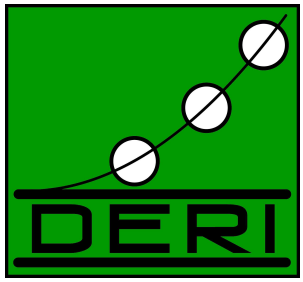

DELTA RELATIONS

Michael Stollberg Uwe Keller

DERI TECHNICAL REPORT 2006-08-18

AUGUST 2006

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

DERI TECHNICAL REPORT
DERI TECHNICAL REPORT 2006-08-18, AUGUST 2006

DELTA RELATIONS: SEMANTIC DIFFERENCE OF FUNCTIONAL DESCRIPTIONS

Michael Stollberg[1] Uwe Keller[1]

Abstract. Matchmaking degrees wherein the functionality provided by a Web service and the one requested by a goal do not match precisely imply certain conditions or impacts for using a specific Web service for solving a goal. This paper presents so-called *delta relations*, or short Δ-relations, as a means for explicating such conditions. This allows to provide users with additional information on how to invoke a Web service such that its execution will resolve a goal, respectively the impacts on how the goal will be resolved when using the Web service. A Δ-relation therefore explicates the semantic difference between functional descriptions with respect to the context of Web service detection. In this paper, we identify the characteristics and requirements for these, provide language-independent formal definitions along with techniques for constructing Δ-relations, and demonstrate their beneficial application within illustrative examples. Besides, we discuss the alignment with semantic matchmaking and the integration of Δ-relations into the WSMO mediation framework.

Keywords: Functional Descriptions, Goals, Web Services, Semantic Difference, Functional Relationship, Delta Relations

[1]Digital 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 Stijn Heymans for proof reading and constructive feedback for this report.

Contents

1 Introduction

This report introduces and defines so-called *delta relations* (or shorter Δ-relations) as a means for explicating the conditions under which a Web service is usable for solving a goal. Such conditions occur when the functionality provided by a Web service does not precisely match with the one requested by a goal but there still are specific executions of the Web service that allow to solve the goal. We discuss the properties of Δ-relations and provide a concise formal definition along with illustrative examples for demonstration and clarification purpose.

The central aspects addressed in this report are:

- the **motivation and application purpose of Δ-relations** with respect to formalized functional descriptions of goals and Web services and semantically enabled mechanisms for determining the usability of Web services for a given goal
- the concept of the **semantic difference between logical formulae** as the basis for denoting the aspirated logical relationship
- extending the definitions and techniques for Δ-relations on logical formulae towards **formal functional descriptions of goals and Web services** in terms of pre- and postconditions
- **clarification and demonstration** of the concepts and definitions by illustrative examples
- **discussion and positioning** of the approach within related work, especially with respect to techniques for "assumption explication"developed in the course of UPML.

The document is structured as follows: the remainder of this section outlines the motivation and aim of Δ-relations and identifies their central characteristics. Section 2 elaborates the formal foundations by defining Δ-relations for conventional logical expressions that represent functionalities requested by goals and provided by Web services. These are extended in Section 3 towards formal functional descriptions that are defined in terms of preconditions and effects within a state-based model of the world. Section 4 discusses related work and positions our approach therein, and Section 5 summarizes and concludes the document.

1.1 Motivation and Aim

The problem context of Δ-relations is detection of usable Web services for solving a given request, commonly referred to as discovery and composition within the field of Semantic Web services. As the core technique, research on semantic matchmaking has identified different degrees under which a Web service is considered to be usable for solve a goal (e.g., [21, 26, 16]). In all situations where the functionality offered by a Web service provider does not exactly match with the one demanded by requester for solving his goal, certain conditions arise under which the Web service is usable for successfully solving the gaol. In particular, for certain matchmaking degrees the client has to provide specific inputs to the Web service such that its execution will provide a result that satisfies the goal; for other degrees, using the Web service for solving a given goal has certain impacts on how the goal will be resolved.

Let's consider an example for clarification. Imagine a generic goal formulation G for purchasing travel tickets in Austria that is instantiated for a concrete client request G_1 of buying a ticket from Innsbruck to Vienna on a certain date. Among all available Web services there is WS_1 for purchasing tickets for all train

connections offered by the Austrian national railway operator ÖBB; another Web service WS_2 offers flight ticket purchasing for Austrian Airlines AUA. Both the ÖBB as well as AUA offer connections between Innsbruck and Vienna. The matchmaking degree between G and WS_1 is that G is more general, i.e. all executions of WS_1 can satisfy all instantiations of G but there also are other solutions that are not provided by WS_1. In this situation, the Web service can be used without any constraints. However, its usage will result in purchasing a train ticket, hence there is a certain impact on how G_1 is resolved when using WS_1.

For WS_2, the national AUA flight connections provided by AUA can satisfy G in case that the origin and destination city have an associated airport; however, international AUA flight connections cannot be used. This situation – commonly referred to as the intersection match in literature – requires that the client provides specific input such that the execution of WS_2 will provide results that satisfy G. In this case, the accurate input is restricted to Austrian cities with an associated airport. This is what we refer to as conditions for usage of a Web service for solving a goal.

The purpose of Δ-relations is to explicate such impacts and usage conditions on Web services for solving a particular goal in order to provide advanced support for Web service detection and usage. Impacts and conditions for Web service usage as illustrated result from differences between the requested and provided functionalities. In frameworks with formal descriptions of goals and Web services, these occur as the semantic difference between functional descriptions. The aim of Δ-relations as introduced and defined in this document is to provide a means for explicating these differences so that it can be *formally expressed* and *explicitly represented*. This additional knowledge can then be *provided to applications* that deal with not precisely matching requested and provided functional descriptions. The following outlines such applications and identifies the consequential characteristics of Δ-relations.

1.2 Some Usage Scenarios for Delta Relations

In accordance to [15], we consider the following procedure for Web service detection. A generic goal description formalizes objectives to be achieved in terms of preconditions and effect descriptions that are mathematically identical to functional descriptions of Web services. A client instantiates such a goal formulation by providing concrete input data that satisfy the goal precondition; these data serve as the inputs for invoking a usable Web service [33]. Discovery or composition with respect to the requested and provided functionalities is performed on the formally described requested and provided functionalities of goals and Web services, i.e. orthogonal to goal instantiation for concrete client objectives.

As a extension for discovery and composition techniques, explication of usage conditions for Web services works on generic goal formulations and functional descriptions of Web services. There are different possibilities for presenting this to clients: we can either construct revised goal formulations with a different matchmaking degree such that the conditions do not arise anymore, or we can explicate the conditions as additional constraints that need to hold for successful goal solving with the particular Web services. For both, Δ-relations provide the basis as we illustrate in the following. For convenience, we stick to the example of buying tickets for travelling from Innsbruck to Vienna throughout the explanations. We examine the arising requirements for Δ-relations below.

Goal Refinement / Adjustment. This refers to the subsequent refinement of generic goal formulations. For instance, imagine that the client who defines G_1 has a preference for travelling by plane because of tight schedules but did not explicate this in the initial specification of G_1. After receiving the discovery result for the initial goal definition, the client may refine G_1 with a preference for flying. In the next step, WS_2 would be selected as the impact for goal resolution by using WS_1 contradicts with the client's preference.

Goal adjustment means that a goal formulation is refined such that the nature of the specified objective is changed. Within our scenario, imagine that some client defines a Goal G_2 for buying a flight ticket from Innsbruck to Vienna for less than € 200,-. The flight booking Web service WS_2 seems to be usable, but all purchasable flight ticket cost significantly more than € 200,-. However, train tickets for the desired itinerary are available for less than € 200,- from WS_1. By analyzing the functional relationship and difference between G_2 and the Web services, we can propose possibilities for adjusting the goal: either weaken it by omitting the price constraint so that WS_2 becomes usable, or change the desired ticket type to train ticket so that WS_1 becomes usable. We discuss examples for goal re-formulations in detail in Section 2.4.

Assumption Explication. This is concerned with determining so-called *hidden assumptions*, that is additional aspects that are not explicated a priori within formalized descriptions of requested and provided functionalities. Such hidden assumption result from the fact that knowledge engineers specify formal descriptions with respect to their individual understanding and view of the world. Following constructivist philosophy, this naturally differs between humans. As missing knowledge might hamper usability of a computational resource for solving a problem, explication of hidden assumptions is considered as an important and critical task for enabling automated software usage on basis of formal descriptions [4]. [9] presents so-called *inverse verification* as a technique for assumption explication by manual analysis of failed proofs on the usability of problem solving methods for a given task.

We can understand the conditions or impacts on using a particular Web service for solving a goal as a specific type of such hidden assumptions. For the usage of the AUA Web service in the above example scenario, we could formalize a constraint that the origin and destination provided as inputs need to be Austrian cities with an airport. We can use this as an additional constraint for using WS_2 for resolving G. We illustrate how to obtain such assumptions by Δ-relations in Section 2.4, and discuss the similarity and differences of our approach and inverse verification in Section 4.

While the above techniques are concerned with explicating usage conditions on Web services, another potential application area of Δ-relations might be **grouping of requested and provided functionalities.** Outlined in [31], this addresses the scalability problem of techniques for automated Web service usage. The bottleneck is the potentially very large number of matchmaking operations that need to be performed for discovery or composition as all available Web services need to be taken into account. For reducing this, we can group available Web services with respect to similar provided functionalities in order to attain an efficient search graph for Web service repositories. Furthermore, we can group goals with respect to the relationship and difference of the requested functionalities and capture knowledge about discovery and composition results in order to perform usability detection by look-up. This is the idea of *Semantic Goal Caching* that will be represented at a later stage of research, along with the usability Δ-relations therefore.

1.3 Characteristics of Delta-Relations

Considering the realization of such techniques reveals that they all deal with certain aspects of the semantic difference between the formalized functional descriptions of Web services and goals that do not precisely match. Goal refinement and adjustment need to analyze the impacts of using specific Web services for a given goal and derive a semantically related goal formulation; representing usage conditions in form of assumption needs to explicate the relevant aspects of the semantic difference between the goal and the Web service; functional grouping of Web services and goals needs to determine taxonomic structures with respect to formal functional descriptions.

Hence, there is a common basis for applications as illustrated above: the semantic relation and difference of formally described requested and provided functionalities with respect to the context of the usability of Web services for solving a given goal. This is what we refer to as a Δ-relations in the following. In principle, it provides a "raw form" of explicit knowledge on the relation and difference of formally described functionalities. This is used and interpreted by distinct techniques for realizing applications as discussed above. Figure 1 illustrates the correlation of Δ-relations and techniques that work upon them for enabling particular applications.

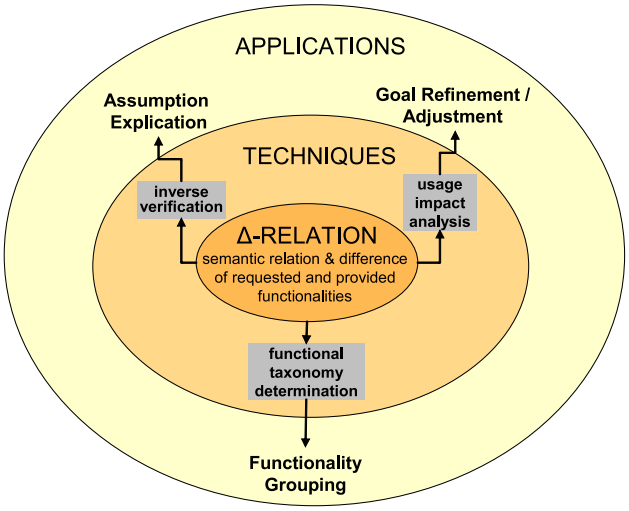

Figure 1: Delta Relations - Techniques - Applications

To provide a sophisticated basis for inference-based techniques and advanced applications for Web service detection, Δ-relations need to meet certain requirements. While providing further explanations and concise formal definitions in the subsequent sections, we depict the requirements here informally:

1. **completeness**, i.e. cover all aspects of difference (necessary condition)
2. **minimality**, i.e. only encompass those aspects needed to cover the complete difference but nothing more (sufficient condition).
3. **formal definition** of the properties and meaning of Δ-relations in a logical framework
4. **declarative representation**, i.e. explicit representation of Δ-relations in a declarative manner.

The remainder of this document subsequently elaborates and defines Δ-relations with respect to the identified requirements. At first, Section 2 discusses the usage of conventional logical expressions as formal descriptions of goals and Web services, and introduces the definition of a Δ-relation as a pair of formulae that properly represent the relevant aspects of the semantic difference between goal and Web service descriptions. This is extended in Section 3 with respect to the structure and semantics of functional descriptions in terms of pre- and postconditions. Moreover, we provide two extensions to the mathematical core of Δ-relations: (1) the alignment with semantic matchmaking in Section 2.3 with respect to the application context of Web service detection, and (2) the integration within the WSMO mediation framework in Section 3.3 to allow directed, explicit, and declarative specification of Δ-relations.

2 Foundations – Delta Relations for Logical Expressions

This section discusses the properties and definition of Δ-relations for conventional logical expressions in some ontology language that formally describe requested and provided functionalities. While this abstracts from the structure and semantics of formal functional descriptions in some capability language, we subsequently extend the definitions to functional descriptions in terms of pre- and postconditions in Section 3.

We first recall the meaning of logical expressions as formulae in logics with model-theoretic semantics and define the semantic difference between such formulae. Then, we provide a formal definition of Δ-relations with respect to the specific requirements arising for Web service detection as the relevant application context. We align this with semantic matchmaking for Web service detection, and finally demonstrate the definitions and their application within an illustrative example.

2.1 Logical Expressions, Model-Theoretic Semantics, and Semantic Difference

Logics are traditional means for knowledge representation in Computer Science. Each logic $\mathfrak{L}$ consists at least of two fundamental parts: a formal language $\mathcal{L}$ and a formal semantics $\mathcal{S}$. As the basis for the following elaborations, we briefly recall the definitions of $\mathcal{L}$ and $\mathcal{S}$ for classical first-order logic as defined in [30].

$\mathcal{L}$ provides the syntactic means to represent knowledge as statements in a limited language. The language depends on a set of elementary symbols Σ (called *signature*) which can be used in a predefined manner to construct statements. Σ contains symbols with a pre-defined meaning inside the particular logic $\mathfrak{L}$ (*logical symbols*, e.g. $\wedge$, $\vee$, $\neg$, $true$), symbols that do not have such a fixed meaning (*non-logical symbols*, e.g. $human(\cdot)$, $jack\ parent-of(\cdot,\cdot)$) and auxiliary symbols without any logical meaning (e.g. opening and closing brackets, commas, etc.). Usually, one only considers the part of the non-logical symbols when talking about a signature Σ.

The formal semantics $\mathcal{S}$ assigns a truth value $val_{\mathcal{I}}(\phi)$ to any consistent statement $\phi \in \mathcal{L}$. The evaluation depends on a particular context $\mathcal{I}$ (usually called *interpretation* of signature Σ) which gives meaning to the non-logical symbols in the signature Σ. In classical KR languages the range of $val_{\mathcal{I}}$ contains precisely two values, namely $true$ and $false$. Regarding the semantics of a formula (or logical expression) $\phi \in \mathcal{L}$, the formal semantics $\mathcal{S}$ essentially divides the set of all possible interpretations (of signature Σ) into two partitions: (1) interpretations $\mathcal{I}$ that satisfy ϕ, i.e. $val_{\mathcal{I}}(\phi) = true$ (these interpretation are called *models* of ϕ), and (2) interpretation where this is not the case, i.e. $val_{\mathcal{I}}(\phi) = false$. The former case is often represented in terms of a semantic relation (read ,,satisfies" or ,,models") between an interpretation and a formula: $\mathcal{I} \models \phi$. In consequence, the class of models of a formula ϕ over the respective signature Σ

$$Mod_{\Sigma}(\phi) = \{\mathcal{I} \,|\, \mathcal{I}\ interpretation\ over\ \Sigma\ such\ that\ \mathcal{I} \models \phi\}$$

can be considered as the semantics of ϕ. This can be used as the formal description $\mathcal{D}$ of some resource. For instance, if ϕ formally describes a Web service then $Mod_{\Sigma}(\phi)$ denotes those interpretations that can be provided by the Web service with respect to some specific inputs. This perspective on the semantics of formulas (or formal descriptions) is called *model-theoretic semantics* and is illustrated in Figure 2.

