pre-variants of each dynamic symbol $\alpha \in \Sigma_D^{pre}$.

Let there be a bijection $\pi : IF_G \rightarrow IF_W$, and let the $[\phi^{\mathcal{D}_G}]_{\pi}$ be the formula that is derived from $\phi^{\mathcal{D}_G}$ by replacing every occurrence of a IF_G -variable by the corresponding renamed variable $\pi(i)$. Let $[\phi^{\mathcal{D}}]_{\beta}$ be the formula that is derived from $\phi^{\mathcal{D}}$ by substituting every occurrence of an IF -variable by the value assigned in β for all $i \in IF$.

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

$$I \models \Omega_{\mathcal{A}} \quad \text{and} \quad I \models [[\phi^{\mathcal{D}_G}]_{\pi}]_{\beta_W} \quad \text{and} \quad I \models [\phi^{\mathcal{D}_W}]_{\beta_W}.$$

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 3 with the matchmaking condition for the goal instance level. Due to their definition, we can simplify the matching condition from Definition 8 for the distinct matchmaking degrees as follows.

Theorem 2. 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 $W \models_{\mathcal{A}} \mathcal{D}_W$.

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 $\bigwedge \Omega_{\mathcal{A}} \wedge [\phi^{\mathcal{D}_W}]_{\beta}$ is satisfiable, or
- (iv) $intersect(\mathcal{D}_G, \mathcal{D}_W)$ and $\bigwedge \Omega_{\mathcal{A}} \wedge [\phi^{\mathcal{D}_G}]_{\beta} \wedge [\phi^{\mathcal{D}_W}]_{\beta}$ 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 $\Omega_{\mathcal{A}}$. 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. The formal proof of this theorem is provided in Appendix A.3 of this document.

5 Illustrative Example

This section provides a proof of concept for the preceding theoretical elaborations. We therefore exhaustively discuss the following scenario: the goal specifies the objective of finding the best restaurant in a city, and the Web service provides a search facility for the best French restaurant in a city. As we shall show below, this is an example for the *intersect* degree and hence requires the full range of the extended matchmaking for Web service discovery on the goal instance level.

We have implemented and verified the matchmaking techniques in VAMPIRE, a resolution-based theorem prover for classical first-order logic with equality [24] that allows to realize matchmaking exactly as specified in this work and has been successfully applied in previous works [27]. The following first illustrates the modelling of functional descriptions for goals and Web services as defined in Section 3, and then demonstrates the matchmaking techniques for semantic Web service discovery defined in Section 4. Thereafter, we briefly address examples for other matchmaking degrees in the same setting.¹

5.1 Goal and Web Service Description

The following illustrates the specification of the functional descriptions in accordance to Definition 2. We use classical first-order logic (FOL, as defined in [26]) as the specification language. In order to express frame-based modelling of concepts and attributes in FOL, we apply the notation introduced in [14] to model functional descriptions: $\text{memberOf}(x, \text{concept})$ denotes class membership of the variable x to a concept, and $\text{hasAttValue}(x, \text{attribute}, \text{value})$ defines an attribute for the variable x whose value can be either a constant or a variable.

In our example setting, the goal template \mathcal{G} describes the objective of finding the best restaurant in a city. For instantiating \mathcal{G} , the concrete city needs to be specified. In the desired state of the world, the best restaurant in this city shall be provided. Specifying this in terms of a functional description, $\mathcal{D}_{\mathcal{G}}$ specifies one *IF*-variable that is constraint in the precondition ϕ^{pre} to be a member of the concept *city*. The effect ϕ^{eff} describes the desired state of the world to be given if and only if the output provided by the Web service is a restaurant in the city such that there does not exist any better restaurant in the city. Analogously, \mathcal{D}_W describes the functionality provided by the Web service W . The mere difference occurs in the effect because the output of W is a French restaurant in the city provided as input such that there does not exist any better French restaurant in the city.

The signature Σ for both $\mathcal{D}_{\mathcal{G}}$ and \mathcal{D}_W consists of the predicate symbols for class membership and attribute values, and the unary predicate $\text{out}(x)$ used in ϕ^{eff} for denoting the output objects (see Definition 2). FOL serves as the logic $\mathcal{L}(\Sigma)$. The domain knowledge Ω is defined in the *best restaurant ontology* that describes restaurants with attributes `in` and `type`, and cities along with their geographic location. $\text{better}(x, y)$ is a transitive predicate that describes the rating of restaurants as a partial order (i.e. *true* if the rating for restaurant x is higher than for restaurant y , and $\text{better}(x, y) \wedge \text{better}(y, z) \Rightarrow \text{better}(x, z)$). We omit the complete ontology specification with respect to space limitations.

It is to remark that in accordance to Definition 2, the *IF*-variable x occurs as a free variable in the precondition and in the effect of both $\mathcal{D}_{\mathcal{G}}$ and \mathcal{D}_W . Hence, the functional descriptions can only be evaluated if an input binding β is defined. For the sake of simplicity, this example by intent does not encompass static and dynamic symbols. We refer to [12] for an exhaustive discussion and illustration for the signature extensions in the ASS model.

¹The 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>

Goal "find best restaurant in a city"

$$\begin{aligned}
\mathcal{D}_G \quad \Sigma: & \quad \text{memberOf}(x, \text{concept}), \text{hasAttValue}(x, \text{name}, \text{value}), \text{out}(x) \\
\Omega: & \quad \text{best restaurant ontology} \\
IF: & \quad \{x\} \\
\phi^{pre}: & \quad \text{memberOf}(x, \text{city}) \\
\phi^{eff}: & \quad \forall y. \text{out}(y) \Leftrightarrow (\\
& \quad \text{memberOf}(y, \text{restaurant}) \\
& \quad \wedge \text{hasAttValue}(y, \text{in}, x) \\
& \quad \wedge \neg \exists z. (\text{memberOf}(z, \text{restaurant}) \\
& \quad \wedge \text{hasAttValue}(z, \text{in}, x) \\
& \quad \wedge \text{better}(z, y))).
\end{aligned}$$

Web Service "find best French restaurant in a city"

$$\begin{aligned}
\mathcal{D}_W \quad \Sigma: & \quad \text{memberOf}(x, \text{concept}), \text{hasAttValue}(x, \text{name}, \text{value}), \text{out}(x) \\
\Omega: & \quad \text{best restaurant ontology} \\
IF: & \quad \{x\} \\
\phi^{pre}: & \quad \text{memberOf}(x, \text{city}) \\
\phi^{eff}: & \quad \forall y. \text{out}(y) \Leftrightarrow (\\
& \quad \text{memberOf}(y, \text{restaurant}) \\
& \quad \wedge \text{hasAttValue}(y, \text{type}, \text{french}) \\
& \quad \wedge \text{hasAttValue}(y, \text{in}, x) \\
& \quad \wedge \neg \exists z. (\text{memberOf}(z, \text{restaurant}) \\
& \quad \wedge \text{hasAttValue}(z, \text{in}, x) \\
& \quad \wedge \text{hasAttValue}(z, \text{type}, \text{french}) \\
& \quad \wedge \text{better}(z, y))).
\end{aligned}$$

5.2 Semantic Web Service Discovery

We now demonstrate the semantic matchmaking techniques for Web service discovery on the goal template and the goal instance level as defined in Section 4. The main focus of our discussion is the accuracy, i.e. to show that the specified techniques allow to precisely determine the usability of a Web service for a concrete request. For verification, we show the relevant proof obligations from the implementation with VAMPIRE along the following explanations.

For illustration, it is sufficient to consider the following two cities: city A wherein the best restaurant is French and city B wherein the best restaurant is not French. We then consider 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 of them. Table 1 shows the descriptions of the cities A and B that constitute the relevant part of the universe \mathcal{U}_A . Table 2 shows the results of evaluating the functional descriptions \mathcal{D}_G and \mathcal{D}_W from above under the input bindings. As the most important information in the following discussion, we here show the obtainable output values. For β_1 , the result for both \mathcal{D}_G and \mathcal{D}_W is $\text{out}(y)=r1A$, i.e. the best restaurant in A . For β_2 , the result for \mathcal{D}_G is $\text{out}(y)=r1B$, and $\text{out}(y)=r2B$ for \mathcal{D}_W . These denote the only possible Σ -interpretations that satisfy the respective effect constraints under the particular input binding.

Table 1: Σ -Interpretations in Universe relevant for Example

City A	City B
$memberOf(A, city)$ $memberOf(r1A, restaurant)$ $hasAttValue(r1A, in, A)$ $hasAttValue(r1A, type, french)$ $memberOf(r2A, restaurant)$ $hasAttValue(r2A, in, A)$ $hasAttValue(r2A, type, italian)$ $better(r1A, r2A)$ $memberOf(r3A, restaurant)$ $hasAttValue(r3A, in, A)$ $hasAttValue(r3A, type, french)$ $better(r2A, r3A)$	$memberOf(B, city)$ $memberOf(r1B, restaurant)$ $hasAttValue(r1B, in, B)$ $hasAttValue(r1B, type, italian)$ $memberOf(r2B, restaurant)$ $hasAttValue(r2B, in, B)$ $hasAttValue(r2B, type, french)$ $better(r1B, r2B)$ $memberOf(r3B, restaurant)$ $hasAttValue(r3B, in, B)$ $hasAttValue(r3B, type, french)$ $better(r2B, r3B)$

Table 2: Results for $\mathcal{D}_{\mathcal{G}}$, \mathcal{D}_W under Input Bindings

$[sim(\mathcal{D}_{\mathcal{G}})]_{\beta_1}$	$[sim(\mathcal{D}_W)]_{\beta_1}$
$memberOf(A, city) \Rightarrow ($ $out(r1A) \Leftrightarrow ($ $memberOf(r1A, restaurant)$ $\wedge hasAttValue(r1A, in, A)$ $\wedge \neg \exists z.(memberOf(z, restaurant)$ $\wedge hasAttValue(z, in, A)$ $\wedge better(z, r1A)$ $\wedge (z = r2A \vee z = r3A)))$.	$memberOf(A, city) \Rightarrow ($ $out(r1A) \Leftrightarrow ($ $memberOf(r1A, restaurant)$ $\wedge hasAttValue(r1A, in, A)$ $\wedge hasAttValue(r1A, type, french)$ $\wedge \neg (memberOf(r3A, restaurant)$ $\wedge hasAttValue(r3A, in, A)$ $\wedge hasAttValue(r3A, type, french)$ $\wedge better(r3A, r1A)))$.
$[sim(\mathcal{D}_{\mathcal{G}})]_{\beta_2}$	$[sim(\mathcal{D}_W)]_{\beta_2}$
$memberOf(B, city) \Rightarrow ($ $out(r1B) \Leftrightarrow ($ $memberOf(r1B, restaurant)$ $\wedge hasAttValue(r1B, in, B)$ $\wedge \neg \exists z.(memberOf(z, restaurant)$ $\wedge hasAttValue(z, in, B)$ $\wedge better(z, r1B)$ $\wedge (z = r2A \vee z = r3A)))$.	$memberOf(B, city) \Rightarrow ($ $out(r2B) \Leftrightarrow ($ $memberOf(r2B, restaurant)$ $\wedge hasAttValue(r2B, in, B)$ $\wedge hasAttValue(r2B, type, french)$ $\wedge \neg (memberOf(r3B, restaurant)$ $\wedge hasAttValue(r3B, in, B)$ $\wedge hasAttValue(r3B, type, french)$ $\wedge better(r3B, r2B)))$.

Along the way, this illustrates the correlation of input bindings and start states for a $\tau = (s_0, \dots, s_m)$ in an Abstract State Space \mathcal{A} : the world is defined by the objects in the universe $\mathcal{U}_{\mathcal{A}}$ wherefore an input binding β defines a variable assignment. If the world changes – e.g. a new French restaurant is opened in city B that has a better rating than all the existing ones – then the solutions for \mathcal{G} as well as the executions of W would be different for the same input bindings.