The model-theoretic perspective on the semantics of formal descriptions provides a natural way to define the semantic difference between two given descriptions. Let ϕ_1, ϕ_2 be formal description defined as logical formulae whose meaning is formally represented by the set of models $Mod_{\Sigma}(\phi_1), Mod_{\Sigma}(\phi_2)$. Descriptions with the same models (i.e. $Mod_{\Sigma}(\phi_1) = Mod_{\Sigma}(\phi_2)$) are semantically equivalent, i.e. they mean the same (though they might look syntactically different). Therefore, every interpretation $\mathcal{I} \in Mod_{\Sigma}(\phi_1)$ that is not

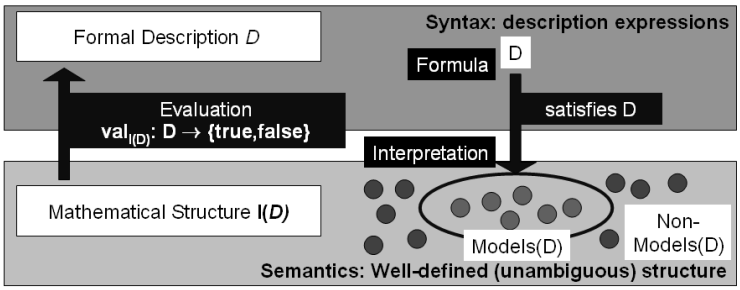

Figure 2: Model-theoretic Semantics of Formal Descriptions

part of the semantics $Mod_\Sigma(\phi_2)$ of another description ϕ_2 (or vise versa) represents a bit of information that distinguishes ϕ_1 and ϕ_2 semantically. This leads straightforwardly to the following characterization of difference on a semantic level which is illustrated in Figure 3:

Definition 1 (Semantic Difference)**.** *Let $\phi_1, \phi_2 \in \mathcal{L}$ be two formulae over signature Σ and let $Int(\Sigma)$ denote the set of all Σ-interpretations.*

*The **semantic difference** $SemDiff(\phi_1, \phi_2)$ between ϕ_1 and ϕ_2 is the set of Σ-interpretations $\mathcal{I}$ that are models of either ϕ_1 or ϕ_2 but not both. Formally, this means*

$$SemDiff(\phi_1, \phi_2) = \{\mathcal{I} \in Int(\Sigma) \mid (\mathcal{I} \models \phi_1 \text{ and } \mathcal{I} \not\models \phi_2) \text{ or } (\mathcal{I} \not\models \phi_1 \text{ and } \mathcal{I} \models \phi_2)\}$$

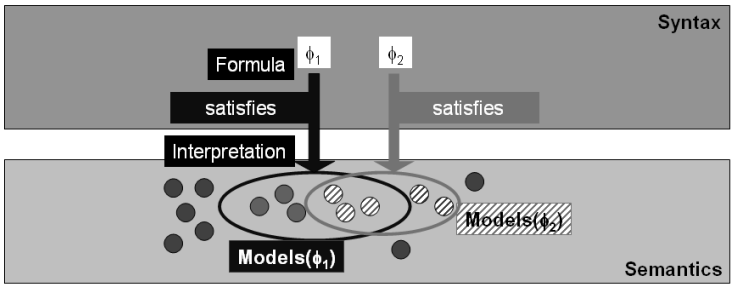

Figure 3: Semantic Difference of Formal Descriptions under Model-theoretic Semantics

According to this definition, semantic difference is a *semantic* (or mathematical) concept, i.e. it is defined to be a particular set of interpretations. However, in order to leverage the semantic difference of logical expressions in applications later on, this object is not directly useful as such. Applications that deal with semantic descriptions never have direct access to the semantic level, but they merely access the semantic level indirectly via formal descriptions in some logic. Hence, we need a concrete representation of the semantic difference of two given logical expressions $\phi_1, \phi_2 \in \mathfrak{L}$ in a logic $\mathfrak{L}'$. More precisely, we need a formal statement $\delta \in \mathfrak{L}'$ such that $Mod_\Sigma(\delta) = SemDiff(\phi_1, \phi_2)$. Depending on the expressiveness of the logic $\mathfrak{L}$ to which the original descriptions belong, $\mathfrak{L}'$ might or might not be the same as $\mathfrak{L}$. In some cases, $\mathfrak{L}'$ might require additional language constructs (i.e. must be more expressive) than $\mathfrak{L}$. In consequence, we identify a formula which faithfully represents the semantic difference and makes it accessible for applications that deal with formal descriptions.

Definition 2 (Symmetric Difference)**.** *Let $\mathfrak{L}$ be a logic with model-theoretic semantics. Let $\phi_1, \phi_2 \in \mathfrak{L}$ be two formulae over signature Σ. Let $\mathfrak{L}'$ be an extension of $\mathfrak{L}$ which provides additionally the two logical connectives conjunction ($\wedge$) and negation ($\neg$) with the usual classical semantics. Then we call the expression $\delta(\phi_1, \phi_2) \in \mathfrak{L}'$ with*

$$\delta_1(\phi_1, \phi_2) = (\phi_1 \wedge \neg\phi_2)$$

$$\delta_2(\phi_1, \phi_2) = (\neg\phi_1 \wedge \phi_2)$$

$$\delta(\phi_1, \phi_2) = \delta_1(\phi_1, \phi_2) \vee \delta_2(\phi_1, \phi_2) = (\phi_1 \wedge \neg\phi_2) \vee (\neg\phi_1 \wedge \phi_2)$$

*the **symmetric difference** of ϕ_1 and ϕ_2*[1]*.*

Proposition 1. *Let $\mathfrak{L}$ be a classical logic with model-theoretic semantics. Let $\phi_1, \phi_2 \in \mathfrak{L}$ be two formulae over signature Σ. Let $\mathfrak{L}'$ be an extension of $\mathfrak{L}$ which provides additionally the two logical connectives conjunction ($\wedge$) and negation ($\neg$) with the usual classical semantics.*

Then it holds that the symmetric difference $\delta(\phi_1, \phi_2)$ of ϕ_1 and ϕ_2 formally represents the semantic difference of ϕ_1 and ϕ_2, i.e.

$$Mod_\Sigma(\delta(\phi_1, \phi_2)) = \mathit{SemDiff}(\phi_1, \phi_2)$$

If $\mathfrak{L}$ already provides classical conjunction and negation, then we can represent the symmetric difference already in $\mathfrak{L}$ itself.

Proof. Consider $\mathcal{I} \in Mod_\Sigma(\delta(\phi_1, \phi_2))$, i.e. a Σ-interpretation $\mathcal{I} \models_{\mathfrak{L}'} \delta(\phi_1, \phi_2)$.

	$\mathcal{I} \models_{\mathfrak{L}'} \delta(\phi_1, \phi_2)$	
iff.	$\mathcal{I} \models_{\mathfrak{L}'} (\phi_1 \wedge \neg\phi_2) \vee (\neg\phi_1 \wedge \phi_2)$	by Def. 2
iff.	$\mathcal{I} \models_{\mathfrak{L}'} (\phi_1 \wedge \neg\phi_2)$ or $\mathcal{I} \models_{\mathfrak{L}'} (\neg\phi_1 \wedge \phi_2)$	by semantics of $\vee$ in $\mathfrak{L}'$
iff.	($\mathcal{I} \models_{\mathfrak{L}'} \phi_1$ and $\mathcal{I} \not\models_{\mathfrak{L}'} \phi_2$) or ($\mathcal{I} \not\models_{\mathfrak{L}'} \phi_1$ and $\mathcal{I} \models_{\mathfrak{L}'} \phi_2$)	by semantics of $\wedge, \neg$ in $\mathfrak{L}'$
iff.	($\mathcal{I} \models_{\mathfrak{L}} \phi_1$ and $\mathcal{I} \not\models_{\mathfrak{L}} \phi_2$) or ($\mathcal{I} \not\models_{\mathfrak{L}} \phi_1$ and $\mathcal{I} \models_{\mathfrak{L}} \phi_2$)	since $\mathfrak{L}'$ conservatively extends $\mathfrak{L}$
iff.	$\mathcal{I} \in \mathit{SemDiff}(\phi_1, \phi_2)$	by Def. 1

Therefore, we can conclude $Mod_\Sigma(\delta(\phi_1, \phi_2)) = \mathit{SemDiff}(\phi_1, \phi_2)$. □

The above definitions hold for all specific logics that have model-theoretic semantics and classical negation. In particular, these are first-order logic [30] and its decidable sub-sets Description Logic [2] and Horn Logic [14], as well as Description Logic Programs as their maximal intersection [12]. Extensions towards applicability for logics with minimal model semantics and negation by failure that underlie Logic Programming languages - e.g. datalog with default negation [36] and the variants “WSML-Flight”and “WSML-Rule”of the WSML language [5] - are considered as future work at this point in time.

[1] As usual, the logical connective disjunction ($\vee$) can be expressed in terms of conjunction and negation: $\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$

2.2 Requirements and Definition of Delta Relations

We now turn to the concrete context of Web service detection. The relevant aspect that we want to represent in Δ-relations is the semantic difference between formally described requested and provided functionalities for Web services. The following derives formal definitions of the requirements of completeness and minimality of Δ-relations as informally identified in Section 1.3. We then show how to construct a formal expression that has these desired properties.

Consider some Web service $\mathcal{W}$ that offers some specific functionality and a goal $\mathcal{G}$ that specifies what functionality a potential client is seeking for. In this section, we assume both functionalities to be described by formulae in a logic with model-theoretic semantics such that $\phi_\mathcal{G}$ is the formal description of $\mathcal{G}$ and $\phi_\mathcal{W}$ the one for $\mathcal{W}$. Both formulae are defined on basis of a signature Σ that defines the terminology and knowledge of the domain of interest; we assume all goal and Web service descriptions to be consistent formulae such that there exists at least one Σ-interpretation that is a model of the description.

As the common result of several research efforts around Web service discovery by semantic matchmaking (e.g. [21, 16], see Section 2.3 for details), there are different logical relationships between $\phi_\mathcal{G}$ and $\phi_\mathcal{W}$ under which the Web service is considered to be usable for solving the goal. For functionalities described by logical expressions, the notable ones are the exact match ($\phi_\mathcal{G} \equiv \phi_\mathcal{W}$), proper logical entailments ($\phi_\mathcal{G} \models \phi_\mathcal{W}$ or $\phi_\mathcal{W} \models \phi_\mathcal{G}$), and existence of at least one common model without entailment, commonly referred to as the intersection match: $\exists \mathcal{I}(\Sigma).\, \mathcal{I}(\Sigma) \in Mod_\Sigma(\phi_\mathcal{G}) \wedge \mathcal{I}(\Sigma) \in Mod_\Sigma(\phi_\mathcal{W})$ and $\phi_\mathcal{G} \not\models \phi_\mathcal{W}$ and $\phi_\mathcal{W} \not\models \phi_\mathcal{G}$.

The aspects of interest for Δ-relations arise when there is a proper entailment or an intersection match between the goal and the Web service descriptions. In each of these situations, there exists at least one Σ-interpretation that is a model for either $\phi_\mathcal{G}$ or $\phi_\mathcal{W}$ but is not common to them. The set of these interpretations constitute the semantic difference between goals and Web services. This causes certain restrictions or impacts on how the goal can be solved by using the particular Web service as illustrated in the introduction. Two central requirements arise for precisely denoting this within Δ-relations:

1. as the **necessary condition** of completeness, the Δ-relation between $\phi_\mathcal{G}$ and $\phi_\mathcal{W}$ needs to encompass *all* Σ-interpretations that are models for one but not the other description. This is given if $\Delta_{\phi_\mathcal{G},\phi_\mathcal{W}}$ allows establishing logical equality between the goal and the Web service description such that $\Sigma \models \phi_\mathcal{G} \wedge \Delta_1 \equiv \phi_\mathcal{W} \wedge \Delta_2$. The reason is that if $\phi_\mathcal{G} \equiv \phi_\mathcal{W}$ holds, then there does not exist any Σ-interpretation that is a model for one formula but not the other; hence, there is no semantic difference that causes undesired impacts of using W for solving G.

2. as the **sufficient condition** for appropriate handling, a Δ-relation should be the *minimal expression* that correctly denotes the semantic difference between $\phi_\mathcal{G}$ and $\phi_\mathcal{W}$. This means that $\Delta_{\phi_\mathcal{G},\phi_\mathcal{W}}$ only explicates those Σ-interpretations that constitute the semantic difference, which is given if there does not exists any $\Delta_{\phi_\mathcal{G},\phi_\mathcal{W}}$' that satisfies the necessary condition and is logically entailed by $\Delta_{\phi_\mathcal{G},\phi_\mathcal{W}}$.

Definition 5 formalizes these requirements on Δ-relations, while Definitions 3 and 4 provide the formal tools therefore. With respect to this, Theorem 1 defines the construction of Δ-relations that satisfy the necessary and sufficient conditions. The central properties of these definitions are first that Δ-relations are defined as a pair of expressions $\Delta = (\Delta_1, \Delta_2)$ whereby Δ_1 denotes the restrictions for the goal description $\phi_\mathcal{G}$ and Δ_2 those for the Web service description $\phi_\mathcal{W}$ for establishing logical equality between them. As discussed below, this provides a general formula for constructing Δ-relations in all situations that can occur between formal descriptions. Secondly, there is not a unique syntactical representation of Δ-relations. Rather, all semantically equivalent expressions that satisfy the necessary and sufficient conditions represent accurate Δ-relations.

Proposition 2 (Entailment Pre-ordering on $\mathcal{L}$)**.** *Let $\phi_1, \phi_2 \in \mathcal{L}$ be formulae over signature Σ in logic $\mathfrak{L}$.*
The logical entailment relation $\models_{\mathfrak{L}} \subseteq \mathcal{L} \times \mathcal{L}$ represents a pre-order[2]*.*

Based on the pre-order $\models_{\mathfrak{L}}$ we can define a partial order[2] on formulae in $\mathfrak{L}$ as follows:

Definition 3 (Entailment Ordering of Formulae)**.** *Let $\phi_1, \phi_2, \phi_1', \phi_2' \in \mathcal{L}$ be formulae over signature Σ in logic $\mathfrak{L}$. Let $[\phi]_{\equiv_{\mathfrak{L}}}$ denote the* ***equivalence class*** *of formula ϕ wrt. logical equivalence in $\mathfrak{L}$, i.e.*

$$[\phi]_{\equiv_{\mathfrak{L}}} = \{\phi' \mid \phi \models_{\mathfrak{L}} \phi' \; and \; \phi' \models_{\mathfrak{L}} \phi\}$$

We define a partial order $\preceq \subseteq \mathcal{L}/_{\equiv_{\mathfrak{L}}} \times \mathcal{L}/_{\equiv_{\mathfrak{L}}}$ between formulae modulo equivalence (i.e. on equivalence classes in $\mathfrak{L}$ wrt. equivalence relation $\equiv_{\mathfrak{L}}$) in $\mathfrak{L}$ by

$$[\phi_1]_{\equiv_{\mathfrak{L}}} \preceq [\phi_2]_{\equiv_{\mathfrak{L}}} \quad \textit{iff}. \quad \phi_1 \models_{\mathfrak{L}} \phi_2$$

We call $\preceq$ ***entailment ordering on*** *$\mathfrak{L}$. By $\prec$ we denote the respective* ***strict entailment order on*** *$\mathfrak{L} \times \mathfrak{L}$, i.e.*

$$[\phi_1]_{\equiv_{\mathfrak{L}}} \prec [\phi_2]_{\equiv_{\mathfrak{L}}} \quad \textit{iff}. \quad [\phi_1]_{\equiv_{\mathfrak{L}}} \preceq [\phi_2]_{\equiv_{\mathfrak{L}}} \; and \; [\phi_1]_{\equiv_{\mathfrak{L}}} \neq [\phi_2]_{\equiv_{\mathfrak{L}}}$$

We can extend the strict entailment order on $\mathfrak{L}$ in a natural way to a strict partial ordering on $\mathfrak{L} \times \mathfrak{L}$:

Definition 4 (Strict Entailment Ordering on Pairs of Formulae)**.** *Let $\phi_1, \phi_2, \phi_1', \phi_2' \in \mathcal{L}$ be formulae over signature Σ in logic $\mathfrak{L}$.*
We define the ***strict entailment ordering*** *$\sqsubset$* ***between pairs of formulae in*** *$\mathfrak{L}$ based on $\prec$ and $\preceq$, i.e. by*

$$(\phi_1, \phi_2) \sqsubset (\phi_1', \phi_2') \quad \textit{iff}. \quad \begin{array}{l} ([\phi_1]_{\equiv_{\mathfrak{L}}} \prec [\phi_1']_{\equiv_{\mathfrak{L}}} \; and \; [\phi_2]_{\equiv_{\mathfrak{L}}} \preceq [\phi_2']_{\equiv_{\mathfrak{L}}}) \; or \\ ([\phi_1]_{\equiv_{\mathfrak{L}}} \preceq [\phi_1']_{\equiv_{\mathfrak{L}}} \; and \; [\phi_2]_{\equiv_{\mathfrak{L}}} \prec [\phi_2']_{\equiv_{\mathfrak{L}}}) \end{array}$$

Proposition 3 (Properties of the Strict Entailment Ordering)**.** *The strict entailment orderings $\prec$ on $\mathfrak{L}$ and $\sqsubset$ on $\mathfrak{L} \times \mathfrak{L}$ are strict partial orderings*[2]*.*

On basis of this formal toolset, we can capture the necessary and sufficient requirements on Δ-relations as follows:

Definition 5 (Formal Requirements for Delta Relations)**.** *Let $\phi_{\mathcal{G}}$ be a formula denoting a functionality requested in a goal and $\phi_{\mathcal{W}}$ be a formula denoting functionality provided by a Web service; both are defined on basis of a signature Σ in the context of some background ontology $\Omega \subseteq \mathfrak{L}$ that specifies domain knowledge. A pair of formulas $\Delta = (\Delta_1, \Delta_2)$ is called* ***Δ-relation*** *of $\phi_{\mathcal{G}}$ and $\phi_{\mathcal{W}}$ over Ω if it has the following properties:*

[2] Definitions of used standard logical notions:

- a **pre-order** $\leq$ is a reflexive and transitive binary relation on a set S with: $\forall a, b, c \in S.\ a \leq a$, if $a \leq b$ and $b \leq c$ then $a \leq c$.
- a **partial order** R is a reflexive, antisymmetric, and transitive binary relation on a set S with: $\forall a, b, c \in S.\ aRa$, if aRb and bRa then $a = b$, if aRb and bRc then aRc;
- a **strict partial order** R is a irreflexive, antisymmetric, and transitive binary relation on a set S with: $\forall a, b, c \in S.\ \neg(aRa)$, if aRb and bRa then $a = b$, if aRb and bRc then aRc;

(i) Δ *must allow establishing logical equivalence between* $\phi_{\mathcal{G}}$ *and* $\phi_{\mathcal{W}}$*:*
$\Omega \models_{\mathfrak{L}} (\phi_{\mathcal{G}} \wedge \Delta_1) \leftrightarrow (\phi_{\mathcal{W}} \wedge \Delta_2)$

(ii) Δ *must be a minimal expression with such a property, i.e. formally there does not exist any pair of formulae* $\Delta' = (\Delta'_1, \Delta'_2)$ *such that* $\Omega \models_{\mathfrak{L}} (\phi_{\mathcal{G}} \wedge \Delta'_1) \leftrightarrow (\phi_{\mathcal{W}} \wedge \Delta'_2)$ *and* $\Delta \sqsubset \Delta'$

In this definition, clause (i) states that Δ specifies the needed restrictions that one has to impose on both, the goal description (Δ_1) and the Web service (Δ_2) such that the descriptions become logically equivalent. There may be more than one such Δ expression and certain Δ expressions might be more specific than others, i.e. they impose more restrictions than others (and thus than actually needed). For instance $\Delta = (false, false)$ is a delta expression satisfying clause (i), but in most cases intuitively not useful. Clause (ii) states in a formal way that amongst the possible Δ expressions, we are only interested in most general ones (in regard of the strict entailment order $\sqsubset$), i.e. those descriptions that impose only minimal necessary restrictions on both descriptions to achieve logical equivalence between $\phi_{\mathcal{W}}$ and $\phi_{\mathcal{G}}$.