5.2.1 Goal Template Level

Let us now consider Web service discovery on the goal template level. From Table 2, we observe that for β_1 there exists a τ that satisfies both \mathcal{D}_G and \mathcal{D}_W and hence denotes a solution for \mathcal{G} that can be provided by W . Thus, the matchmaking condition for $\text{intersect}(\mathcal{D}_G, \mathcal{D}_W)$ is satisfied. For β_2 , the execution of W results in a different end state than the solution for \mathcal{G} . Hence, $\neg\forall\beta. \phi^{\mathcal{D}_G} \Rightarrow \phi^{\mathcal{D}_W}$ and $\neg\forall\beta. \phi^{\mathcal{D}_G} \Leftarrow \phi^{\mathcal{D}_W}$, so that neither the condition for the *plugin* degree nor for the *subsume* degree is satisfied. Thus, the matchmaking degree between \mathcal{D}_G and \mathcal{D}_W is *intersect*.

The following shows the proof of the intersection match within VAMPIRE. This is a resolution-based theorem prover for first-order classical logic with equality [24] that we use for demonstration and proof of correctness throughout the example. For modelling the goal and the Web service descriptions, their functional descriptions are separated into three formula: one that specifies the inputs and preconditions, one for the output and postconditions, and one that defines the relationship between the former two (i.e. the semantics of the functional description). We also need to explicitly define that the *better*(x, y) relation is a partial order. The proof obligation for the intersection match is defined as in Section 4.1: existence of at least one Σ -interpretation for an input-output pair that is a common model of \mathcal{D}_G and \mathcal{D}_W while there is no logical entailment: $\mathcal{D}_G \not\models \mathcal{D}_W$ and $\mathcal{D}_W \not\models \mathcal{D}_G$. For realizing this in VAMPIRE, we use so-called generic instances that have been introduced in [27]: a generic instance defines existence of an instance of a concept with universally quantified variables. As therewith the theorem prover always finds an existing instance for concepts and relations defined in the signature, we can work with incomplete functional descriptions (such as that the goal description in our example does not define restrictions on the restaurant type).²

```
% SIGNATURE
% better-relation is a partial order
input_formula(transitivityBetterRelation, axiom,(
  ! [R1,R2] : (
    memberOf(R1, restaurant) & memberOf(R2, restaurant)
    & better(R1,R2) => ~better(R2,R1) )
)).
% transitivity of better-relation
input_formula(transitivityBetterRelation, axiom,( ! [R1,R2,R3] : (
  memberOf(R1, restaurant) & memberOf(R2, restaurant)
  & memberOf(R3, restaurant) & better(R1,R2) & better(R2,R3)
  => better(R1,R3) )
)).
% GOAL: find best restaurant in a city
input_formula(goalin, axiom,(
  ! [X] : ( goalin(X) <=> (memberOf(X, city) ) ))).
input_formula(goalout, axiom,( ! [X,Y] :
  ( goalout(X,Y) <=> ( memberOf(Y, restaurant) & hasAttValue(Y, in, X)
    & ~ ? [Z] : ( memberOf(Z, restaurant) & hasAttValue(Z, in, X)
    & better(Z,Y)) ) ) ) ).
```

²VAMPIRE supports TPTP, a first-order logic syntax used for automated theorem proving, see homepage: www.tptp.org. For traceability, the most important constructs are quantifiers (universal: !, existential: ?), logical connectives (and: &, or: |, not: ~, implication: =>, equivalence: <=>); variables are denoted by capital letters. FOL formulae are defined as `input-formulae(name, type, ϕ)` with `axiom` denoting a knowledge definition and `conjecture` as a proof obligation.

```

input_formula(goaldescription, axiom,( ! [I,O] : (
  goal(I,O) <=> (goalin(I) => goalout(I,O) ) ) )).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% WEB SERVICE: give best French restaurant in a city
input_formula(wsin, axiom,( ! [X] : (
  wsin(X) <=> ( memberOf(X, city) ) ) )).
input_formula(wsout, axiom,( ! [X,Y,Z] : (
  wsout(X,Y) <=> ( memberOf(Y, restaurant)
    & hasAttValue(Y, in, X) & hasAttValue(Y, type, french)
    & ~ ? [Z] : ( memberOf(Z, restaurant) & hasAttValue(Z, in, X)
      & hasAttValue(Z, type, french) & better(Z,Y) ) ) ) )).
input_formula(wsdescription, axiom,( ! [I,O] : (
  ws(I,O) <=> (wsin(I) => wsout(I,O) ) ) )).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% proof obligation for intersection match G, WS.
input_formula(po, conjecture,( ? [I,O] : ( goal(I,O) & ws(I,O) )
  &~(! [I,O] : ( (goal(I,O) => ws(I,O)) | (ws(I,O) => goal(I,O)) ) ) )).
%% PROVED

```

5.2.2 Goal Instance Level

Because of the *intersect* degree on the goal template level, clause (iv) of Theorem 2 must hold for W to be usable for solving a goal instance $GI(\mathcal{G})$. This requires that for the same input values there must be a τ in \mathcal{A} that is both a solution for $GI(\mathcal{G})$ and a possible execution of W . Following Definition 8, this is given if there exists a Σ -interpretation I over $\mathcal{U}_{\mathcal{A}}$ that is a model for both $sim(\mathcal{D}_{\mathcal{G}})$ and $sim(\mathcal{D}_W)$ when they are instantiated by substituting the *IF*-variables with the value assignments in β . Let us consider the following two goal instances: $GI(\mathcal{G})_1$ that defines β_1 for instantiating $\mathcal{D}_{\mathcal{G}}$, and $GI(\mathcal{G})_2$ that defines β_2 .