It is easy to see that delta relations are not unique: If $\Delta = (\Delta_1, \Delta_2)$ is a delta relation for $\phi_{\mathcal{G}}$ and $\phi_{\mathcal{W}}$, then any $\Delta' = (\Delta'_1, \Delta'_2)$ such that $\Delta_1 \equiv_{\mathfrak{L}} \Delta'_1$ and $\Delta_2 \equiv_{\mathfrak{L}} \Delta'_2$ is a delta relation as well. However, a delta relation is *semantically* unique, i.e. for any other delta relation $\Delta' = (\Delta'_1, \Delta'_2)$ it must hold that $\Delta_1 \equiv_{\mathfrak{L}} \Delta'_1$ and $\Delta_2 \equiv_{\mathfrak{L}} \Delta'_2$.

Having formally characterized what kind of formal description we are interested in, the question arises of how to find concrete Δ-relations for a given Web service and goal description. Our previous considerations on the semantic difference of logical expressions and how to represent them guide us to the following idea. To satisfy clause (i), Δ_1 needs to restrict the actual models of $\phi_{\mathcal{G}}$ to only those that are also models of $\phi_{\mathcal{W}}$. For the sufficient condition (clause ii), Δ_1 needs to be the most general expression (with respect to strict entailment ordering) that satisfies the necessary condition. The same holds for Δ_2. In essence, we are describing here the complement of some specific component of the semantic difference between $\phi_{\mathcal{G}}$ and $\phi_{\mathcal{W}}$, namely $\delta_1(\phi_{\mathcal{G}}, \phi_{\mathcal{W}})$ and $\delta_2(\phi_{\mathcal{G}}, \phi_{\mathcal{W}})$. With respect to Definition 2, $\delta_1(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}) \vee \delta_2(\phi_{\mathcal{G}}, \phi_{\mathcal{W}})$ denotes the symmetric difference of $\phi_{\mathcal{G}}$ and $\phi_{\mathcal{W}}$. The following theorem underpins this intuition formally:

Theorem 1 (Construction of a Delta Relation)**.** *Let* $\phi_{\mathcal{G}}$ *be a formula denoting a functionality requested in a goal and* $\phi_{\mathcal{W}}$ *be a formula denoting functionality provided by a Web service; both are defined on basis of a signature* Σ *in the context of some background ontology* $\Omega \subseteq \mathfrak{L}$ *that specifies domain knowledge.*

Then the pair of formulas $\Delta(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}) = (\neg\delta_1(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}), \neg\delta_2(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}))$ *with*

$$\delta_1(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}) = (\phi_{\mathcal{G}} \wedge \neg\phi_{\mathcal{W}})$$
$$\delta_2(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}) = (\neg\phi_{\mathcal{G}} \wedge \phi_{\mathcal{W}})$$

is a Δ*-relation of* $\phi_{\mathcal{G}}$ *and* $\phi_{\mathcal{W}}$ *over* Ω*.*

Proof. Let $\Delta(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}) = (\Delta_1, \Delta_2)$ be as defined above, i.e. $\Delta_1 = \neg(\phi_{\mathcal{G}} \wedge \neg\phi_{\mathcal{W}})$ and $\Delta_2 = \neg(\neg\phi_{\mathcal{G}} \wedge \phi_{\mathcal{W}})$.

We show that $\Delta(\phi_{\mathcal{G}}, \phi_{\mathcal{W}})$ satisfies clause (i) of Definition 5. Let $\mathcal{I}$ be a Σ-interpretation that satisfies Ω, and let $\mathcal{I} \models \Delta_1 \wedge \phi_{\mathcal{G}}$, i.e. $\mathcal{I} \models \neg(\phi_{\mathcal{G}} \wedge \neg\phi_{\mathcal{W}}) \wedge \phi_{\mathcal{G}}$. Since $\neg(\phi_{\mathcal{G}} \wedge \neg\phi_{\mathcal{W}}) \wedge \phi_{\mathcal{G}} \equiv_{\mathfrak{L}} \phi_{\mathcal{W}} \wedge \phi_{\mathcal{G}}$, we have $\mathcal{I} \models \phi_{\mathcal{W}} \wedge \phi_{\mathcal{G}}$ $\langle 0 \rangle$. Furthermore, as $\phi_{\mathcal{W}} \wedge \Delta_2 = (\phi_{\mathcal{W}} \wedge \neg(\neg\phi_{\mathcal{G}} \wedge \phi_{\mathcal{W}})) \equiv_{\mathfrak{L}} \phi_{\mathcal{W}} \wedge \phi_{\mathcal{G}}$, it also holds that $\mathcal{I} \models \phi_{\mathcal{W}} \wedge \Delta_2$. Since we only used equivalence transformations, the same argument gives us the opposite direction and thus $\Omega \models (\Delta_1 \wedge \phi_{\mathcal{G}}) \leftrightarrow (\phi_{\mathcal{W}} \wedge \Delta_2)$ $\langle 1 \rangle$.

Next, we show that clause (ii) of Definition 5 is satisfied by $\Delta(\phi_{\mathcal{G}}, \phi_{\mathcal{W}})$ as well. We prove by contradiction and assume that there exists a $\Delta' = (\Delta'_1, \Delta'_2)$ such that $\Delta(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}) \sqsubset \Delta'$ $\langle 2 \rangle$ and $\Omega \models (\Delta'_1 \wedge \phi_{\mathcal{G}}) \leftrightarrow$

$(\phi_{\mathcal{W}} \wedge \Delta_2')$ $\langle 3 \rangle$. From $\langle 3 \rangle$, we conclude that $Mod_\Sigma(\Omega \cup \{\Delta_1' \wedge \phi_{\mathcal{G}}\}) = Mod_\Sigma(\Omega \cup \{\phi_{\mathcal{W}} \wedge \Delta_2'\})$ $\langle 4 \rangle$. Similarly, from $\langle 1 \rangle$ we get that $Mod_\Sigma(\Omega \cup \{\Delta_1 \wedge \phi_{\mathcal{G}}\}) = Mod_\Sigma(\Omega \cup \{\phi_{\mathcal{W}} \wedge \Delta_2\})$. By $\langle 2 \rangle$ we know that $[\Delta_1]_{\equiv_\mathfrak{L}} \prec [\Delta_1']_{\equiv_\mathfrak{L}}$ or $[\Delta_2]_{\equiv_\mathfrak{L}} \prec [\Delta_2']_{\equiv_\mathfrak{L}}$. Without loss of generality, assume that $[\Delta_1]_{\equiv_\mathfrak{L}} \prec [\Delta_1']_{\equiv_\mathfrak{L}}$ (the other case works analogously). Then, $Mod_\Sigma(\Delta_1) \subset Mod_\Sigma(\Delta_1')$ and $Mod_\Sigma(\Omega \cup \{\Delta_1\}) \subset Mod_\Sigma(\Omega \cup \{\Delta_1'\})$ and $Mod_\Sigma(\Omega \cup \{\Delta_1 \wedge \phi_{\mathcal{G}}\}) \subset Mod_\Sigma(\Omega \cup \{\Delta_1' \wedge \phi_{\mathcal{G}}\})$. By $\langle 0 \rangle$, we know that $Mod_\Sigma(\Omega \cup \{\Delta_1 \wedge \phi_{\mathcal{G}}\}) = Mod_\Sigma(\Omega \cup \{\phi_{\mathcal{G}} \wedge \phi_{\mathcal{W}}\}) \subset Mod_\Sigma(\Omega \cup \{\Delta_1' \wedge \phi_{\mathcal{G}}\})$ $\langle 5 \rangle$.

We now show that $Mod_\Sigma(\Omega \cup \{\Delta_1' \wedge \phi_{\mathcal{G}}\}) \subseteq Mod_\Sigma(\Omega \cup \{\phi_{\mathcal{G}} \wedge \phi_{\mathcal{W}}\})$ has to hold too which gives us the desired contradiction, since then $Mod_\Sigma(\Omega \cup \{\Delta_1 \wedge \phi_{\mathcal{G}}\}) = Mod_\Sigma(\Omega \cup \{\Delta_1' \wedge \phi_{\mathcal{G}}\})$ would have to hold which is not compatible with $\langle 5 \rangle$. For this last step, assume that $Mod_\Sigma(\Omega \cup \{\Delta_1' \wedge \phi_{\mathcal{G}}\}) \not\subseteq Mod_\Sigma(\Omega \cup \{\phi_{\mathcal{G}} \wedge \phi_{\mathcal{W}}\})$, i.e. there exists $\mathcal{I} \in Mod_\Sigma(\Omega \cup \{\Delta_1' \wedge \phi_{\mathcal{G}}\}$ such that $\mathcal{I} \notin Mod_\Sigma(\Omega \cup \{\phi_{\mathcal{G}} \wedge \phi_{\mathcal{W}}\}$. For this $\mathcal{I}$ it then must hold that $\mathcal{I} \not\models \phi_{\mathcal{W}}$, since $\mathcal{I} \models \Omega$ and $\mathcal{I} \models \phi_{\mathcal{G}}$. At the same time, by $\langle 4 \rangle$ it must hold that $\mathcal{I} \in Mod_\Sigma(\Omega \cup \{\phi_{\mathcal{W}} \wedge \Delta_2'\})$ and thus $\mathcal{I} \models \phi_{\mathcal{W}}$. Therefore, we have a contradiction by which we can conclude $Mod_\Sigma(\Omega \cup \{\Delta_1' \wedge \phi_{\mathcal{G}}\}) \subseteq Mod_\Sigma(\Omega \cup \{\phi_{\mathcal{G}} \wedge \phi_{\mathcal{W}}\})$ which completes the proof.

□

Proposition 4 (Properties of Delta Relations)**.** *The goal and Web service descriptions restricted by* Δ_1 *and* Δ_2 *logically entail the original descriptions:* $\phi_{\mathcal{G}} \wedge \Delta_1 \models \phi_{\mathcal{G}}$ *and* $\phi_{\mathcal{W}} \wedge \Delta_2 \models \phi_{\mathcal{W}}$.

2.3 Alignment with Semantic Matchmaking

With respect to Web service detection as the problem and application context of Δ-relations, the following presents the alignment of the above definitions with semantic matchmaking for Web service detection.

As the core and common result of several research efforts, five degrees of match are distinguished: `exact, plugin, subsume, intersection` wherein a Web service is usable for solving a goal, and `disjoint` wherein this is not given. These notions denote specific logical relationships between formalized descriptions of goals and Web services and are used as proof obligations for determining the usability of a Web service for a given request.[3] The central aspect of aligning Δ-relations with the semantic matchmaking notions is that certain properties hold for the Δ-relation for each distinct situation of matching. In particular, for each matchmaking notion there are known values for the formulae that the Δ-relations consists of. These can be used either for simpler construction of the Δ-relation in case that the matchmaking degree is known (as values for parts of the Δ-relation are known), or determination of the matchmaking degree if the Δ-relation is known.

As a basis for formally defining this relationship, the following recalls the definition of the matchmaking notions. In accordance to the above, definitions, we consider a goal and a Web service described by formulae in a logic with model-theoretic semantics: $\phi_{\mathcal{G}}$ and $\phi_{\mathcal{W}}$. We define the matchmaking notions from the perspective of the goal. Moreover, for each matchmaking notion we discuss the implications of Web service usage as well as the a priori known values for the Δ-relation.

exact($\phi_{\mathcal{G}}, \phi_{\mathcal{W}}$) This denotes that the goal and the Web service description are semantically identical, i.e. $\phi_{\mathcal{G}} \equiv \phi_{\mathcal{W}}$. Here, the goal can be resolved by the Web service without any impacts on the result. There is no semantic difference between $\phi_{\mathcal{G}}$ and $\phi_{\mathcal{W}}$, as there does not exists any Σ-interpretation that is a model of one description but not the other. Hence, $\delta_1 = \delta_2 = \mathit{false}$ so that $\Delta_{\phi_{\mathcal{G}}, \phi_{\mathcal{W}}} = (\mathit{true}, \mathit{true})$.

[3] These have been presented in [21] for Web service discovery on basis of DL-descriptions, used in [26] for matching in- and outputs of OWL-S profiles, defined in [16] as the basis for Web service discovery in WSMO and applied in [34], and used in [7] for candidate detection in Web service composition; each work defines them in terms of the respective specification language used.

plugin($\phi_\mathcal{G}, \phi_\mathcal{W}$) This degree denotes that the goal logically entails the Web service description, i.e. $\phi_\mathcal{G} \models \phi_\mathcal{W}$ such that $Mod_\Sigma(\phi_\mathcal{G}) \subseteq Mod_\Sigma(\phi_\mathcal{W})$. In this situation, the Web service can be used to solve the goal under the condition that the inputs provided for invoking the Web service result in an Σ-interpretation that is a model of $\phi_\mathcal{G}$. Hence, the client needs to ensure that appropriate inputs are provided to the Web service in order to successfully solve the desired goal.For the semantic difference, it holds that $\delta_2 = \neg\phi_\mathcal{G} \wedge \phi_\mathcal{W} = \mathit{false}$ so that the $\Delta_{\phi_\mathcal{G},\phi_\mathcal{W}} = (\neg\delta_1, \mathit{true})$. [4]

subsume($\phi_\mathcal{G}, \phi_\mathcal{W}$) As the opposite to the plugin degree, this denotes that the Web service description entails the goal, i.e. $\phi_\mathcal{W} \models \phi_\mathcal{G}$ such that $Mod_\Sigma(\phi_\mathcal{W}) \subseteq Mod_\Sigma(\phi_\mathcal{G})$. In this situation, every Σ-interpretation that is returned as a result of invoking $\mathcal{W}$ always satisfies $\mathcal{G}$. Hence, this degree guarantees successful resolution of the goal without any impacts or conditions on the result. However, there may still be impacts on how the goal is resolved because the possible results of executing the Web service merely denote a subset of Σ-interpretations that satisfy the goal. For the semantic difference, it holds that $\delta_1 = \phi_\mathcal{G} \wedge \neg\phi_\mathcal{W} = \mathit{false}$ so that in this situation $\Delta_{\phi_\mathcal{G},\phi_\mathcal{W}} = (\mathit{true}, \neg\delta_2)$.

intersect($\phi_\mathcal{G}, \phi_\mathcal{W}$) This degree denotes that there is no proper logical entailment relation but there exists at least one Σ-interpretation that is a common model for $\phi_\mathcal{G}$ and $\phi_\mathcal{W}$, formally: $\exists\mathcal{I}(\Sigma).\ \mathcal{I}(\Sigma) \in Mod_\Sigma(\phi_\mathcal{G}) \wedge \mathcal{I}(\Sigma) \in Mod_\Sigma(\phi_\mathcal{W}) \wedge \neg(Mod_\Sigma(\phi_\mathcal{G}) \subseteq Mod_\Sigma(\phi_\mathcal{W}) \vee Mod_\Sigma(\phi_\mathcal{G}) \supseteq Mod_\Sigma(\phi_\mathcal{W}))$. Here, the Web service can be used for solving the goal under the condition that the inputs for invoking $\mathcal{W}$ result in provision of a Σ-interpretations that satisfy the goal; there might be only one invocation of the Web service that allows solving the goal. Hence, this is the weakest matchmaking situation that allows using the Web service for solving a given goal. Regarding the semantic difference, the Δ-relation needs to explicate restrictions on both the goal and the Web service description that allow establishing logical equivalence so that $\Delta_{\phi_\mathcal{G},\phi_\mathcal{W}} = (\neg\delta_1, \neg\delta_2)$.

disjoint($\phi_\mathcal{G}, \phi_\mathcal{W}$) The fifth and final matchmaking degree denotes that the goal and Web service description are disjoint, i.e. $\neg\exists\mathcal{I}(\Sigma).\ \mathcal{I}(\Sigma) \in Mod_\Sigma(\phi_\mathcal{G}) \wedge \mathcal{I}(\Sigma) \in Mod_\Sigma(\phi_\mathcal{W})$. Obviously, in this situation the Web service can not be used for solving the goal - also not by refining one of the descriptions. Although this situation is not of interest for Web service detection, we can still represent the semantic difference. Here, with respect to the degree definition, it holds that $\delta_1 = \phi_\mathcal{G} \wedge \neg\phi_\mathcal{W} = \mathcal{G}$ and $\delta_2 = \neg\phi_\mathcal{G} \wedge \phi_\mathcal{W} = \mathcal{W}$ so that $\Delta_{\phi_\mathcal{G},\phi_\mathcal{W}} = (\neg\mathcal{G}, \neg\mathcal{W})$.

Furthermore, there are logical relationships between the matchmaking degrees. In particular, it holds that (1) if `plugin`$(\phi_\mathcal{G}, \phi_\mathcal{W})$ and `subsume`$(\phi_\mathcal{G}, \phi_\mathcal{W})$, then `exact`$(\phi_\mathcal{G}, \phi_\mathcal{W})$, and (2) - under the assumption that all goal and Web service descriptions are consistent formulae - if `plugin`$(\phi_\mathcal{G}, \phi_\mathcal{W})$ then `intersect`$(\phi_\mathcal{G}, \phi_\mathcal{W})$ as well as if `subsume`$(\phi_\mathcal{G}, \phi_\mathcal{W})$ then `intersect`$(\phi_\mathcal{G}, \phi_\mathcal{W})$. Accordingly, [16] defines the following precedence of the matchmaking degrees: $exact \preceq plugin, subsume \prec intersect$. Naturally, this is also reflected in the relationship of the structure of Δ-relations for the distinct matchmaking notions. When grouping the degrees into levels P_x such that $P_1 =$ `exact`$(\phi_\mathcal{G}, \phi_\mathcal{W})$, $P_2 =$ texttttplugin$(\phi_\mathcal{G}, \phi_\mathcal{W})$ and `subsume`$(\phi_\mathcal{G}, \phi_\mathcal{W})$, $P_3 =$ `intersect`$(\phi_\mathcal{G}, \phi_\mathcal{W})$, and $P_4 =$ `disjoint`$(\phi_\mathcal{G}, \phi_\mathcal{W})$, then the structure of the Δ-relations follow a strict entailment order from the lower level to the next higher level. Figure 4 illustrates this correlation with the directed arrows $\longrightarrow$ denoting the strict entailment ordering of Δ-relations, and Theorem 2 underpins this formally.