- For $GI(\mathcal{G})_1$, there is a Σ -interpretation I_1 with the variable assignment $x = A, y = r1A, z = r3A$ satisfies both $[\phi^{\mathcal{D}_{\mathcal{G}}}]_{\beta_1}$ and $[\phi^{\mathcal{D}_W}]_{\beta_1}$; thus, W is usable for solving $GI(\mathcal{G})_1$.
- For $GI(\mathcal{G})_2$, the Σ -interpretations that satisfy $[\phi^{\mathcal{D}_{\mathcal{G}}}]_{\beta_2}$ are I_2 with $y = r1B, z = r2B$ or I_3 with $y = r1B, z = r3B$. The only Σ -interpretation that satisfies $[\phi^{\mathcal{D}_W}]_{\beta_2}$ is I_4 with the variable assignment $y = r2B, z = r3B$. As $I_2 \neq I_4$ and $I_3 \neq I_4$, W is not usable for solving $GI(\mathcal{G})_2$.

The following shows the proof obligation in from VAMPIRE. We therefore define the input bindings by the respective facts as shown in Table 1, and define the matchmaking condition for the goal instance level as defined in Definition 8. The goal template and Web service description are the same as in the listing above.

```

% Universe for City A (input binding 1)
input_formula(cityA, axiom,(memberOf(A,city))). input_formula(r1A,
axiom,(memberOf(r1A,restaurant) &
  hasAttValue(r1A, in, A) &
  hasAttValue(r1A, type, french) )).
input_formula(r2A, axiom,(memberOf(r2A,restaurant) &

```



```

    hasAttValue(r2A, in, A) &
    hasAttValue(r2A, type, italian) )).
input_formula(r3A, axiom,(memberOf(r3A,restaurant) &
    hasAttValue(r3A, in, A) &
    hasAttValue(r3A, type, french) )).

% proof obligation for goal instance level match.
input_formula(po, conjecture,( ? [cA,O] : ( goal(cA,O) & ws(cA,O) ).
% PROVED

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Universe for City B (input binding 2)
input_formula(cityB, axiom,(memberOf(B,city))). input_formula(r1B,
axiom,(memberOf(r1B,restaurant) &
    hasAttValue(r1B, in, B) &
    hasAttValue(r1B, type, italian) )).
input_formula(r2B, axiom,(memberOf(r2B,restaurant) &
    hasAttValue(r2B, in, B) &
    hasAttValue(r2B, type, french) )).
input_formula(r3B, axiom,(memberOf(r3B,restaurant) &
    hasAttValue(r3B, in, B) &
    hasAttValue(r3B, type, french) )).

% proof obligation for goal instance level match.
input_formula(po, conjecture,( ? [cB,O] : ( goal(cB,O) & ws(cB,O) ).
% NOT PROVED

```

5.3 Web Service Discovery for Goal Instances under other Matching Degrees

For demonstrating Web service discovery on the goal instance level under the other matchmaking degrees, let us consider a Web service W_2 that provides a lookup functionality for the best restaurants in Austrian cities. \mathcal{D}_{W_2} defines $\phi^{pre} = memberOf(x, city) \wedge hasAttValue(x, in, austria)$, and the same effect as in the goal description above. As an example for the *subsume* degree, let $GI(\mathcal{G})_3$ be a goal instance that defines $\beta_{W_2} = \{x|Berlin\}$, i.e. the German capital. Clause (iii) in Theorem 2 defines that $[\phi^{\mathcal{D}_{W_2}}]_{\beta_{W_2}}$ needs to hold for W_2 to be usable for solving a goal instance. β_{W_2} does not satisfy the precondition of \mathcal{D}_{W_2} so that we can not determine a Σ -interpretation that satisfies $[\phi^{\mathcal{D}_{W_2}}]_{\beta_{W_2}}$, hence W_2 is not usable for $GI(\mathcal{G})_3$.

For the *plugin* degree, let us consider another goal template \mathcal{G}_2 for finding the best restaurant in a city in Tyrol (a state of Austria). $\mathcal{D}_{\mathcal{G}_2}$ defines $\phi^{pre} = memberOf(x, city) \wedge hasAttValue(x, in, tyrol)$, and the background ontology Ω defines that Tyrol is located in Austria. Any goal instance $GI(\mathcal{G}_2)$ must define an input binding $\beta_{\mathcal{G}_2}$ with a city that is located in Tyrol. With respect to Ω , W_2 can be invoked with a complement of $\beta_{\mathcal{G}_2}$, and the triggered execution will provide the best restaurant in the city. The same holds for a Web service W_3 with $\mathcal{D}_{W_3} \equiv \mathcal{D}_{\mathcal{G}}$. This demonstrates clauses (ii) and (i) of Theorem 2.

6 Related Work

We are not aware of any other approach for semantically enabled Web service discovery that defines accurate matchmaking techniques in a sufficiently rich description framework for requested and provided functionalities. However, there are several works that are related to or have influenced the presented approach. The following discusses them and positions our work with respect to the following aspects: the usage and description of goals for Web services, the formal semantics of prominent frameworks for Semantic Web services, and existing approaches for semantically enabled Web service discovery.

The conception of goals as formalized client requests in our approach has been adopted from technologies for automated problem solving that have been developed in different AI disciplines, such as cognitive architectures [22] or BDI architectures for intelligent agents [5]. As analyzed in [28], therein goals denote client requests for moving from the current state of the world to a state wherein some objective is satisfied. The distinction of goal templates and goal instances in our approach as well as their description as requested functionalities has been inspired by the conception of generic tasks in the UPML framework [9].

The idea of goal-driven architectures re-emerges in the context of Web services, wherein the distinction of service providers and requesters becomes a central aspect of technology design [4]. However, the only framework for Semantic Web service that identifies goals as a top level element is WSMO [8]. While the WSMO specification itself does not distinguish goal templates and goal instances, related system implementations such as IRS [6] or SWF [29] define similar notions in order to achieve a better scalability and ease of use. In contrast to [16], 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 not accessible to the interaction partners.