[4] proof for $\delta_2 = \neg\phi_\mathcal{G} \wedge \phi_\mathcal{W} = \mathit{false}$:

$(\phi_\mathcal{W} \Rightarrow \phi_\mathcal{G}) \Rightarrow (\neg\phi_\mathcal{G} \wedge \phi_\mathcal{W} \Leftrightarrow \mathit{false})$ $\Leftrightarrow$ $(\neg\phi_\mathcal{W} \vee \phi_\mathcal{G}) \Rightarrow ((\neg\phi_\mathcal{G} \wedge \phi_\mathcal{W} \wedge \mathit{false}) \vee (\neg(\neg\phi_\mathcal{G} \wedge \phi_\mathcal{W}) \wedge \neg\mathit{false})$
$\Leftrightarrow$ $\neg(\neg\phi_\mathcal{W} \vee \phi_\mathcal{G}) \vee (\neg(\neg\phi_\mathcal{G} \wedge \phi_\mathcal{W}) \wedge \mathit{true})$ $\Leftrightarrow$ $(\phi_\mathcal{W} \wedge \neg\phi_\mathcal{G}) \vee \phi_\mathcal{G} \vee \neg\phi_\mathcal{W}$
$\Leftrightarrow$ $(\phi_\mathcal{W} \vee \phi_\mathcal{G} \vee \neg\phi_\mathcal{W}) \wedge (\neg\phi_\mathcal{G} \vee \phi_\mathcal{G} \vee \neg\phi_\mathcal{W})$ $\Leftrightarrow$ true □

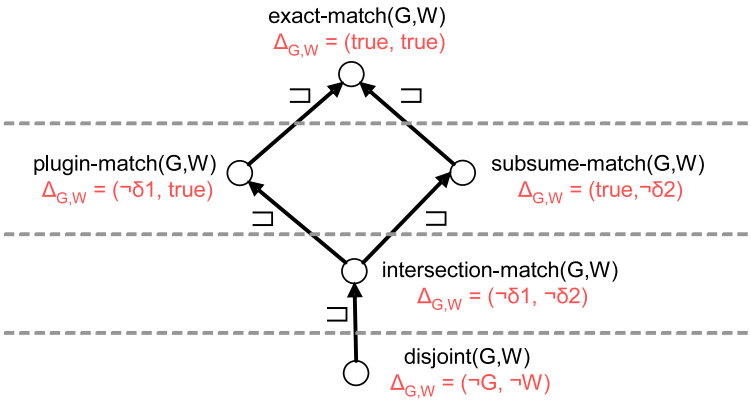

Figure 4: Delta Relations and Semantic Matchmaking Notions - Alignment and Correlations

Theorem 2 (Strict Entailment of Delta Relations on Matchmaking Levels)**.** *Let $\phi_{\mathcal{G}}$ be a formula denoting a functionality requested in a goal and $\phi_{\mathcal{W}}$ be a formula denoting a functionality provided by a Web service; let the semantic matchmaking notions be arranged in precedence levels P such that:*

$P_1 =$ `exact(`$\phi_{\mathcal{G}}, \phi_{\mathcal{W}}$`)` $\quad P_3 =$ `intersect(`$\phi_{\mathcal{G}}, \phi_{\mathcal{W}}$`)`
$P_2 =$ `plugin(`$\phi_{\mathcal{G}}, \phi_{\mathcal{W}}$`), subsume(`$\phi_{\mathcal{G}}, \phi_{\mathcal{W}}$`)` $\quad P_4 =$ `disjoint(`$\phi_{\mathcal{G}}, \phi_{\mathcal{W}}$`),`

then the Δ-relations on the precedence levels denote a strict entailment ordering:

$$\Delta_{P_x}(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}) \sqsubset \Delta_{P_{x-1}}(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}), \;\; x \leq 2 \leq n \leq 4$$

Proof. As one condition, Definition 4 defines a strict entailment order of pairs of formulae $(\phi_1, \phi_2) \sqsubset (\phi'_1, \phi'_2)$ to be given if $([\phi_1]_{\equiv_{\mathfrak{L}}} \preceq [\phi'_1]_{\equiv_{\mathfrak{L}}}$ *and* $[\phi_2]_{\equiv_{\mathfrak{L}}} \prec [\phi'_2]_{\equiv_{\mathfrak{L}}})$. Further, we recall that in logics with model-theoretic semantics logical entailment of two formulae $\phi_1 \models \phi_2$ is given if $\phi_1 \Rightarrow \phi_2$.

We first show strict entailment ordering for $\Delta_{disjoint} = (\neg\mathcal{G}, \neg\mathcal{W}) \sqsubset \Delta_{intersect} = (\neg\delta_1, \neg\delta_2)$ with $\delta_1 = \phi_{\mathcal{G}} \wedge \neg\phi_{\mathcal{W}}$ and $\delta_2 = \neg\phi_{\mathcal{G}} \wedge \phi_{\mathcal{W}}$. For the first part of the condition, it has to hold that $\neg\phi_{\mathcal{G}} \models \neg\delta_1$ which we can prove as follows: $(\neg\phi_{\mathcal{G}} \Rightarrow \neg(\phi_{\mathcal{G}} \wedge \neg\phi_{\mathcal{W}})\ ) \Leftrightarrow (\phi_{\mathcal{G}} \vee \neg\phi_{\mathcal{G}} \vee \phi_{\mathcal{W}}) \Leftrightarrow true$. For the second part, it has to hold that $\neg\phi_{\mathcal{W}} \models \neg\delta_2$ and $\neg\phi_{\mathcal{W}} \neq \neg\delta_2$. While the latter is trivial, we can prove the former by: $(\neg\phi_{\mathcal{W}} \Rightarrow \neg(\neg\phi_{\mathcal{G}} \wedge \phi_{\mathcal{W}})\ ) \Leftrightarrow (\phi_{\mathcal{W}} \vee \phi_{\mathcal{G}} \vee \neg\phi_{\mathcal{W}}) \Leftrightarrow true$. Hence, $\Delta_{P_4}(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}) \sqsubset \Delta_{P_3}(\phi_{\mathcal{G}}, \phi_{\mathcal{W}})$.

Next, we show strict entailment ordering for $\Delta_{intersect} = (\neg\delta_1, \neg\delta_2) \sqsubset \Delta_{plugin} = (\neg\delta_1, true)$. For the first part of the condition, it trivially holds that $\neg\delta_1 \models \neg\delta_1$. For the second part, it has to hold that $\neg\delta_2 \models true$ and $\neg\delta_2 \neq true$. While the latter is trivial, the former can be proved by $(\neg\delta_2 \Rightarrow true\ ) \Leftrightarrow (\delta_2 \vee true) \Leftrightarrow true$. Similarly, it holds that $\Delta_{intersect} = (\neg\delta_1, \neg\delta_2) \sqsubset \Delta_{subsume} = (true, \neg\delta_2)$ so that $\Delta_{P_3}(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}) \sqsubset \Delta_{P_2}(\phi_{\mathcal{G}}, \phi_{\mathcal{W}})$.

Finally, we show $\Delta_{P_2}(\phi_{\mathcal{G}}, \phi_{\mathcal{W}}) \sqsubset \Delta_{P_1}(\phi_{\mathcal{G}}, \phi_{\mathcal{W}})$. For $\Delta_{subsume} = (true, \neg\delta_2) \sqsubset \Delta_{exact} = (true, true)$ we can apply the condition for strict entailment ordering used above. For the first part, it trivially holds that $true \models true$; for the second part, $\neg\delta_2 \models true$ can be shown by $(\neg\delta_2 \Rightarrow true) \Leftrightarrow (\delta_2 \vee true) \Leftrightarrow true$ and $\neg\delta_2 \neq true$ trivially holds. For $\Delta_{plugin} = (\neg\delta_1, true) \sqsubset \Delta_{exact} = (true, true)$ we need to apply the other conditions for strict entailment ordering defined in Definition 4: $([\phi_1]_{\equiv_{\mathfrak{L}}} \prec [\phi'_1]_{\equiv_{\mathfrak{L}}} and [\phi_2]_{\equiv_{\mathfrak{L}}} \preceq [\phi'_2]_{\equiv_{\mathfrak{L}}})$. For the first part, it has hold that $\neg\delta_1 \models true$ and $\neg\delta_1 \neq true$. The latter is trivial, while the former can be proved by: $(\neg\delta_1 \Rightarrow true) \Leftrightarrow (\delta_1 \vee true) \Leftrightarrow true$. For the second part of the condition, it trivially holds that $true \models true$. This completes the proof.

□

2.4 Illustrative Example and Discussion

The following demonstrates and discusses the above definitions within a non-trivial example: a goal specifies the objective of finding the best restaurant in a city, and an available Web service provides a search facility for the best French restaurant in a city. We first explain the example setting with the formal functional descriptions in classical first-order logic. Then, we construct the Δ-relation between and present a technique for formulae simplification by conventional tautologies. Finally, we discuss possibilities for providing gained knowledge on the conditions for using the Web service for solving the goal to the client.

2.4.1 Example Setting and Functional Descriptions with FOL

According to the scope of the above definitions, we describe the requested and provided functionality as formulae in classical first-order logic (FOL) with the syntax defined in [11]. FOL is a specific logic $\mathfrak{L}$ with model-theoretic semantics as defined in Section 2.1 that serves as an umbrella for several ontology languages [8]. Hence, we can apply the above definitions for Δ-relations between logical formulae to FOL. We use a common structure for describing requested and provided functionalities $\mathcal{D}$ as FOL formulae:

$$\mathcal{D}(\Sigma) = in(x_1, \ldots, x_n) \wedge \phi(\vec{x}) \rightarrow (out(y) \leftrightarrow \psi(\vec{x}, y)).$$

The meaning is as follows. The functionality is specified with respect to a signature Σ that defines terminology and knowledge of the domain of discourse, which typically is defined in terms of an ontology. $in(x_1, \ldots, x_n)$ denotes possibly several inputs which typically contain datatype constraints. Commonly referred as preconditions, $\phi(\vec{x})$ defines conditions on the inputs as well as other conditions that need to hold before the functionality can be requested (goal), respectively achieved (Web service). If the inputs and conditions are true, then the outputs are true (defined as an implication $\rightarrow$). The expected or provided output is denoted by $out(y)$; this is considered to always be one object that might be an aggregation of several others, so that it is denoted by only one variable. Commonly referred to as postconditions or effects, $\psi(\vec{x}, y)$ defines conditions on the output in dependence of input. The equivalence $out(y) \leftrightarrow \psi(\vec{x}, y)$ explicitly defines the output object in a necessary ($out(y) \rightarrow \psi(\vec{x}, y)$) and sufficient ($out(y) \leftarrow \psi(\vec{x}, y)$) manner.

To properly model the requested and provided functionality in our example, we apply the notation introduced in [19] that allows representing frame-based modelling in FOL. This provides two central constructs: `memberOf(x,concept)` denotes class membership of the variable `x` to a concept, and attributes with values for concepts by `hasAttValue(x,attribute,value)`. The signature for the example defines domain terminology and knowledge about cities as well as restaurants and their types. While omitting its complete definition with respect to the scope and focus of this example, one predicate is important in the following: $better(x, y)$ denotes rating of restaurants as a partial order (i.e. $true$ if the rating for restaurant x is higher than for restaurant y). On this basis, we can describe the goal and the Web service as follows.

Goal "find best restaurant in a city"

$$\begin{aligned}\mathcal{D}_G: \quad & \forall x, y.\, in(x) \wedge memberOf(x, city) \\ & \Rightarrow (out(y) \Leftrightarrow (\\ & \quad memberOf(y, restaurant) \\ & \quad \wedge hasAttValue(y, in, x) \\ & \quad \wedge \neg\exists z.(memberOf(z, restaurant) \\ & \quad \wedge hasAttValue(z, in, x) \\ & \quad \wedge better(z, y)))).\end{aligned}$$

Web Service "provide best French restaurant in a city"

$$\begin{aligned}\mathcal{D}_W: \quad & \forall x, y.\, in(x) \wedge memberOf(x, city) \Rightarrow (out(y) \\ & \Leftrightarrow (memberOf(y, restaurant) \\ & \quad \wedge hasAttValue(y, in, x) \\ & \quad \wedge hasAttValue(y, type, french) \\ & \quad \wedge \neg\exists z.(memberOf(z, restaurant) \\ & \quad \wedge hasAttValue(z, in, x) \\ & \quad \wedge hasAttValue(z, type, french) \\ & \quad \wedge better(z, y)))).\end{aligned}$$

Examining the descriptions shows that input $in(x)$ and output objects $out(y)$ as well as preconditions $\phi(x)$ are identical, while the differences occur in the postconditions $\psi(x, y)$. Therein, the goal requests the best restaurant in a city with respect all restaurants while the Web service only considers French restaurants.

Regarding the usability of the Web service for solving the goal, there in an intersection match between $\mathcal{D}_G$ and $\mathcal{D}_W$. As discussed in Section 2.3, this is the weakest matchmaking degree wherein the client needs to provide a particular input in order to satisfy the goal by using the Web service. For intuitive understanding, consider two cities A and B with the best restaurant in A is of type French, while the best restaurant in B is of some other type. If a client instantiates the goal with A as the input, then the Web service will return the best restaurant - which is French in this special case; if client provides B as the input, then the Web service will return the best French restaurant in B; this is not the best one in B, thus the Web service is not usable for solving the goal in this case. The purpose of the following elaborations is to provide additional information to the client on the conditions under which the Web service is usable for solving the goal on basis of Δ-relations.

Before constructing the Δ-relation between $\mathcal{D}_G$ and $\mathcal{D}_W$, the following shows the proof of the intersection match within VAMPIRE. This is resolution-based theorem prover for first-order classical logic with equality [27] 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 2.3: 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 [34]: 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). [5]

```
% SIGNATURE
% better-relation is a partial order
input_formula(transitivityBetterRelation, axiom,(
 ! [R1,R2] : (
    member_of(R1, restaurant) & member_of(R2, restaurant)
    & better(R1,R2) => ~better(R2,R1) )
)).
% transitivity of better-relation
input_formula(transitivityBetterRelation, axiom,( ! [R1,R2,R3] : (
  member_of(R1, restaurant) & member_of(R2, restaurant)
  & member_of(R3, restaurant) & better(R1,R2) & better(R2,R3)
  => better(R1,R3) )
)).
```

[5] VAMPIRE 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: $\sim$, implication: $\Rightarrow$, equivalence: $\Leftrightarrow$); 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.

```
% GOAL: find best restaurant in a city
input_formula(goalin, axiom,(
  ! [X] : ( goalin(X) <=> (member_of(X, city) ) ))).
input_formula(goalout, axiom,( ! [X,Y] :
  ( goalout(Y) <=> ( member_of(Y, restaurant) & has_att_value(Y, in, X)
      & ~ ? [Z] : ( member_of(Z, restaurant) & has_att_value(Z, in, X)
      & better(Z,Y)) ) ) )).
input_formula(goaldescription, axiom,( ! [I,O] : (
  goal(I,O) <=> (goalin(I) => goalout(O) )  ) )).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% WEB SERVICE: give best French restaurant in a city
input_formula(wsin, axiom,( ! [X] : (
 wsin(X) <=> ( member_of(X, city) ) ) )).
input_formula(wsout, axiom,( ! [X,Y,Z] : (
 wsout(Y) <=> ( member_of(Y, restaurant)
   & has_att_value(Y, in, X) &  has_att_value(Y, type, french)
   & ~ ? [Z] : ( member_of(Z, restaurant) & has_att_value(Z, in, X)
      & has_att_value(Z, type, french) & better(Z,Y) ) ) )  )).
input_formula(wsdescription, axiom,( ! [I,O] : (
  ws(I,O) <=> (wsin(I) => wsout(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
```

2.4.2 Constructing the Delta Relation

The following illustrates the construction of the Δ-relation between the goal and the Web service description. In accordance to Theorem 1, this is defined as $\Delta_{\mathcal{D}_G,\mathcal{D}_W} = (\neg\delta_1, \neg\delta_2)$ with $\delta_1 = \mathcal{D}_G \wedge \neg\mathcal{D}_W$ and $\delta_2 = \mathcal{D}_W \wedge \neg\mathcal{D}_G$.

In principle, we can write down the expressions for δ_1 and δ_2 immediately by inserting the description formulae $\mathcal{D}_G$ and $\mathcal{D}_W$. However, the resulting formulae are not intuitively comprehendible for human consumption. To obtain a simpler representation, we can re-write the formulae into a more intuitive form by applying conventional FOL tautologies. For a given formula ϕ we can obtain a simpler representation ϕ' without changing the meaning of the formula, i.e $\phi \equiv \phi'$ so that all obtainable ϕ' constitute the equivalence class $[\phi]_{\equiv_{FOL}}$ (see Definition 3). Unfortunately, formula simplification with tautologies as a technique for attaining desirable representations of Δ-relations can not be automated, as the stop conditions cannot be defined in a general manner. However, we can therefore apply heuristics such as a stop point is achieved when the occurrence of symbols is minimal; we exemplify this in the following for our example. [6]

[6] The following lists the most common tautologies for simplifying some FOL formula α, β, γ:

$\alpha \rightarrow \beta \Leftrightarrow \neg\alpha \vee \beta.$	$\neg(\alpha \vee \beta) \Leftrightarrow \neg\alpha \wedge \neg\beta.$	$\alpha \wedge \beta \Leftrightarrow \beta \wedge \alpha.$	$\alpha \wedge (\beta \wedge \gamma) \Leftrightarrow (\alpha \wedge \beta) \wedge \gamma.$
$\alpha \rightarrow \beta \Leftrightarrow \neg(\alpha \wedge \neg\beta).$	$\neg(\alpha \wedge \beta) \Leftrightarrow \neg\alpha \vee \neg\beta.$	$\alpha \vee \beta \Leftrightarrow \beta \vee \alpha.$	$\alpha \vee (\beta \vee \gamma) \Leftrightarrow (\alpha \vee \beta) \vee \gamma.$
$\alpha \leftrightarrow \beta \Leftrightarrow (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha).$	$\neg\neg\alpha \Leftrightarrow \alpha$	$\alpha \vee (\alpha \wedge \gamma) \Leftrightarrow \alpha.$	$\alpha \vee (\beta \wedge \gamma) \Leftrightarrow (\alpha \vee \beta) \wedge (\alpha \vee \gamma).$
$\alpha \vee \beta \Leftrightarrow \neg(\neg\alpha \wedge \neg\beta).$	$\alpha \wedge \neg\alpha \Leftrightarrow \mathit{false}.$	$\alpha \wedge (\alpha \vee \gamma) \Leftrightarrow \alpha.$	$\alpha \wedge (\beta \vee \gamma) \Leftrightarrow (\alpha \wedge \beta) \vee (\alpha \wedge \gamma).$
$\alpha \wedge \beta \Leftrightarrow \neg(\neg\alpha \vee \neg\beta).$	$\alpha \vee \neg\alpha \Leftrightarrow \mathit{true}.$	$\alpha \wedge \alpha \Leftrightarrow \alpha.$	$\alpha \vee \alpha \Leftrightarrow \alpha.$

Table 1 shows the simplification of the expressions for δ_1, δ_2 for our example. Therefore, we re-write the goal and Web service description to prenex normal form and apply the same heuristic strategy for simplifying the formulae. Table 2 shows the obtained formulae in their simplified representation. [7] [8]

Table 1: Simplification of δ_1, δ_2 with FOL Tautologies*

$\delta_1 = \mathcal{D}_G \wedge \neg\mathcal{D}_W$	
	$(c \rightarrow ry \wedge \neg(rz \wedge b)) \wedge \neg(c \rightarrow ry \wedge fy \wedge \neg(rz \wedge fz \wedge b))$
$\Leftrightarrow$	$(\neg c \vee ry \wedge \neg(rz \wedge b)) \wedge \neg(\neg c \vee (ry \wedge fy \wedge \neg(rz \wedge fz \wedge b)))$
$\Leftrightarrow$	$(\neg c \vee (ry \wedge \neg(rz \wedge b))) \wedge (c \wedge (\neg ry \vee \neg fy \vee (rz \wedge fz \wedge b)))$
$\Leftrightarrow$	$c \wedge (\neg c \vee (ry \wedge \neg(rz \wedge b))) \wedge (\neg ry \vee \neg fy \vee (rz \wedge fz \wedge b))$
$\Leftrightarrow$	$((c \wedge \neg c) \vee (c \wedge ry \wedge \neg(rz \wedge b))) \wedge (\neg ry \vee \neg fy \vee (rz \wedge fz \wedge b))$
$\Leftrightarrow$	$c \wedge ry \wedge \neg(rz \wedge b) \wedge (\neg ry \vee \neg fy \vee (rz \wedge fz \wedge b))$
$\Leftrightarrow$	$c \wedge ((\neg ry \wedge ry \wedge \neg(rz \wedge b)) \vee (\neg fy \wedge ry \wedge \neg(rz \wedge b)) \vee (rz \wedge fz \wedge b \wedge ry \wedge \neg(rz \wedge b)))$
$\Leftrightarrow$	$c \wedge ry \wedge \neg fy \wedge \neg(rz \wedge b)$
$\delta_2 = \mathcal{D}_W \wedge \neg\mathcal{D}_G$	
	$(c \rightarrow ry \wedge fy \wedge \neg(rz \wedge fz \wedge b)) \wedge \neg(c \rightarrow ry \wedge \neg(rz \wedge b))$
$\Leftrightarrow$	$(\neg c \vee (ry \wedge fy \wedge \neg(rz \wedge fz \wedge b))) \wedge \neg(\neg c \vee (ry \wedge \neg(rz \wedge b)))$
$\Leftrightarrow$	$c \wedge (\neg c \vee (ry \wedge fy \wedge \neg(rz \wedge fz \wedge b))) \wedge (\neg ry \vee (rz \wedge b))$
$\Leftrightarrow$	$((c \wedge \neg c) \vee (c \wedge ry \wedge fy \wedge \neg(rz \wedge fz \wedge b))) \wedge (\neg ry \vee (rz \wedge b))$
$\Leftrightarrow$	$c \wedge ry \wedge fy \wedge \neg(rz \wedge fz \wedge b) \wedge (\neg ry \vee (rz \wedge b))$
$\Leftrightarrow$	$(c \wedge ry \wedge fy \wedge rz \wedge b \wedge \neg(rz \wedge fz \wedge b))$
$\Leftrightarrow$	$(c \wedge ry \wedge fy \wedge rz \wedge b \wedge \neg fz)$

* symbol substitution for better traceability:

c = $memberOf(x, city)$	ry = $memberOf(y, restaurant) \wedge hasAttValue(y, in, x)$
fy = $hasAttValue(y, type, french)$	fz = $hasAttValue(z, type, french)$
b = $better(z, y)$	rz = $memberOf(z, restaurant) \wedge hasAttValue(z, in, x)$

Table 2: Simplified Formulae for δ_1, δ_2

$\delta_1:\ \forall x, y.z.\ memberOf(x, city)$ $\wedge\ memberOf(y, restaurant)$ $\wedge\ hasAttValue(y, in, x)$ $\wedge\ \neg hasAttValue(y, type, french)$ $\wedge\ \neg(memberOf(z, restaurant)$ $\wedge\ hasAttValue(z, in, x)$ $\wedge\ better(z, y)).$	$\delta_2:\ \forall x, y.z.\ memberOf(x, city)$ $\wedge\ memberOf(y, restaurant)$ $\wedge\ hasAttValue(y, in, x)$ $\wedge\ hasAttValue(y, type, french)$ $\wedge\ memberOf(z, restaurant)$ $\wedge\ hasAttValue(z, in, x)$ $\wedge\ \neg hasAttValue(y, type, french)$ $\wedge\ better(z, y).$

[7] Here, the critical aspect for re-writing the goal and Web service description to prenex normal form are the postconditions of the form: $\forall y.\ out(y) \leftrightarrow \neg\exists z.\ \phi(z)$. In this special case, the prenex normal form $\forall y, z.\ out(y) \leftrightarrow \neg\phi(z)$ is a semantically equivalent formulae because for all possible interpretations it holds that $((\forall y.\ out(y) \rightarrow \neg\exists z.\ \phi(z)) \wedge (\forall y.\ out(y) \leftarrow \neg\exists u.\ \phi(u))\ ) \Leftrightarrow (\forall y.\ out(y) \leftrightarrow \neg\exists z.\ \phi(z))$. Hence, we can re-write the descriptions as $\forall x, y, z.\ in(x) \wedge \phi(x) \rightarrow (out(y) \leftrightarrow \phi(y) \wedge\ (\phi(z))$.

[8] simplification strategy: (1) resolve implications in the formula, (2) apply outer negation, (3) extract common symbols, (4) apply distribution: $\alpha \wedge (\beta \vee \gamma) \Leftrightarrow (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$, (5) omit trivial terms (i.e. terms that are equivalent to $false$ or $true$.

The obtained formulae denote the constituting aspects of the Δ-relation between $\mathcal{D}_G$ and $\mathcal{D}_W$. As we have strictly followed Theorem 1 for construction, the obtained formulae allow establishing logical equivalence between $\mathcal{D}_G$ and $\mathcal{D}_W$ (necessary condition) and are the minimal representation of the semantic differences (sufficient condition); VAMPIRE allows to proof the correctness.

Hence, the obtained formulae for δ_1 and δ_2 precisely denote the difference of functionality requested by the goal and provided by the Web service - δ_1 from the perspective of the goal and δ_2 from the perspective of the Web service. However, the obtained knowledge has no meaning per se; this reveals when using the Δ-relation for particular applications as we discuss in the following.

2.4.3 Applying the Delta Relation

The following discusses two possibilities for providing the knowledge gained on the conditions for utilizing the Web service for solving the goal to clients. Introduced in Section 1.2, the first one is goal refinement and adjustment that provides revised goal formulations with respect to the obtained Δ-relation, and the second one is explicating the usage conditions as an assumption.

Goal Refinement / Adjustment. The first possibility is to refine the original goal G towards a new goal G' such that the matchmaking degree between G' and the Web service W guarantees solvability of G'. As introduced above, G' is called a *refinement* of G if it strengthens but does not change the objective description such that $G' \models G$; *goal adjustment* denotes that G' changes the objective definition so that the entailment relation between G and G' is not given. Both types of goal re-formulation can be obtained by the same techniques, notably with Δ-relations as we exemplify in the following.

In the example, the original goal G is only solvable by the Web service W if specific cities are provided as input (namely cities whose best restaurant is French). As discussed in Section 2.3, solvability of a goal is guaranteed if $\mathcal{D}_W \models \mathcal{D}_G$, i.e. if the matching degree is `subsume(G,WS)` or `exact(G,WS)`. Hence, if for our example we can obtain a new goal description G' such that $\mathcal{D}_W \models \mathcal{D}_{G'}$, then successful solvability of G' by the Web service is assured. For determining a new goal G', let's consider the relationship of the goal description $\mathcal{D}_G$, the Web service description $\mathcal{D}_W$, and δ_1, δ_2 as the constituting elements of the Δ-relation between $\mathcal{D}_G$ and $\mathcal{D}_W$. With respect to its definition, it holds that $\mathcal{D}_G \wedge \neg\delta_1 \equiv \mathcal{D}_W \wedge \neg\delta_2$ (see Theorem 1) and $\mathcal{D}_G \wedge \neg\delta_1 \models \mathcal{D}_G$ and $\mathcal{D}_W \wedge \neg\delta_2 \models \mathcal{D}_W$ (see Proposition 4). Hence, a goal description $\mathcal{D}_{G'} = \mathcal{D}_G \wedge \neg\delta_1$ is a refinement of the original goal that is solvable by the Web service under a plugin-matching degree because $\mathcal{D}_G \wedge \neg\delta_1 \models \mathcal{D}_W$.

When simplifying the formula for $\mathcal{D}_G \wedge \neg\delta_1$ with FOL tautologies as illustrated above, we obtain a goal that requests French restaurants with respect to all restaurants in a city that is provided as input (see left column in Table 3). However, this goal formulation does not have a model in case that a city is provided as input whose best restaurant is not French (if for a given city there is a restaurant that is better than the best French restaurant, then the postcondition of $\mathcal{D}_{G'}$ evaluates to $false$). As such a goal description is not desirable, we can construct a goal formulation G'' that has a model for the output object for all possible cities provided as input by restricting the restaurant types to be considered to French. This goal description is equivalent to the Web service description, i.e. $\mathcal{D}_{G''} \equiv \mathcal{D}_W$. Hence, the matching degree is `exact(G'',W)`, which is a desirable matchmaking notion as the goal can be resolved without any impacts or conditions for Web service usage. However, G'' is no longer a refinement of the original goal because $G'' \not\models G$ (see the discussion on the intersection match between $\mathcal{D}_G$ and $\mathcal{D}_W$ above), but represents an adjusted goal. Table 3 shows both G' and G''. Below, we provide the correctness proof of the plugin-match between $\mathcal{D}_{G'}$ and $\mathcal{D}_W$ in VAMPIRE while omitting the trivial proof for `exact(G'',W)`.

Table 3: Refined and Adjusted Goal Descriptions

refined goal G': $\mathcal{D}_{G'} = \mathcal{D}_G \wedge \neg\delta_1$	adjusted goal G''
$\mathcal{D}_{G'}:\ \forall x, y.z.\ in(x) memberOf(x, city)$ $\Rightarrow (out(y)$ $\Leftrightarrow (memberOf(y, restaurant)$ $\wedge\ hasAttValue(y, in, x)$ $\wedge\ hasAttValue(y, type, french)$ $\wedge \neg(memberOf(z, restaurant)$ $\wedge\ hasAttValue(z, in, x)$ $\wedge\ better(z, y))))$	$\mathcal{D}_{G''}:\ \forall x, y.\ in(x) \wedge memberOf(x, city) \Rightarrow (out(y)$ $\Leftrightarrow (memberOf(y, restaurant)$ $\wedge\ hasAttValue(y, in, x)$ $\wedge\ hasAttValue(y, type, french)$ $\wedge \neg\exists z.(memberOf(z, restaurant)$ $\wedge\ hasAttValue(z, in, x)$ $\wedge\ hasAttValue(z, type, french)$ $\wedge\ better(z, y)))).$

```
% PROOF: goal' <=> goal & ~delta1
input_formula(po, conjecture,(
(goalRefined <=> (! [X,Y,Z] : (
  member_of(X, city) => (
    member_of(Y, restaurant) &
      has_att_value(Y, in, X) & has_att_value(Y, type, french) &
  ~(member_of(Z, restaurant) & has_att_value(Z, in, X) &
      better(Z,Y) ) )  )))
& (goal & ~delta1Simplified) <=> goalRefined )).
%% PROVED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PROOF: goal' |= ws
input_formula(po, conjecture,( goalRefined => ws )).
%% PROVED
```

Assumption Explication The second possibility for explicating the conditions of using W for solving the original goal G is to represent this as an assumption. As discussed in the introduction, explication of hidden assumption is concerned with determining additional constraints for Web service usage. We can understand the conditions under which a Web service is usable for solving a goal as a specific type of assumptions.

Within our example, the Web service W is usable for the original goal G if the best restaurant in the city that is provided as input in of type French. For all other cities, the executing W will not provide the best restaurant in the city as requested by G. Hence, if we find a formulae $asmt$ such that $asmt \Rightarrow (\mathcal{D}_W \Rightarrow \mathcal{D}_G)$, then $asmt$ explicitly denotes the conditions that need to hold for W to be usable for solving G.

As explained above, the only situation wherein successful resolution of the goal without any implications on the result is `subsume(G,WS)`. For $\mathcal{D}_G, \mathcal{D}_W$ being FOL formulae, this is given if $\mathcal{D}_W \models \mathcal{D}_G$ which we can also specify as $\mathcal{D}_W \Rightarrow \mathcal{D}_G$. To express the usage conditions as an assumption $asmt$ such that $asmt \Rightarrow (\mathcal{D}_W \Rightarrow \mathcal{D}_G)$, it has to hold that $asmt$ restricts the domain such that there can not be any interpretation $\mathcal{I}(\Sigma, asmt)$ that is a model for $\mathcal{D}_W$ but not for $\mathcal{D}_G$. By definition, $\delta_2 = \neg\mathcal{D}_G \wedge \mathcal{D}_W$ exactly denotes all interpretations $\mathcal{I}(\Sigma)$ that are models of $\mathcal{D}_W$ but not of $\mathcal{D}_G$. As $asmt$ needs to ensure that these are not valid for using W for solving G, the desired assumption is the negation of δ_2, hence $asmt = \neg\delta_2$.

Table 4 shows the transformation of the negation of δ_2 that states for all cities it has to hold that the best restaurant is of type French – which formalizes our intuitive observation. Below, we provide the proofs with VAMPIRE for $asmt \Rightarrow (\mathcal{D}_W \Rightarrow \mathcal{D}_G)$.

Table 4: Assumption that explicates the condition for using W to solve G

δ_2	$asmt = \neg\delta_2$
$\delta_2: \quad \forall x, y.z.\ memberOf(x, city)$ $\land\ memberOf(y, restaurant)$ $\land\ hasAttValue(y, in, x)$ $\land\ hasAttValue(y, type, french)$ $\land\ memberOf(z, restaurant)$ $\land\ hasAttValue(z, in, x)$ $\land\ \neg hasAttValue(y, type, french)$ $\land\ better(z, y).$	$asmt: \quad \forall x, y.\ memberOf(x, city)$ $\land\ memberOf(y, restaurant)$ $\land\ hasAttValue(y, in, x)$ $\land\ hasAttValue(y, type, french)$ $\rightarrow\ \neg\exists z.\ (memberOf(z, restaurant)$ $\land\ hasAttValue(z, in, x)$ $\land\ \neg hasAttValue(y, type, french)$ $\land\ better(z, y)).$

```
% proof obligation for ~delta2 implies WS |= Goal
input_formula(po, conjecture,( ~delta2Simplified => (ws => goal) )).
%% PROVED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% proof obligation for asmt <=> ~delta2
input_formula(po, conjecture,(
( asmt <=> (! [X,Y,Z] : (
   memberOf(X, city) &
   memberOf(Y, restaurant) &
     has_att_value(Y, in, X) &
     has_att_value(Y, type, french)
   => ~ ? [Z] :(
   memberOf(Z, restaurant) &
       has_att_value(Z, in, X) &
      ~has_att_value(Y, type, french) &
     better(Z,Y) )
 ) ))
 & (asmt <=> ~delta2Simplified) )).
%% PROVED
```

2.5 Summary

This section encompasses several central aspects on the definition, construction, and usage of Δ-relations that we summarize in the following for better traceability. We have considered functional descriptions that are defined by conventional logical formulae. More precisely, we have considered formula in logics with model-theoretic semantics that are used for formally specifying requested and provided functionalities. After recalling the relevant aspects of logics with model-theoretic semantics, we introduced the symmetric difference as the straight forward notion for denoting the semantic difference. For two formulae ϕ_1, ϕ_2 defined over a signature Σ, the symmetric difference precisely denotes those Σ-interpretations that are models for either of the formulae but not common to them, formally: $(\phi_1 \land \neg\phi_2) \lor (\neg\phi_1 \land \phi_2)$.

For Web service detection as the application context of Δ-relations we are interested in very specific aspect of the semantic difference between the formal functional descriptions of a goal and a Web service. In particular, we are interest in the differences that hamper usability of a Web service for solving a goal,

respectively that cause certain impacts on how the goal is resolved. Therefore, we have defined the Δ-relation between a formula ϕ_G that describes a goal and a formula ϕ_W that describes a Web service as a pair of formula $\Delta_{\phi_G,\phi_W} = (\neg\delta_1, \neg\delta_2)$ with $\delta_1 = \phi_G \wedge \neg\phi_W$ and $\delta_2 = \neg\phi_G \wedge \phi_W$. We have proven that this definition satisfies the requirements of *completeness* (by allowing to establish logical equivalence as $\phi_G \wedge \neg\delta_1 \Leftrightarrow \phi_W \wedge \neg\delta_2$), and *minimality* (there does not exists any Δ' that properly denotes the semantic difference and is more general than Δ). Next, we have explained the alignment of Δ-relations with the semantic matchmaking degrees commonly used for Web service detection. The central aspect is that there are pre-known values for δ_1, δ_2 for each matchmaking degree that allow more sophisticated handling; we have proven that the structure of Δ-relations on ascending levels of matchmaking follow a strict entailment ordering. These definitions hold for all logics that have model-theoretic semantics, especially for classical first-order logic and its decidable subsets (Description Logics and Horn Logics).