In the introduction, we have claimed that most frameworks for Semantic Web services do not define an unambiguous formal semantics which hampers the specification of accurate matchmaking techniques. As the most prominent ones, let us briefly examine the four approaches submitted to be W3C as standardization proposals. Chronologically the first, OWL-S [20] defines an upper ontology for annotating Web services. Allocated in the Service Profile, the overall provided functionality is described in terms of inputs, outputs, preconditions, and effects (short: IOPE) with OWL has the specification language. This description model has several drawbacks as criticized in [17].

The most important one in comparison to the ASS model used in our approach is that the correlation between pre-states and post-states of a Web service can not be expressed, as each IOPE is a closed logical formula. The same holds for the WSDL-S that proposes to annotate WSDL descriptions with preconditions and effects [1]. This is overcome in WSMO [18] wherein provided and requested functionalities are described by capabilities that consist of preconditions, assumptions, postconditions, and effects (short: PAPE). To specify the correlation and dependencies of these elements, so-called *shared variables* are defined whose scope is the complete capability – similar to *IF*-variables in the ASS model. However, the formal distinction of the four description elements remains to be unclear.

The second aspect to be discussed here is the underlying model of the world wherein Web services act. OWL-S understands Web services as atomic or composite processes; replacing the initial OWL-S process model, SWSF [3] defines a formally sound process description model and language. In contrast, WSMO as well as our approach considers a state-based model wherein Web services perform transitions between states. The reason therefore is that we understand Web services as passive computational facilities that are dynamically detected and combined in a specific problem solving context – which can be independent from Web services. The primary property of such a facility is to perform a (deterministic) transition, wherefore a state-based model with functional descriptions in terms of preconditions and effects as widely used in

several AI technologies.

As the final aspect of related to be discussed, we examine existing approach for semantically enabled Web service discovery with special attention to the accuracy as the central focus of this paper. As the earliest works, [23] and [19] present matchmaking on OWL-S service profiles and introduce the matchmaking degrees that we defined for discovery on the goal template level discovery. Both works define discovery in terms of concept subsumption in a Description Logic framework. This merely allows to determine semantic relationships between requested and provided concepts, but does not properly reason about functionalities.

The approach for Web service discovery in WSMO is presented in [11]. The matchmaking degrees are defined as set-theoretic criteria that is not bound to a particular formal description of requested and provided functionalities. Different types of matches are distinguished for the usability of a Web service for a goal description. In particular, the *partial match* denotes that the usability of a Web service cannot be definitely determined. The matchmaking technique for the goal instance level presented in this work allows to precisely do this. The only approach known to us that performs discovery with concrete inputs is presented in [15]: the inputs defined in the goal are inserted into the internal knowledge base, and then it is checked whether the hypothetical execution result allows to solve the goal. However, this technique performs hypothetical executions of Web services instead of matchmaking of declarative descriptions.

7 Conclusions

This paper has presented an approach for accurate semantically enabled Web service discovery that works on a sufficiently rich descriptions of the requested and provided functionalities.

We have introduced the notions of goal templates and goal instances. The former are generic descriptions of client objectives whereof the latter are created for expressing a concrete request by defining concrete input values. Based on the formal relationship, we have defined a two-phase discovery process wherein discovery on the goal template level is performed at design time and then serves as a filter mechanism for discovery on the goal instance level at runtime.

We apply a state-based model of the world wherein Web service perform state transitions. To properly describe the functionality provided by a Web service as well as the one requested in a goal description, we define functional descriptions that formally describe sequences of state transitions with respect to their start- and end-state. In order to define logical relationships and operations on functional descriptions of goals and Web services, we have presented a first-order logic structure that allows to simulate functional descriptions as conventional formulae.

On the basis of this, we have presented semantic matchmaking techniques that allow to precisely determine the usability of a Web service for solving a goal. For the goal template level, we have adjusted the matchmaking degrees commonly identified for Web service discovery. In our framework, they denote the relationship of the set of possible solutions for a goal and possible executions of a Web service. For the goal instance level, we have presented a novel matchmaking technique that allows to precisely determine the usability of a Web service for concrete requests. On basis of the available description elements, this determines the existence of a solution for a goal instance that can be provided by Web service when it is invoked with the inputs defined in the goal instance. Finally, we have integrated the matchmaking techniques on both levels and demonstrated them in an extensive example.

In conclusion, we have provided a comprehensive framework for semantic Web service discovery. Defined in form of rigorous formal definitions, it is independent of the specification language used. Therewith, this paper provides a generic specification for accurate Web service discovery that can be adopted to several frameworks that deal with requested and provided functionalities.

References

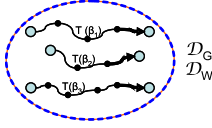
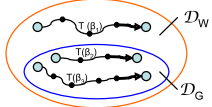
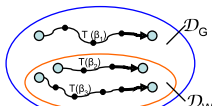
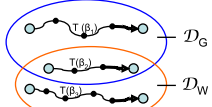
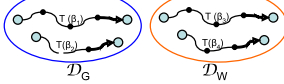
- [1] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M.-T. Schmidt, A. Sheth, and K. Verma. Web Service Semantics - WSDL-S. W3C Member Submission 7 November 2005, 2005. online: <http://www.w3.org/Submission/WSDL-S/>.
- [2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Data-Centric Systems and Applications. Springer, Berlin, Heidelberg, 2004.
- [3] S. Battle, A. Bernstein, H. Boley, B. Grosz, M. Gruninger, R. Hull, M. Kifer, Martin. D., McIlraith. S., D. McGuinness, J. Su, and S. Tabet. Semantic Web Services Framework (SWSF). W3C Member Submission 9 September 2005, 2005. online: <http://www.w3.org/Submission/SWSF/>.
- [4] D. Booth, H. Haas, F. McCabe, E. Newcomer, I. M. Champion, C. Ferris, and Orchard. D. Web Services Architecture. Working group note 11 february 2004, W3C, 2006. online at: <http://www.w3.org/TR/ws-arch/>.
- [5] M. E. Bratman. *Intention, Plans and Practical Reason*. Harvard University Press, Cambridge, MA (USA), 1987.
- [6] 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 *In Proc. of the 5th International Semantic Web Conference (ISWC 2006), Athens(GA), USA, 2006*.
- [7] T Erl. *Service-Oriented Architecture SOA. Concepts, Technology, and Design*. Prentice Hall PTR, 2005.
- [8] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domigue. *Enabling Semantic Web Services. The Web Service Modeling Ontology*. Springer, 2006.
- [9] D. Fensel et al. The Unified Problem Solving Method Development Language UPML. *Knowledge and Information Systems Journal (KAIS)*, 5(1), 2003.
- [10] C. A. R. Hoare. An Axiomatic Basis for Computer Programming. *Communications of the ACM*, 12(10):576–580, 1969.
- [11] 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.
- [12] U. Keller and H. Lausen. Functional Description of Web Services. Deliverable D28.1, WSML Working Group, 2006. Most recent version available at: <http://www.wsmo.org/TR/d28/d28.1/>.
- [13] U. Keller, H. Lausen, and M. Stollberg. On the Semantics of Functional Descriptions of Web Services. In *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006), Montenegro, 2006*.
- [14] G. Kifer, M. and Lausen and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *JACM*, 42(4):741–843, 1995.

- [15] M. Kifer, R. Lara, A. Polleres, C. Zhao, U. Keller, H. Lausen, and D. Fensel. A Logical Framework for Web Service Discovery. In *Proc. of the ISWC 2004 workshop on Semantic Web Services: Preparing to Meet the World of Business Applications, Hiroshima, Japan, Nov. 2004*, 2004.
- [16] R. Lara, M. A. Corella, and P. Castells. A Flexible Model for Web Service Discovery. In *Proc. of the 1st International Workshop on Semantic Matchmaking and Resource Retrieval: Issues and Perspectives, Seoul, South Korea, 2006*.
- [17] R. Lara, D. Roman, A. Polleres, and D. Fensel. A Conceptual Comparison of WSMO and OWL-S. In *Proc. of the European Conf. on Web Services, 2004*.
- [18] H. Lausen, A. Polleres, and D. Roman (eds.). Web Service Modeling Ontology (WSMO). W3C Member Submission 3 June 2005, 2005. online: <http://www.w3.org/Submission/WSMO/>.
- [19] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the 12th International Conference on the World Wide Web, Budapest, Hungary, 2003*.
- [20] D. Martin. OWL-S: Semantic Markup for Web Services. W3C Member Submission 22 November 2004, 2004. online: <http://www.w3.org/Submission/OWL-S/>.
- [21] A. Mocan, E. Cimpian, M. Stollberg, F. Scharffe, and J. Scicluna. WSMO Mediators. WSMO deliverable D29 final draft 21 Dec 2005, 2005. available at: <http://www.wsmo.org/TR/d29/>.
- [22] A. Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA (USA), 1990.
- [23] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference, Springer, 2002*.
- [24] A. Riazanov and A. Voronkov. The Design and Implementation of VAMPIRE. *AI Communications*, 15(2):91–110, 2002. Special Issue on CASC.
- [25] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 2 edition, 2005.
- [26] R. M. Smullyan. *First Order Logic*. Springer, 1968.
- [27] M. Stollberg, U. Keller, and D. Fensel. Partner and Service Discovery for Collaboration Establishment on the Semantic Web. In *Proceedings of the Third International Conference on Web Services, Orlando, Florida, 2005*.
- [28] M. Stollberg and F. Rhomberg. Survey on Goal-driven Architectures. Technical Report DERI-TR-2006-06-04, DERI, 2006.
- [29] 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, 2005*.

A Appendix

A.1 Overview of Matchmaking Degrees For $\mathcal{D}_G, \mathcal{D}_W$

Table 3: Definition of Matchmaking Degrees for $\mathcal{D}_G, \mathcal{D}_W$

Denotation & Visualization for $\mathcal{D}_G, \mathcal{D}_W$	Definition for: $sim(\mathcal{D}_G) \simeq \mathcal{D}_G$ $sim(\mathcal{D}_W) \simeq \mathcal{D}_W$	Meaning for $\{\tau\}_G, \{\tau\}_W$ with $W \models_{\mathcal{A}} \mathcal{D}$
exact ($\mathcal{D}_G, \mathcal{D}_W$) 	$sim(\mathcal{D}_G) \equiv sim(\mathcal{D}_W)$, i.e. $\Omega \models \forall \beta. \phi^{\mathcal{D}_G} \Leftrightarrow \phi^{\mathcal{D}_W}$.	$\{\tau\}_G = \{\tau\}_W$ and $\tau \in \{\tau\}_{G(\beta)} \Leftrightarrow \tau \in \{\tau\}_{W(\beta)}$
plugin ($\mathcal{D}_G, \mathcal{D}_W$) 	$sim(\mathcal{D}_G) \models sim(\mathcal{D}_W)$, i.e. $\Omega \models \forall \beta. \phi^{\mathcal{D}_G} \Rightarrow \phi^{\mathcal{D}_W}$.	$\{\tau\}_G \subseteq \{\tau\}_W$ and $\tau \in \{\tau\}_{G(\beta)} \Rightarrow \tau \in \{\tau\}_{W(\beta)}$
subsume ($\mathcal{D}_G, \mathcal{D}_W$) 	$sim(\mathcal{D}_W) \models sim(\mathcal{D}_G)$, i.e. $\Omega \models \forall \beta. \phi^{\mathcal{D}_G} \Leftarrow \phi^{\mathcal{D}_W}$.	$\{\tau\}_G \supseteq \{\tau\}_W$ and $\tau \in \{\tau\}_{G(\beta)} \Leftarrow \tau \in \{\tau\}_{W(\beta)}$
intersect ($\mathcal{D}_G, \mathcal{D}_W$) 	$\Omega \models \exists \beta. \phi^{\mathcal{D}_G} \wedge \phi^{\mathcal{D}_W}$ $\wedge \neg(\forall \beta. (\phi^{\mathcal{D}_G} \Rightarrow \phi^{\mathcal{D}_W})$ $\vee (\phi^{\mathcal{D}_G} \Leftarrow \phi^{\mathcal{D}_W}))$.	$\{\tau\}_G \cap \{\tau\}_W \neq \emptyset$ and there exists a β such that $\tau \in \{\tau\}_{G(\beta)}$ and $\tau \in \{\tau\}_{W(\beta)}$
disjoint ($\mathcal{D}_G, \mathcal{D}_W$) 	$\Omega \models \neg \exists \beta. \phi^{\mathcal{D}_G} \wedge \phi^{\mathcal{D}_W}$.	$\{\tau\}_G \cap \{\tau\}_W = \emptyset$ such that there does not exist a $\tau \in \{\tau\}_G$ and $\tau \in \{\tau\}_W$