Throughout the illustrative example and its discussion, we have introduced several relevant aspects. At first, we have defined a meta-structure of FOL-formulae for formally describing functionalities: $\mathcal{D}(\Sigma) = in(x_1, \ldots, x_n) \wedge \phi(\vec{x}) \rightarrow (out(y) \leftrightarrow \psi(\vec{x}, y))$. This consists of inputs and preconditions that imply the output object and the postconditions. Secondly, we have utilized a modelling approach that – on basis of two central constructs: `memberOf(x,concept)` and `hasAttValue(x,attribute,value)` – allows representing frame-based knowledge definitions in FOL. Thirdly, we have utilized FOL-tautologies for simplifying formulae for Δ-relations as well as related aspects. This allows to attain simpler, human understandable representation of a formulae without changing its meaning; however, it cannot be automated and thus requires human intervention.

We have shown elaborated definitions allow obtaining correct Δ-relations. Moreover, we have exemplified how to beneficially apply these for obtaining refined or adjusted goal descriptions that are solvable by a given Web service with more desirable matchmaking degrees. Such refined goals G' can be obtained by neglecting those aspects that constitute the semantic difference from the perspective of the goal, i.e. $\mathcal{D}'_G = \mathcal{D}_G \wedge \neg\delta_1$. We also have shown how to explicate the conditions of using a Web service for successful resolution of a goal in form of an assumption $asmt$ such that $asmt = \neg\delta_2$. However, the declarative representation of the gained knowledge as human understandable formulae requires human intervention.

3 Delta Relations for Functional Descriptions

In this section we extend the elaborated definitions and techniques for Δ-relations to functional descriptions described in terms of preconditions and effects. To put it briefly, we argue and show that the above definitions are straight forward applicable to functional descriptions as well.

The section is structured as follows. Commencing with the structure, usage, and central properties of formal functional descriptions, we recall Abstract State Spaces as a language independent formal model that allows to define the semantics of functional descriptions in a sophisticated manner. Then, we discuss possibilities for defining Δ-relations for functional descriptions. The central argument is that functional descriptions defined in terms of preconditions and effects can properly be represented as single, conventional logical expressions. Hence, we can apply the above definitions straight forward for constructing Δ-relations for functional descriptions. Finally, we outline the integration of Δ-relations into the WSMO mediation framework as a technique for handling functional differences between goals and Web services.

3.1 Functional Descriptions – Properties and Semantics

Functional descriptions are a central element of frameworks for semantically describing Web services. Within the most prominent frameworks (here: those approaches submitted to the W3C as standardization proposals), they are used as a black box description of normal runs of a Web service with respect to the overall functionality provided (e.g. OWL-S Service Profiles [22] or WSMO capabilities [20]) or the functionality of operations for service consumption (e.g. in WSDL-S for semantically describing WSDL operations [1] or in SWSF for formal process descriptions [3]). For goal formulation, formal functional descriptions are used for specifying the functionality requested for achieving a client objective; therein, a goal represents a generic objective descriptions that is instantiated with concrete inputs for denoting concrete client objectives [33].

Examining the structure of functional descriptions within the different approaches reveals that they commonly follow the description model of preconditions and effects. Introduced in [13], this approach has been widely used for declaratively describing functionalities within different AI sub-disciplines (see [17],[35] for extensive studies). Informally, the meaning is that if the precondition is satisfied, then the effect will be achieved. For provided functionalities (e.g. Web services), this formally describes all possible executions; for requested functionalities in goal formulations, this defines all changes of the world that result in achieving the client objective. Although the precondition-effect description model appears to be intuitive and straight forward, there are critical aspects that need to be taken into account for formally specifying the semantics of functional descriptions. We discuss them in the following and recall a sophisticated formal description model that has specifically been defined for the context of Web services.

3.1.1 Central Properties

The first aspect for formally describing functionalities in terms of preconditions and effects is the underlying model of the world. Most approaches rely on a state-based model, meaning that the world is understood as a dynamic environment that is changed by the execution of Web services. Every state s is a static snapshot of the world whereby the current state s_c is defined by all facts and rules that are true at the current point of time. In general, a Web service execution results in a sequence $\tau = (s_0, \ldots, s_m)$ of state transitions in the world with s_0 as the start state and s_m as the termination state. A functional description $\mathcal{D}$ formally describes all possible executions of a Web service such that its *precondition* is a state constraint for the initial state s_0, and its *effect* a state constraint for the termination state s_m. Similar, a goal is formulated in an initial state s_i and is resolved if a state of the world s_g is reached wherein the objective is achieved. When describing

this by a functional description with its precondition as a state constraint on the initial state s_i and its *effect* as a state constraint for the final desired state s_g, then such a goal formulation denotes a reusable objective specification that can be instantiated with concrete data for expressing a specific client request (this follows the concept of task descriptions in UPML [10], see [33] for details).

In order to allow specification of inference mechanisms for Web service detection and other techniques, a precise and unambiguous formal semantics of such functional descriptions is needed. While state constraints (i.e. the logical expressions that denote preconditions or effects) are commonly specified in some static knowledge or ontology langauge $\mathfrak{L}$, the meaning and relationship of the specification elements of functional descriptions needs to be defined formally. In this respect, the following two aspects need to be taken into consideration:

1. **Effects depend on the precondition.** In general, a functional description can describe several possible executions or changes of the world. Thereby, the achievable state of the world (described by the effect) depends on the state of the world wherein the functionality has been invoked (described by the precondition). For instance, imagine a Web service W that provides standard arithmetic calculations. If W is invoked with 2 numbers a, b and the multiplication functionality has been chosen, then the result of executing W will be the product of a and b; if the division functionality has been chosen, then the result will be the quotient of the inputs.

2. **Shared objects in pre- and post-states.** The objects of the world that are constraint by the precondition and the effect most commonly are related and dependent of each other. For instance consider the multiplication facility of W: if $c = a * b$ specifies the effect, then this correlates to the numbers a, b that the precondition requires as input for invoking W. A formal model for describing functionalities needs to allow specifying such shared objects.

3.1.2 Abstract State Spaces – A Formal Model

The following recalls so-called *Abstract State Spaces*, short ASS. This is a formal model for Web services and the world they act in that allows to specify the formal semantics of functional descriptions with respect to the critical aspects discussed above; commonly, the description frameworks for semantically describing Web services lack of an unambiguous and precise definition in this respect. Presented in [18], the approach aims at overcoming this by presenting a language independent formal model for Web services with special attention to functional descriptions. While referring to the paper as well as to its extended version [17] for details and definitions, we here briefly recall the aspects relevant in our context.

The ASS model assumes that Web services act in a state-based world, as discussed above. This is formally defined so that each state s is a static snapshot of the world that is described on basis of a signature Σ and some domain knowledge Ω. Within ontology-based settings like Semantic Web services, Σ and Ω are usually defined in terms of ontologies. The universe of the ASS world are all interpretations $\mathcal{I}(\Sigma, \Omega)$ denotes all valid with regard to the signature and domain knowledge - which refers to all possible ontology instances within ontology-based settings. A statement ϕ denotes a state constraint which is satisfied by a state s if there exists at least one interpretation $\mathcal{I}(\Sigma, \Omega) \in s$ that is a model of ϕ. The execution of a Web service for some concrete input represents a sequence $\tau = (s_0, \ldots, s_m)$ of states. Similar, the requested functionality is understood as a desired sequence of state transitions $\tau = (s_i, \ldots, s_g)$ that allows to traverse from the initial state s_i of the goal formulation to the desired final state s_g (see [33]). The purpose of functional descriptions is to properly describe all possible executions of Web services, respectively all possible state transition sequences that all resolving a goal.

To properly define functional descriptions and their formal meaning, the ASS model defines several extensions to the signature Σ. Subsets of Σ allow to explicitly denote symbols with specific characteristics in a functional description: Σ_S denotes *static symbols* that are interpreted the same way in each state s' that is a successor of s, i.e. symbols that are not changed by the execution of a Web service; Σ_D denotes *dynamic symbols* that are explicitly changed by the execution, and Σ_D^{pre} is the set of pre-variants α_{pre} of symbols $\alpha \in \Sigma_D$ that are interpreted in each state s' as in the initial state s_0. Moreover, the ASS model defines a symbol out for denoting the objects that are provided as outputs from a Web service execution, and the notion of so-called capability interfaces $IF = (i_1, \ldots, i_n)$ that denotes all inputs values required for invoking a Web service, respectively for instantiating a goal description. On basis of this, we can define a functional description as follows.

Definition 6. *In an Abstract State Space $\mathcal{A}$ wherein $\omega(s)$ assigns Σ-interpretations to a state s, a **Functional Description** is described as a 7-tuple $\mathcal{D} = (\Sigma, \Omega, \Sigma_S, \Sigma_D, IF, \phi^{pre}(\Sigma, \Omega), \phi^{eff}(\Sigma, \Omega))$ such that:*

- *(i)* Σ *is the signature that defines symbols of a formal language* $\mathfrak{L}$
- *(ii)* $\Omega \subseteq \mathfrak{L}(\Sigma)$ *denotes formalized terminology and knowledge of the domain*
- *(iii)* Σ_S *denotes static symbols such that for all* $s, s' \in \mathcal{A}$ *and* $\alpha \in \Sigma_S : meaning(s)(\alpha) = meaning(s')(\alpha)$
- *(iv)* Σ_D *denotes dynamic symbols such that for all* $s_0, s' \in \mathcal{A}$ *and* $\alpha \in \Sigma_D : meaning(s_0)(\alpha) = meaning(s')(\alpha_{pre})$
- *(v)* IF *is a set of universally quantified variables* $i_1, \ldots, i_n$ *that denote all required inputs and whose scope is the complete Functional Description* $\mathcal{D}$
- *(vi)* $\phi^{pre}_{\Sigma,\Omega}$ *is a state constraint in* $\mathfrak{L}$ *for the initial state* s_0 *with* $i_1, \ldots, i_n$ *as free variables*
- *(vii)* $\phi^{eff}_{\Sigma,\Omega}$ *is a state constraint in* $\mathfrak{L}$ *for the final state* s_m *with* $i_1, \ldots, i_n$ *as free variables*

with the meaning that if $\omega(s_0) \models_{\mathfrak{L}(\Sigma)} \phi^{pre}$ *then* $\omega(s_m) \models_{\mathfrak{L}(\Sigma)} \phi^{eff}$.

In essence, the definition states that if the current state of the world satisfies the precondition, then the execution of the functionality will result in a state of the world that satisfies the effect. Clause (i) requires a functional description to be defined over some signature, under consideration of formalized domain knowledge (clause (ii)). In ontology-based settings, these two elements are usually provided by an ontology such that the used ontology language denotes the signature Σ and its concepts, attributes, relations, axioms, and instances provide the formalized domain knowledge Ω. Clauses (iii) and (iv) specify the signature extensions for static and dynamic symbols; we exemplify their usage below. Clause (v) denotes a set of named variables $i_1, \ldots, i_n$ that denote place holders for the input required for invoking a Web service described by $\mathcal{D}$, respectively for instantiating a goal described by $\mathcal{D}$. Clauses (vi) and (vii) specify the precondition and effect to defined as logical expressions in a static knowledge representation language $\mathfrak{L}$ for $i_1, \ldots, i_n$ as free variables. Clause (v) defines $i_1, \ldots, i_n$ as universally quantified variables whose scope is the complete functional description, which allows to explicitly specify the shared objects between the pre- and post-state of $\mathcal{D}$. The output objects are explicitly defined in $\phi^{eff}_{\Sigma,\Omega}$ by using the out symbol.

One may argue that it should be allowed that both preconditions and effects can also be defined over variables that are not explicitly required as inputs (here: $i_1, \ldots, i_n$). However, the ASS model is restricted to deterministic Web services, i.e. functionalities that provide certain results for specific inputs. If one would model that a Web service execution results in existence of some object that is not dependent on an input value, this would mean that the Web service arbitrarily creates objects in the world. As such scenarios are not desirable in the context of Web services, the above definitions prohibits this. For illustration and clarification, the following exemplifies the definition. As the signature, we FOL as the specification language $\mathfrak{L}$ for static knowledge.

Example: a purchase contract for a product
Ω: purchase-ontology
$IF : \{p\}$
$\phi^{pre} : product(p).$
$\phi^{eff} : \forall x.\, out(x) \leftrightarrow purchaseContract(x) \wedge for(x, p).$

The first example shows the functional description of a purchase Web service. It provides a purchase contract for a product that is given as input. A domain ontology purchase-ontology provide the signature and domain knowledge definition (among others, this contains the symbols used in the precondition and effect formulae). The only variable $\in IF$ is p. The precondition ϕ^{pre} restricts this to be a of type product, meaning that the pre-state for invoking the Web service is given if there exists a variable assignment for p. Note that because all variable $\in IF$ are implicitly all-quantified, p appears as a free variable in ϕ^{pre}. The effect ϕ^{eff} states that there all output objects are purchase contracts for the product provided as input. The output object is the contract, which is explicitly denoted by the out symbol; as within the example discussed in Section 2.4, the output object is denoted by one variable and connected to the post-state constraints via an equivalence relation. As the scope of variables $\in IF$ is the complete functional description, it is explicitly modelled that the purchase contract is for the product provided as input.

Example for static and dynamic symbols: bank account withdrawal
Ω: arithmetics-ontology
$\Sigma_S : \{a\}$
$\Sigma_D : \{balance\}$
$IF : \{a, x\}$
$\phi^{pre} : account(a) \wedge float(x) \wedge balance(a) \geq x.$
$\phi^{eff} : account(a) \wedge balance(a) = balance_{pre}(a) - x.$

The second example shows a functional description of a Web service for withdrawal from a bank account, addressing the usage and purpose of static and dynamic symbols. Again, there is an ontology that specifies the signature Σ and domain knowledge Ω. As input variables, we define an account a and the amount x that is to be withdrawn from it. In addition, a is defined as a static symbol to explicitly express that the account is not changed by executing the Web service. $balance$ is a unary predicate symbol defined in Ω that is declared as a dynamic symbol here, because its value is changed by the Web service execution. The precondition requires an account and the withdrawal amount such that the account balance is higher or equal; all occurring variables are elements of IF, so that no explicit quantification is needed. The effect specifies that the balance of the account after execution will be the balance before execution minus the withdrawal amount. Although one could also specify this setting without static and dynamic symbols by using different variables, their usage allows to explicitly state that the balance of the particular account is changed.

Concluding, the ASS model allows to explicitly and precisely specify functional descriptions. Its language independence allows the application within different frameworks for semantically describing Web services. Moreover, the explicit definition of the used modelling constructs as well as the meaning of functional descriptions allows to formally specify specific relationships and operations relevant for inference-based techniques for Web service detection. [18] presents this for desirable formal notions like realizability and functional refinement as adoptions from standard logical notions. Besides, the ASS model is extendible towards further aspects that become relevant for functional descriptions; [17] discusses the notion of so-called execution invariants (aspects that are guaranteed to hold during a Web service execution) and handling of the Frame problem.

3.2 Constructing Delta Relations for Functional Descriptions

We now turn towards the definition and construction of Δ-relations for functional descriptions within the ASS model. The application purpose of Δ-relations remains the same as in the previous discussions: to explicitly represent those aspects of the semantic difference that hamper successful resolution of a given goal by a Web service.

Intuitively, there are two possibilities for constructing a Δ-relation between the functional description $\mathcal{D}_G$ of a goal and $\mathcal{D}_W$ of a Web service. The first one is to define a set of Δ-relations between the corresponding description elements of functional descriptions, i.e. distinct Δs between the preconditions ϕ^{pre} and the effects ϕ^{eff} of $\mathcal{D}_G$ and $\mathcal{D}_W$. Both expressions are conventional logical formulae so that we could straight forward apply the definitions and techniques for Δ-relations presented in Section 2. However, for properly denoting the semantic difference of between the effect formulae we would need to take their dependence of the precondition into account. Consider the above purchase example. Under consideration of the universal quantification of all variables $\in IF$, the effect formula states: $\forall x, p.\ out(x) \leftrightarrow purchaseContract(x) \wedge for(x, p)$. Here, we miss the type restriction $product(p)$ that is defined in the precondition; this is also relevant for the effect constraint, which is supported in the ASS model by scoping the IF-variables over the complete functional description. For the account withdrawal example, we need to have knowledge about the value of $balance_{pre}(a)$ for evaluating the effect formula. Hence, for realizing this opportunity for construction Δ-relations, we would have to re-write the effect formulae of functional descriptions such that the relevant constraints of the preconditions are incorporated.

The second possibility is to represent functional descriptions as conventional logical expressions in a static knowledge specification language $\mathcal{L}$, and then construct Δ-relations over these formulae. In particular, we consider representing a functional description $\mathcal{D}$ as a logical formula $\phi^{\mathcal{D}}$ of the form $\forall i_1, \ldots, i_n.\ \phi^{pre} \Rightarrow \phi^{\mathit{eff}}$, whereby $i_1, \ldots, i_n$ corresponds to the IF-variables, ϕ^{pre} to the precondition, and ϕ^{eff} to the effect in Definition 6 above. In accordance to the motivation of the Situation Calculus [23], such a representation of functional descriptions appears to be desirable in order to allow definition of inference rules within the framework of a static knowledge representation language $\mathcal{L}$ – in particular for semantic matchmaking [21, 7, 18] as well as for denoting and handling semantic differences with Δ-relations. As for the first possibility, we have to take the dependence between the preconditions and effects in to account, with special attention to the dynamic aspects that are denoted by dynamic symbols Σ_D and their pre-variants Σ_D^{pre} within the ASS model. It is to remark that representing a functional description $\mathcal{D}$ as conventional logical formulae $\phi^{\mathcal{D}}$ is only possible if $\mathcal{D}$ only encompasses constraints on the pre- and post-state but not on any intermediate state of the sequence of state transitions described by $\mathcal{D}$.