A.2 Proof for Theorem 1: Simulation of Functional Description as FOL Structure

Proof. We need to show that clauses (i) and (ii) of Definition 5 hold for \mathcal{D} and $\text{sim}(\mathcal{D})$. In essence, these clauses require an equivalence relation between a Web service $W \models_{\mathcal{A}} \mathcal{D}$ and the models of $\text{sim}(\mathcal{D})$: for all executions $\tau(\beta) = (s_0, \dots, s_n)$ of W the corresponding interpretation $\mathcal{I}_W(\beta)$ that simulates the termination state s_n must be a model of $\text{sim}(\mathcal{D})$, and vice versa. Let us consider a Web service W such that $W \models_{\mathcal{A}} \mathcal{D}$, and three different input bindings $\beta : (i_1, \dots, i_n) \rightarrow \mathcal{U}_{\mathcal{A}}$ that cover all relevant cases:

- (1) $\beta_1 \models \phi^{pre}$ and $\beta_1 \models \phi^{eff}$.
- (2) $\beta_2 \not\models \phi^{pre}$.
- (3) $\beta_3 \models \phi^{pre}$ and $\beta_3 \not\models \phi^{eff}$.

As clause (iii) of Definition 6 is merely a symbol substitution, it holds that if $\beta \models \phi^{pre}$ then also $\beta \models [\phi^{pre}]_{\Sigma_D^{pre} \rightarrow \Sigma_D}$. In case (1), for all $\Sigma_{\mathcal{A}}$ -interpretations $\mathcal{I}_W(\beta_1) = (\mathcal{U}_{\mathcal{A}}, I_W(\beta_1))$ it trivially holds that $\mathcal{I}_W(\beta_1) \models \text{sim}(\mathcal{D})$. For $\tau_W(\beta_1) = (s_0, \dots, s_n)$ as the execution of W for β_1 , the meaning of \mathcal{D} is that the termination state s_n will be reached because of $\omega(s_0) \models \phi^{pre}$ (see Definition 2). It trivially holds that $\tau_{W_1}(\beta_1) \simeq \mathcal{I}_W(\beta_1)$, as \mathcal{D} and $\text{sim}(\mathcal{D})$ are both defined over $\Sigma_{\mathcal{A}}$ and use the same precondition and effect formula along with the substitution for dynamic symbols. Hence, $\mathcal{D} \simeq \text{sim}(\mathcal{D})$ is given for this case.

In case (2), for all $\mathcal{I}_W(\beta_2) = (\mathcal{U}_{\mathcal{A}}, I_W(\beta_2))$ it holds that $\mathcal{I}_W(\beta_2) \not\models \text{sim}(\mathcal{D})$ because *false* implies anything. However, the definition of \mathcal{D} only allows to make a concrete statement about the execution of W in the positive case, i.e. when $\omega(s_0) \models \phi^{pre}$. As this is not given in this case, the termination state s_n of execution of W for this input binding $\tau_W(\beta_2) = (s_0, \dots, s_n)$ can be any state – either such that $\mathcal{I}_{\tau_W}^1(\beta_2) \models \phi^{eff}$ or such that $\mathcal{I}_{\tau_W}^2(\beta_2) \not\models \phi^{eff}$. This correlates with the possibilities for $\mathcal{I}_W(\beta_2) \models \text{sim}(\mathcal{D})$, and for both possibilities, it trivially holds that $\tau_W(\beta_2) \simeq \mathcal{I}_W(\beta_2)$. Hence, under the conceptual assumption that a not satisfied precondition does not allow to make any concrete statement about the behavior of a Web service, $\mathcal{D} \simeq \text{sim}(\mathcal{D})$ is given for this case as well.

In case (3), it unambiguously holds that for all $\mathcal{I}_W(\beta_3) = (\mathcal{U}_{\mathcal{A}}, I_W(\beta_3))$ holds $\mathcal{I}_W(\beta_3) \not\models \text{sim}(\mathcal{D})$. The definition \mathcal{D} requires that if $\omega(s_0) \models \phi^{pre}$ then the execution of W results in a state s_n with $\omega(s_n) \models \phi^{pre}$. However, this is only given if \mathcal{D} is modelled correctly: if there is an input binding such that the precondition is satisfiable but the effect is not (or the other way around), then there can not be any Web service that provides \mathcal{D} . Adapting the notion of satisfiability in logic, [13] refer to this as the *realizability* of functional descriptions. As there can not be any Web service that provides a not realizable functional description, there cannot be any $\tau_W(\beta_3)$ so that $\mathcal{D} \simeq \text{sim}(\mathcal{D})$ is given for this case as well.

This completes the proof. □

A.3 Proof for Theorem 2: Integrated Matchmaking for 2-Phased Web Service Discovery

Proof. We commence with the usability of a Web service for a goal instance under the *intersect* degree. This requires the complete matching on the goal instance level from Definition 8.

The other matchmaking degrees that satisfy the basal matching condition in clause (i) of Definition 1 denote specializations of the *intersect* as – due to their definitions in Section 4.1 – it holds: $\neg \text{disjoint} \Rightarrow \text{intersect}$, $\text{intersect} \Rightarrow \text{subsume}$, $\text{intersect} \Rightarrow \text{plugin}$, and $\text{plugin} \wedge \text{subsume} \Rightarrow \text{exact}$. We further recall the following definitions: β_G is the input binding for \mathcal{D}_G that is defined in $GI(\mathcal{G})$, and β_W is the input binding for \mathcal{D}_W that is used for invoking W and whose input values are semantically equivalent to the ones defined in β_G (cf Definition 7). $[\phi^{\mathcal{D}}]_{\beta}$ is the contextualized functional description obtained from substituting all occurrences of *IF*-variables by the value assignments in β , and $[\phi^{\mathcal{D}}]_{\tau}$ is a symbol renaming of the *IF*-variables in $\phi^{\mathcal{D}}$ (cf Definition 8).

$\text{intersect}(\mathcal{D}_G, \mathcal{D}_W)$ is defined as $\Omega \models \exists \beta. \phi^{\mathcal{D}_G} \wedge \phi^{\mathcal{D}_W}$. So, there is at least one $\tau_1 \in (\{\tau\}_G \cap \{\tau\}_W)$ but there can also be a $\tau_2 \in \{\tau\}_G$ but $\tau_2 \notin \{\tau\}_W$ as well as a $\tau_3 \in \{\tau\}_W$ but $\tau_3 \notin \{\tau\}_G$. For a τ to be a solution for a goal instance $GI(\mathcal{G})$ that instantiates the goal template \mathcal{G} with β_G such that this τ can be provided by W when invoked with β_W , the τ needs to be in the intersection of $\{\tau\}_G$ and $\{\tau\}_W$ under the input binding β_G and its compatible counterpart β_W . If it there exists a Σ -interpretation I that is a common model for the instantiated functional descriptions $[\phi^{\mathcal{D}_G}]_{\beta_G}$ and $[\phi^{\mathcal{D}_W}]_{\beta_W}$ under the extended domain knowledge Ω_A , then this I simulates a τ that is a solution for $GI(\mathcal{G})$ and can be provided by W when invoked with β_W (cf Theorem 1). If such a common model does not exist, then there does not exist any $\tau \in (\{\tau\}_{G(\beta_G)} \cap \{\tau\}_{W(\beta_W)})$ (cf Definition 8), and – because of $\neg \text{match}(\mathcal{G}, W) \Rightarrow \neg \text{match}(GI(\mathcal{G}), W)$ – W is not usable for solving $GI(\mathcal{G})$ (cf Definition 1). This shows clause (iv) of Theorem 2.

We now show clause (iii). The $\text{subsume}(\mathcal{D}_G, \mathcal{D}_W)$ degree is defined as $\Omega \models \forall \beta. \phi^{\mathcal{D}_G} \Leftarrow \phi^{\mathcal{D}_W}$ so that $\{\tau\}_G \supseteq \{\tau\}_W$ and for all input bindings β , if $\tau \in \{\tau\}_{W(\beta)}$ then $\tau \in \{\tau\}_{G(\beta)}$ (cf Table 3). If $[\phi^{\mathcal{D}_W}]_{\beta_W}$ is not satisfied, then there does not exist any possible execution of W for the input binding defined in $GI(\mathcal{G})$. If $[\phi^{\mathcal{D}_W}]_{\beta_W}$ is satisfied, then, due to the degree definition $\langle 1 \rangle$ also $[\phi^{\mathcal{D}_G}]_{\beta_G}$ is satisfied so that $\{\tau\}_{W(\beta_W)} \subseteq \{\tau\}_G$, and $\langle 2 \rangle$ the provided functionality is instantiated with the same concrete input values as defined in $GI(\mathcal{G})$ such that the invocation of W with β_W will provide a solution for $GI(\mathcal{G})$.

For clause (ii), $\text{plugin}(\mathcal{D}_G, \mathcal{D}_W)$ is defined as $\Omega \models \forall \beta. \phi^{\mathcal{D}_G} \Rightarrow \phi^{\mathcal{D}_W}$ so that $\{\tau\}_G \subseteq \{\tau\}_W$ and $\tau \in \{\tau\}_{G(\beta)} \Rightarrow \tau \in \{\tau\}_{W(\beta)}$ (cf Table 3). Here, W is usable for every possible goal instance $GI(\mathcal{G})$ of \mathcal{G} because $\langle 3 \rangle \{\tau\}_{GI(\mathcal{G})} \subset \{\tau\}_G \subseteq \{\tau\}_W$ and for each input binding if $\tau \in \{\tau\}_{G(\beta)}$ then $\tau \in \{\tau\}_{W(\beta)}$, and $\langle 4 \rangle$ via β_W , W is invoked with the inputs defined in β_G so that the execution will provide a solution for $GI(\mathcal{G})$. Hence, under the *plugin* degree we do not need to perform an additional matchmaking step for determining the usability of W for solving a goal instance $GI(\mathcal{G})$. As a specialization of the *plugin* degree, the $\text{exact}(\mathcal{D}_G, \mathcal{D}_W)$ degree is defined by $\Omega \models \forall \beta. \phi^{\mathcal{D}_G} \Leftrightarrow \phi^{\mathcal{D}_W}$ so that $\{\tau\}_G = \{\tau\}_W$. Hence, $\langle 3 \rangle$ and $\langle 4 \rangle$ also prove clause (i) of Theorem 2.

Finally, it holds that clauses (i) - (iv) define all possible situations wherein W is usable for solving $GI(\mathcal{G})$, because by definition under $\text{disjoint}(\mathcal{D}_G, \mathcal{D}_W)$ there does not exist any execution of W that can satisfy any goal instance of \mathcal{G} (cf Table 3), and $\neg \text{match}(\mathcal{G}, W) \Rightarrow \neg \text{match}(GI(\mathcal{G}), W)$ (cf Definition 1).

This completes the proof. \square