The following discusses the representation functional descriptions as conventional logical formulae. Thereafter, we explain the applicability of the definitions and techniques for Δ-relations elaborated in Section 2 for functional descriptions that are represented as logical formulae.

3.2.1 Presenting Functional Descriptions as Conventional Logical Expressions

The problem of representing a functional description $\mathcal{D}$ that is defined in accordance to Definition 6 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 (see Section 2.1). We commence with discussing the meaning of functional descriptions, and derive a formal substantiation for representing functional descriptions as conventional FOL formulae.

In the ASS model, a functional description $\mathcal{D}$ restricts the possible sequences of state transitions $\tau = (s_1, \ldots, s_n)$ in $\mathcal{A}$ with respect to their pre-state s_1 and their post-state s_n. Its meaning is that if for some concrete values for the input variables $i_1, \ldots, i_n \in IF$ the information space $\omega(s_1)$ satisfies the precondition ϕ^{pre}, then the execution of a Web service W that provides $\mathcal{D}$ denotes a sequence of state transitions $\tau = (s_1, \ldots, s_n)$ such that $\omega(s_n) \models \phi^{\mathit{eff}}$, i.e. the information space of the termination state satisfies the effect. The post-state s_n depends on the pre-state s_1, which is defined via the IF-variables and the signature extensions for static and dynamic symbols. A Web service W is called a *capability model* of a functional description $\mathcal{D}$, , denoted by $W \models_{\mathcal{A}} \mathcal{D}$, if all possible executions of W satisfy $\mathcal{D}$.

With respect to their application purpose, functional descriptions merely consist of constraints on the pre- and post-state of sequences of state transitions but elide the 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 $[\phi^{pre}]_{\Sigma_D^{pre} \to \Sigma_D} \Rightarrow \phi^{\mathit{eff}}$ with the following correspondence to Definition 6: $IF = (i_1, \ldots, i_n)$ 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 [18], $[\phi]_{\Sigma_D^{pre} \to \Sigma_D}$ 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 $[\phi^{pre}]_{\Sigma_D^{pre} \to \Sigma_D}$.

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 $i_1, \ldots, i_n$. 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 5 illustrates this correlation that we formally substantiate in the following.

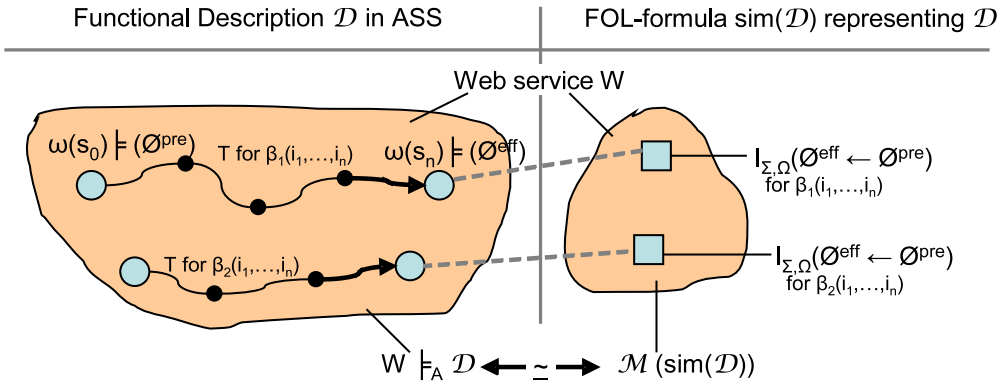

Figure 5: Correlation of a Functional Description $\mathcal{D}$ and $sim(\mathcal{D})$

Definition 7 (Web Service Execution Simulation)**.** *Let a Web service W be a pair $W = (IF, \iota)$ with a set of input variables $IF = (i_1, \ldots, i_n)$ 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 $\tau_W = (s_0, \ldots, s_n)$ 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 $\beta : (i_1, \ldots, i_n) \rightarrow \mathcal{U}_\mathcal{A}$ 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

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

(i) for all predicates $\alpha \in \Sigma$ with the arity m holds
$\forall x_1, \ldots, x_m \in \mathcal{U}_\mathcal{A}.\ (x_1, \ldots, x_m) \in I_{\tau_W}(\beta)(\alpha)$
$\Leftrightarrow (x_1, \ldots, x_m) \in I_W(\beta)(\alpha).$

(ii) for all functions $f \in \Sigma$ with the arity m holds
$\forall x_1, \ldots, x_m \in \mathcal{U}_\mathcal{A}.\ I_{\tau_W}(\beta)(f)(x_1, \ldots, x_m) = x_0$
$\Leftrightarrow I_W(\beta)(f)(x_1, \ldots, x_m) = x_0.$

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 6). 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 $\Sigma_\mathcal{A}$ such that all predicate and function symbols have the same meaning as in $\omega(s_n)$.

Definition 8 (Description Simulation)**.** *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 $i_1, \ldots, i_n$ 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 $\tau_W = (s_0, \ldots, s_n)$ 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

$\mathcal{D} \simeq \phi$ *iff.* for all $s_0 \in \mathcal{A}$ and for all $\beta \in In_\mathcal{A}(IF)$ holds that

(i) for each execution $\tau_W(\beta)$ of each Web service W with $W \models_\mathcal{A} \mathcal{D}$ holds that
for all Σ-interpretations $\mathcal{I}$ such that $\tau_W(\beta) \simeq \mathcal{I}$ holds that $\mathcal{I} \models \phi$.

(ii) for all Σ-interpretations $\mathcal{I}$ such that $\mathcal{I} \models \phi$ holds that
for all Web services W such that for each execution $\tau_W(\beta)$ of W
holds $\tau_W(\beta) \simeq \mathcal{I}$ holds that $W \models_\mathcal{A} \mathcal{D}$.

This definition states that a functional description $\mathcal{D}$ that is specified in accordance to Definition 6 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 7, this provides the correctness criterion of a first-order structure that represents a functional description $\mathcal{D}$ by maintaining the formal semantics.

Definition 9 (FOL Structure for representing a Functional Description)**.** *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 $\Sigma_{\mathcal{A}}$ as a pair $sim(\mathcal{D}) = (IF, \phi^{\mathcal{D}})$ with $IF = (i_1, \ldots, i_n)$ 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 $i_1, \ldots, i_n$ occur as free variables,*

(ii) *ϕ^{eff} is the formula defining the effect of $\mathcal{D}$ wherein $i_1, \ldots, i_n$ 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 6, 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 $\beta : (i_1, \ldots, i_n) \rightarrow \mathcal{U}_{\mathcal{A}}$ 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 form above. The input variables are $IF = \{a, x\}$, and the precondition specifies $\phi^{pre} = account(a) \wedge float(x) \wedge balance(a) \geq x$ (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 3 (Semantic Simulation of a Functional Description as a Logical Formula)**.** *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 $i_1, \ldots, i_n$ 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, $\mathcal{D} \simeq sim(\mathcal{D})$.

Proof. We need to show that clauses (i) and (ii) of Definition 8 hold for $\mathcal{D}$ and $sim(\mathcal{D})$. In essence, these clauses require an equivalence relation between a Web service $W \models_{\mathcal{A}} \mathcal{D}$ and the models of $sim(\mathcal{D})$: for all executions $\tau(\beta) = (s_0, \ldots, s_n)$ of W the corresponding interpretation $\mathcal{I}_W(\beta)$ that simulates the termination state s_n must be a model of $sim(\mathcal{D})$, and vice versa. Let us consider a Web service W such that $W \models_{\mathcal{A}} \mathcal{D}$, and three different input bindings $\beta : (i_1, \ldots, i_n) \rightarrow \mathcal{U}_{\mathcal{A}}$ 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 9 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 $\mathcal{I}_W(\beta_1) \models sim(\mathcal{D})$. For $\tau_W(\beta_1) = (s_0, \ldots, s_n)$ 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 6). It trivially holds that $\tau_{W_1}(\beta_1) \simeq \mathcal{I}_W(\beta_1)$, as $\mathcal{D}$ and $sim(\mathcal{D})$ are both defined over $\Sigma_{\mathcal{A}}$ and use the same precondition and effect formula along with the substitution for dynamic symbols. Hence, $\mathcal{D} \simeq sim(\mathcal{D})$ 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 $\mathcal{I}_W(\beta_2) \models sim(\mathcal{D})$ 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 $\tau_W(\beta_2) = (s_0, \ldots, s_n)$ can be any state – either such that $\mathcal{I}^1_{\tau_W}(\beta_2) \models \phi^{\mathit{eff}}$ or such that $\mathcal{I}^2_{\tau_W}(\beta_2) \not\models \phi^{\mathit{eff}}$. This correlates with the possibilities for $\mathcal{I}_W(\beta_2) \models sim(\mathcal{D})$, 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, $\mathcal{D} \simeq sim(\mathcal{D})$ is given for this case as well.

In case (3), it unambiguously holds that for all $\mathcal{I}_W(\beta_3) = (\mathcal{U}_\mathcal{A}, I_W(\beta_2))$ holds $\mathcal{I}_W(\beta_3) \not\models sim(\mathcal{D})$. 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, [18] 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 $\mathcal{D} \simeq sim(\mathcal{D})$ is given for this case as well.

This completes the proof. □

Proposition 5 (Stronger Representation of Functional Descriptions)**.** *Another representation of $\mathcal{D}$ is a first-order logic structure $sim(\mathcal{D})_2 = (IF, \phi^{\mathcal{D}}_{stronger})$ with $IF = (i_1, \ldots, i_n)$ being the set of input variables defined in $\mathcal{D}$, and $\phi^{\mathcal{D}}_{stronger}$ being a first-order logic formula of the form $[\phi^{pre}]_{\Sigma^{pre}_D \rightarrow \Sigma_D} \wedge \phi^{\mathit{eff}}$. It holds that $\phi^{\mathcal{D}}_{stronger} \models \phi^{\mathcal{D}}$.*

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 $\beta \models \phi^{\mathit{eff}}$. 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 6. 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. [17]); accordingly, $sim(\mathcal{D})$ is called the *modelling perspective*.

3.2.2 Delta-Relations for Logical Expressions that represent Functional Descriptions

The preceding elaborations have shown that we can simulate a functional description $\mathcal{D}$ by a semantically corresponding FOL structure $sim(\mathcal{D}) = (IF, \phi^{\mathcal{D}})$. While neglecting dynamic aspects about states and state transitions performed by execution of Web services, this representation allows to describe the objects retrievable from using Web services. This information are sufficient for the purpose of Δ-relations as a means for denoting the semantic difference of the functionality requested by a goal and the one provided by a Web service that is to be used for solving the goal. $\phi^{\mathcal{D}}$ is a conventional FOL formula that semantically simulates the relationship between preconditions and effects in $\mathcal{D}$. Hence, we can straight forward apply the definitions and techniques for constructing Δ-relations that introduced above in Section 2.

Let us consider a goal G described by a functional description $\mathcal{D}_G$ in the ASS model (see Definition 6), and a Web service W described by $\mathcal{D}_W$. As discussed in the previous sections, the purpose of Δ-relations is to explicate the aspects of the semantic difference between $\mathcal{D}_G$ and $\mathcal{D}_W$ that cause certain conditions on how the Web service can be used for solving the goal. We have argued that a suitable structure of a Δ-relation

is a pair $\Delta = (\Delta_1, \Delta_2)$ such that $\mathcal{D}_G, \Delta_1 \equiv \mathcal{D}_W, \Delta_2$ (see Definition 5). For conventional FOL formulae ϕ_1, ϕ_2, we have shown that $\Delta = (\neg\delta_1, \neg\delta_2)$ with $\delta_1 = \phi_1 \wedge \neg\phi_2$ and $\delta_2 = \phi_2 \wedge \neg\phi_1$ is a sophisticated definition for a Δ-relation (see Theorem 1).

Within the ASS model, the paired structure of a Δ-relation shall mean that δ_1 explicitly denotes the preconditions and effects of those possible sequences of state transitions $\tau_{G,\ not\ W}$ in an Abstract State Space $\mathcal{A}$ that allow solving G but are not possible executions of W. Correspondingly, δ_2 describes those sequences of state transitions $\tau_{W,\ not\ G}$ in $\mathcal{A}$ that are possible executions of W but do not result in a state that solves G. Applying the definition $\Delta = (\neg\delta_1, \neg\delta_2)$ to FOL structures $sim(\mathcal{D}_G)$ and $sim(\mathcal{D}_W)$ that semantically simulate $\mathcal{D}_G$ and $\mathcal{D}_W$ (see Theorem 3), it holds that for all Σ-interpretations $\mathcal{I}$ with $\mathcal{I} \models \delta_1$ holds that $\mathcal{I} \models sim(\mathcal{D}_G)$ and $\mathcal{I} \not\models sim(\mathcal{D}_W)$, and for all Σ-interpretations $\mathcal{I}$ with $\mathcal{I} \models \delta_2$ holds that $\mathcal{I} \models sim(\mathcal{D}_W)$ and $\mathcal{I} \not\models sim(\mathcal{D}_G)$. Thereby, the formulae that constitute δ_1 as well as δ_2 are constructed over the FOL formula $\phi^{\mathcal{D}_G} \in sim(\mathcal{D}_G)$ and $\phi^{\mathcal{D}_W} \in sim(\mathcal{D}_W)$ (see Definition 9). These formulae describe the logical relationship between the precondition and effect whose free IF-variables are bound by a specific input binding $\beta : (i_1, \ldots, i_n) \rightarrow \mathcal{U}_{\mathcal{A}}$. Figure 6 illustrates this correspondence, and Definition 10 defines the construction of Δ-relations for functional descriptions by adapting Theorem 1).

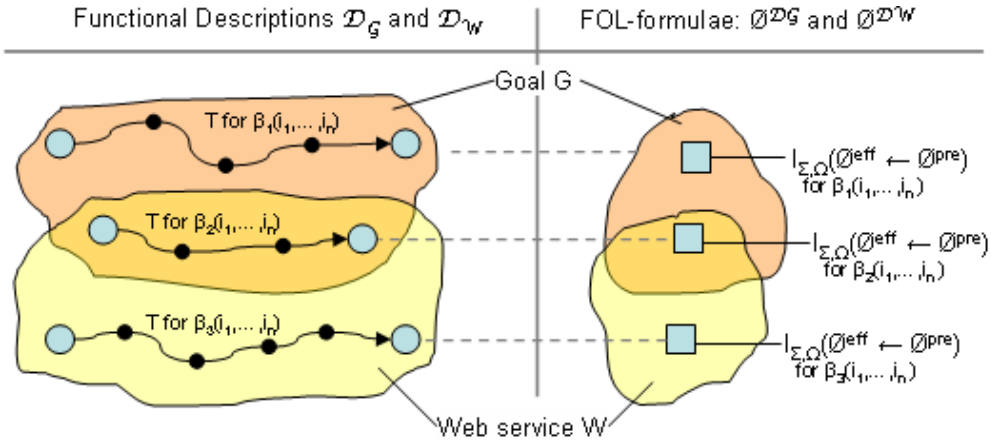

Figure 6: Correspondence of Differences for Functional Descriptions described by $\mathcal{D}$ and $\phi^{\mathcal{D}}$

Definition 10 (Construction of Delta Relations for Functional Descriptions)**.** *Let $\mathcal{D}_G$ describe the a functionality requested in a goal and $\mathcal{D}_W$ describe the functionality provided by a Web service. Let $\mathcal{D}_G$ and $\mathcal{D}_W$ be defined in an Abstract State Space $\mathcal{A}$ over the signature $\Sigma_{\mathcal{A}}$ and with respect to formalized domain knowledge Ω such that the in variables $IF = (i_1, \ldots, i_n)$ occur as free variables in the precondition ϕ^{pre} and an effect ϕ^{eff}. Let $sim(\mathcal{D}) = (IF, \phi^{\mathcal{D}})$ be a first-order logic structure with the formula $\phi^{\mathcal{D}}$ being of the form $[\phi^{pre}]_{\Sigma_D^{pre} \rightarrow \Sigma_D} \Rightarrow \phi^{eff}$. Let $sim(\mathcal{D}_G) = (IF_G, \phi^{\mathcal{D}_G})$ semantically simulate $\mathcal{D}_G$ and $sim(\mathcal{D}_W) = (IF_W, \phi^{\mathcal{D}_W})$ semantically simulate $\mathcal{D}_W$ such that $sim(\mathcal{D}_G) \simeq \mathcal{D}_G$ and $sim(\mathcal{D}_W) \simeq \mathcal{D}_W$.*

Then, the pair of formulae $\Delta(\phi^{\mathcal{D}_G}, \phi^{\mathcal{D}_W}) = (\neg\delta_1(\phi^{\mathcal{D}_G}, \phi^{\mathcal{D}_W}), \neg\delta_2(\phi^{\mathcal{D}_G}, \phi^{\mathcal{D}_W}))$ with

$$\delta_1(\phi^{\mathcal{D}_G}, \phi^{\mathcal{D}_W}) = (\phi^{\mathcal{D}_G} \wedge \neg\phi^{\mathcal{D}_W})$$
$$\delta_2(\phi^{\mathcal{D}_G}, \phi^{\mathcal{D}_W}) = (\neg\phi^{\mathcal{D}_G} \wedge \phi^{\mathcal{D}_W})$$

is a Δ-relation of $\mathcal{D}_G$ and $\mathcal{D}_W$ over Ω.

In essence, this definition states that for some FOL structures that semantically simulate functional descriptions we can straight forward apply the definitions and techniques for constructing and handling Δ-relations as elaborated in Section 2. In particular, the necessary requirement on *completeness* – i.e. establishing logical equality of $\mathcal{D}_G$ and $\mathcal{D}_W$ via $sim(\mathcal{D}_G) \wedge \Delta_1 \equiv sim(\mathcal{D}_W) \wedge \Delta_2$ – and the sufficient condition of *minimality* – i.e. a Δ' that satisfies the necessary condition and $\Delta \sqsubset \Delta'$ does not exist – hold for this construction of Δ-relations as well (see Theorem 1). Also, the relationship to semantic matchmaking (see Section 2.3) with the strict entailment ordering of Δ-relations on matchmaking levels (see Theorem 2) hold for Δ-relations on functional descriptions as well.

Moreover, the technique of formula simplification by tautologies as well as the inference rules for applying Δ-relations that we have presented in Section 2.4 are straight forward applicable for functionalities formalized in the ASS model. For demonstration purpose, Table 5 shows the modelling of the goal for finding best restaurants in a city as a functional description as a functional description $\mathcal{D}_G$ and its semantically corresponding representation as an FOL formula $sim(\mathcal{D}_G)$ following the above definitions.

Table 5: Semantically Corresponding Representations of a Goal

Functional Description $\mathcal{D}_G$		**Semantic Correspondence** $sim(\mathcal{D}_G)$	
Σ:	first-order logic	Σ:	first-order logic
Ω:	better restaurant ontology	Ω:	better restaurant ontology
IF:	{x}	IF:	{x}
ϕ^{pre}:	$memberOf(x, city)$	$\phi^{\mathcal{D}_G}$:	$\forall y.\ in(x) \wedge memberOf(x, city)$
ϕ^{eff}:	$\forall y.\ out(y) \Leftrightarrow ($		$\Rightarrow (out(y) \Leftrightarrow ($
	$memberOf(y, restaurant)$		$memberOf(y, restaurant)$
	$\wedge\ hasAttValue(y, in, x)$		$\wedge\ hasAttValue(y, in, x)$
	$\wedge \neg\exists z.(memberOf(z, restaurant)$		$\wedge \neg\exists z.(memberOf(z, restaurant)$
	$\wedge\ hasAttValue(z, in, x)$		$\wedge\ hasAttValue(z, in, x)$
	$\wedge\ better(z, y))).$		$\wedge\ better(z, y)))).$

(Not) surprisingly, $sim(\mathcal{D}_G)$ is nearly the same formula that we have defined for describing the goal as a conventional FOL formula in Section 2.4.1. The mere difference is that the IF-variable x occurs as a free variable in $\phi^{\mathcal{D}_G}$, while in the example we have explicitly defined a universal quantification for x. The meaning is that $sim(\mathcal{D}_G)$ can only be interpreted when a concrete input binding is provided. The same correspondence holds for the functional description of the Web service for providing the best French restaurant in a city. In consequence, the techniques for and applications of Δ-relations presented throughput the example extensively discussed in Section 2.4 are straight forward applicable for goals and Web services described by functional descriptions in terms of preconditions and effects with the structure and meaning as defined in Definition 6.

In particular for the example from Section 2.4, consider a β_1 that defines a city wherein the best restaurant is of type French, then the provide-best-French-restaurant Web service is usable for solving G. The reason is that for β_1 the assumption obtained from $\neg\delta_2$ is satisfied (see Section 2.4.3). For a β_2 with a city wherein the best restaurant is not French, the usage assumption is violated so that we now that the Web service can not be used. Hence, the representation of functional descriptions by FOL structures that semantically simulate the formal semantics allows to construct Δ-relations and to apply them for explicating the conditions under which a Web service is usable for solving a goal.

3.3 Integration with WSMO Mediators

The final aspect of defining Δ-relations is their integration into the WSMO mediation framework. Presented in [32] with more detailed specifications in [24], the aim is to provide an integrated technology for handling heterogeneities that may hamper successful Web service usage. WSMO therefore defines different types of modularized mediators that connect potentially heterogeneous resources and utilize specific techniques for handling and resolving distinct types of heterogeneities such as data level or process level mismatches.

We can understand the semantic differences between the functionality requested by a goal and the one provided by a Web service as a separate type of heterogeneity. As discussed in the preceding elaborations, these differences cause conditions or impacts on the usability of a Web service for solving a given goal. Δ-relations provide a means for explicating the semantic difference of functional descriptions as the basis for advanced techniques for Web service detection such as goal refinement or explication of usage conditions, hence they serve as a mediation technique for this heterogeneity type. When specifying a mediator that connects a goal and a Web service and explicates the matchmaking degree along with a Δ-relation that describes the semantic difference between them, this mediator carries all relevant information on usability of the Web service for solving the goal. We shall refer to this as the *functional level of mediation* wherefore we provide the definitions for integration into the WSMO mediation framework in the following. The main merit of this is to obtain a directed relationship between goals and Web services described in a declarative manner; moreover, the modularized structure of WSMO mediators allows to incorporate mediation facilities for other heterogeneity types that possibly occur between goals and Web services.

Definition 11 (Mediator)**.** *A Mediator $\mathcal{M}$ connects heterogenous resources and mediation techniques for heterogeneity handling. It is defined as a 5-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{T}, \mathcal{MD}, \mathcal{MS}, \mathcal{UR})$ such that:*

- *(i)* *$\mathcal{S}$ denotes the source element of $\mathcal{M}$,*
- *(ii)* *$\mathcal{T}$ denotes the target element of $\mathcal{M}$,*
- *(iii)* *$\mathcal{MD}$ is a* mediation definition *for resolving heterogeneities between $\mathcal{S}$ and $\mathcal{T}$,*
- *(iv)* *$\mathcal{MS}$ is a* mediation service *that is capable of processing $\mathcal{MD}$, and*
- *(v)* *$\mathcal{UR}$ denotes other resource incorporated in $\mathcal{M}$.*

This general definition states that a mediator is a directed connection of a source element $\mathcal{S}$ to a target element $\mathcal{T}$ along with techniques for resolving potentially occurring heterogeneities between $\mathcal{S}$ and $\mathcal{T}$. A mediation definition $\mathcal{MD}$ denotes the technique for explicating and resolving distinct types of heterogeneities; this refers to ontology mappings in the context of data level mediation [29], or to process mediation patterns for process level mediation [6]. A mediation service $\mathcal{MS}$ in this definition denotes a computational resource that is capable of executing the mediation definition. Such facilities can most suitably be provided as Web services themselves (with the capability of processing mediation definitions); an example is the WSMX data mediator that allows to mediate ontologies on basis of mapping rules [25]. Finally, $\mathcal{UR}$ denotes resources used for defining the mediator $\mathcal{M}$ such as domain ontologies, mapping languages, or other mediators $\mathcal{M}_1, \ldots, \mathcal{M}_n$ that are re-used within $\mathcal{M}$.

As a refinement of Definition 11, we specify mediators that carry Δ-relations as follows:

1. a mediator $\mathcal{M}$ has one source element $\mathcal{S}$ that can be a goal or a Web service, and one target element $\mathcal{T}$ that can be a goal or a Web service
2. a Δ-relation is a mediation definition $\mathcal{MD}$ that describes the semantic difference between $\mathcal{S}$ and $\mathcal{T}$ as a pair of formula $\Delta(\phi^{\mathcal{D}_\mathcal{S}}, \phi^{\mathcal{D}_\mathcal{T}}) = (\neg\delta_1, \neg\delta_2)$ with $\delta_1 = (\phi^{\mathcal{D}_\mathcal{S}} \wedge \neg\phi^{\mathcal{D}_\mathcal{T}})$ and $\delta_2 = (\neg\phi^{\mathcal{D}_\mathcal{S}} \wedge \phi^{\mathcal{D}_\mathcal{T}})$ such that $\phi^{\mathcal{D}_\mathcal{S}} \simeq \mathcal{D}_\mathcal{S}$ and $\phi^{\mathcal{D}_\mathcal{T}} \simeq \mathcal{D}_\mathcal{T}$

3. in addition to the semantic difference, the mediator explicitly denotes the matchmaking degree between $\mathcal{S}$ and $\mathcal{T}$.

Aspect (1) states that a mediator with a Δ-relation connects WSMO elements that have functional description, i.e. goals with goals, goals with Web services, or Web service with Web services. It is important that such a mediator only has one source and one target element, as the semantic difference is unique between each pair of functional descriptions. (2) states that the Δ-relation between the source and target is specified in accordance to Theorem 3, i.e. on basis of formulae that semantically correspond to the functional descriptions of the source and target component. Here, δ_1 denotes the semantic difference aspects from the perspective of the source and δ_2 from perspective of the target. Finally, (3) states that in addition to the Δ-relation, the mediator explicates the matchmaking degree between the source and the target component. As discussed in Section 2.3, this information is integrally correlated to Δ-relations: from a given Δ-relation we can deduce the matchmaking degree, and from a given matchmaking degree we gain knowledge on the structure of the Δ-relation (see Theorem 2).

The application purpose of such a mediator is as follows. Imagine we have given a goal G and a Web service W. Let the matchmaking degree be `plugin(`G, W`)` so that only a subset of the possible executions of W can solve the goal. Following Definition 6, the results of executing W depend on the provided inputs. Hence, in order to use W for solving G, the client needs to ensure that the inputs provided to W are such that the results from executing W will satisfy the goal. Now. let $\mathcal{M}$ be a mediator with G as the source and W as the target so that its Δ-relation is of the form $\Delta(\phi^{\mathcal{D}_G}, \phi^{\mathcal{D}_W}) = (\neg\delta_1, true)$ (see Section 2.3). Here, δ_1 represents those sequences of state transitions in the world that result in solving of the goal but are not possible executions of W. This denotes an additional constraint for using W for solving G: for all bindings β for the input variables $i_1, \ldots, i_n$ satisfy $\phi^{pre} \wedge \neg\delta_1$ the executions of W with the input β will result in a state that satisfies G. Therewith, the Δ-relation defines the usage conditions for the particular Web service W for solving the goal G that are explicated in the mediator $\mathcal{M}$ in a declarative manner.

For integration into WSMO, Listing 1 shows the description model of mediators with Δ-relations, in language used in WSMO for MOF meta-model layer specifications [28]. Below, Figure 7 shows the integrated topology of the distinct WSMO mediator types along with their correlation and the used mediation techniques. It is to remark that with respect to the scope of the preceding elaborations, the presented definition of Δ-relations merely covers languages with model-theoretic semantics; for WSML [5] as the specification language for WSMO, these are the variants WSML-Core and WSML-DL, and possibly WSML-Full that is under construction at the time of writing. Moreover, obtaining a human understandable representation of Δ-relations most likely requires the technique for formula simplification on the basis of tautologies. As illustrated and discussed in Section 2.4.2, this technique can not be automated and hence requires manual inspection.

```
Class mediator
  importsOntology type ontology
  usesMediator type ooMediator
  hasSource type {goal, webService} multiplicity = single−valued
  hasTarget type {goal, webService} multiplicity = single−valued
  hasDeltaRelation type deltarelation
Class deltarelation
  hasMatchingDegree type {equal, subsume, plugin, intersection}
  hasDelta1Definition type axiom
  hasDelta2Definition type axiom
```

Listing 1: Meta-Model for Mediators with Delta-Relations

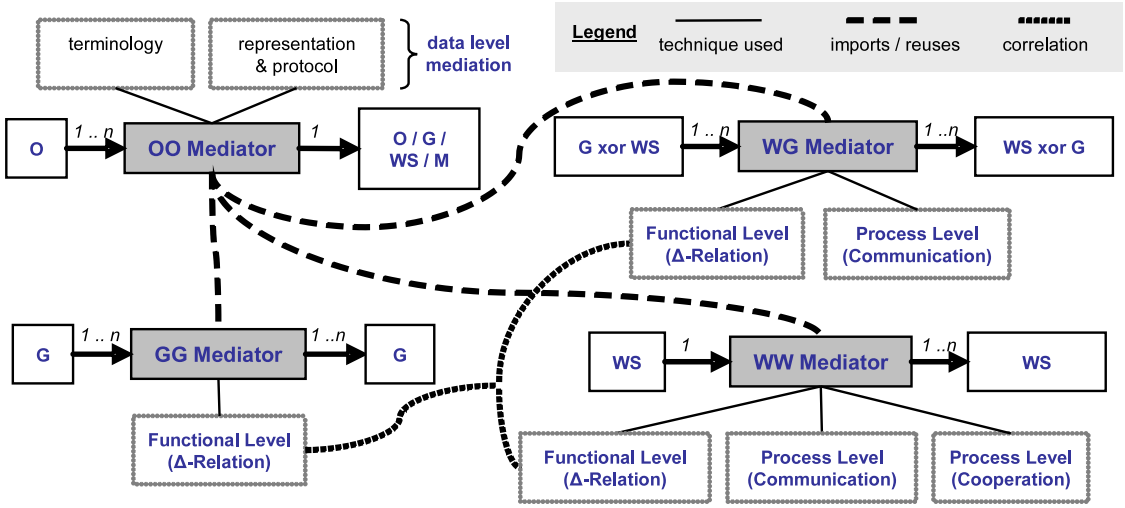

Figure 7: WSMO Mediator Topology

OO Mediators provide a general data level mediation component for ontology-based applications. The source elements are ontologies or other OO Mediators, while the target can be any WSMO top level element. The mediation techniques are data level mediation techniques such as ontology mapping and integration, and lifting and lowering from data to and from the ontological level.

GG Mediators connect WSMO goals, i.e. both the source and target are goals. The mediation techniques used are: (1) data mediation by usage of OO Mediators, and (2) functional level mediation on basis of Δ-relations.

WG Mediators connect Web services and goals in case a Web service is not usable for solving a goal a priori, or only under certain conditions. The used mediation techniques are: (1) data level mediation by usage of OO Mediators, (2) Δ-relations for handling functional heterogeneity, and (3) process level mediation for resolving potential mismatches on the communication level between the source and target component.

WW Mediators connect Web services that interact but are not compatible a priori. The used mediation techniques are: (1) data level mediation by usage of OO Mediators, (2) Δ-relations for handling functional heterogeneities, and (3) process level mediation for resolving mismatches between the source and target service with respect to communication and coordination of interaction.

4 Related Work

We are not aware of any other approach for explicating the usage conditions of Web services for goals. However, Δ-relations as specified in this paper present an extension to the state of the art in semantic matchmaking for Web service detection that has been inspired by earlier works on explicating hidden assumptions in formal resource descriptions. We position our approach in the following.

While the need for a technique for explicating usage conditions of Web services for solving a goal has emerged from works on Web service detection (e.g. [34]), the definition of Δ-relations presented in this paper has been inspired by the work on assumption explication and refiners within UPML [10], a framework for formally describing the reasoning behavior of knowledge-based systems that work with Problem Solving Methods (PSMs). Therein, assumptions are used as integral description for explicating the gap between the functionality provided by a PSM and the one required for achieving a specific task [4]; this correlates to the purpose of Δ-relations. Refiners are logical elements that connect tasks and PSMs and restrict their formal descriptions such that its source and target are semantically equivalent under inclusion of the refiner. Such refiners have the same purpose as mediators with Δ-relations as specified in Section 3.3.

Already mentioned in the introduction, [9] presents so-called *inverse verification* as a technique for determining assumptions as well as the logical expressions in UPML-refiners. In a nutshell, this is performed by manual analysis of failed proofs on the usability of PSMs for a given task. An interactive theorem prover (here: KIV) takes the formal descriptions of the task and the PSM are used as input, and returns "open goals" as the gaps where knowledge is missing for the next proof step. Then, logical expressions are identified that allow to handle the open goals; this step requires manual analysis. The determined statements denote the additional constraints for using the PSM to solve the task. As an example, [9] takes a task requires a global search for a set of elements and the available PSM that provides a local search facility. This is usable under the condition that the input element set has a strict total order, which is determined by inverse verification. The characteristics of this example setting is the same as the best-restaurant example discussed in Section 2.4: there is an intersection match between the task and the PSM, and the determined assumption explicates a usage constraint for the PSM. If this is satisfied by the concrete inputs, then the result of the PSM will satisfy the task.

Hence, inverse verification and Δ-relations are defined here are complementary techniques for detecting usage conditions for formally described provided functionalities that do not precisely match a requested functionality. Δ-relations appear to be an improvement because the need for human intervention is minimized. The main difference is that inverse verification relies on manual failure analysis while Δ-relations by definition explicate the semantic difference of functional descriptions. In consequence, inverse verification leaves the hard part of finding a logical statement that allows solving the open goals from the theorem prover to human inspection; in contrast, a Δ-relation immediately provides a correct and minimal logical formula so that merely its simplification by tautologies requires manual intervention. Nevertheless, the requirements of completeness and minimality as well as certain logical relationships between Δ-relations, goals, and Web services have been adopted from the works around UPML.

Exiting approaches on semantic matchmaking for Web service detection are mainly concerned with the definition of the matchmaking degrees within respective logical frameworks and the implementation of general purpose matchmakers. The distinct works address certain related aspects – e.g. a precedence order of the matchmaking degrees [21], or the concept of client intentions and partial matches in [16]. However, non of these approaches addresses the need for explicating the usage conditions of Web services that are explicated by Δ-relations: without these information, the client needs to guess which inputs may be suitable for solving the goal with the Web service.

5 Conclusions and Future Work

This paper has presented Δ-relations as a technique for explicating conditions under which a Web service is usable for solving a goal if the provided and requested functionality do not match precisely.

Research on semantic matchmaking as the core technique for semantically enabled Web service detection has identified four degrees of matching under which a Web service W is considered to be usable for solving a goal G. Apart from the exact match (the requested and provided functionality are semantically equivalent), each of this matchmaking degrees implies conditions on how to use the Web service. For the plugin and the intersection match, only certain executions of W can satisfy G; hence, the client needs to ensure that the concrete inputs provided for invocation of W will result in such executions that satisfy G. For the subsume match, the usage of W will have certain impacts on how G is resolved, which may be of interest of the client. A Δ-relation explicitly describes the semantic difference between functional descriptions, providing the basis for explicating such conditions and impacts of using a particular Web service for solving a goal.

In this paper we have defined Δ-relations for classical first-order logic as the specification language for static knowledge. Naturally, the definitions and techniques can straight forward be applied to formal languages with model-theoretic semantics such as Description Logic, Horn Logic, and Description Logic Programs. For two formulae ϕ_G, ϕ_W in such a language that represent a goal and a Web service, we have defined a Δ-relation as a pair of formulae such that $\Delta(\phi_G, \phi_W) = (\neg\delta_1, \neg\delta_2)$ with $\delta_1 = (\phi_G \wedge \neg\phi_W)$ and $\delta_2 = (\neg\phi_G \wedge \phi_W)$. We have shown that such a Δ-relation is *complete* as it allows establish logical equivalence by $\phi_G \wedge \neg\delta_1 \Leftrightarrow \phi_W \wedge \neg\delta_2$) and *minimal* such that there is no Δ' that is complete and $\Delta \models \Delta'$. We also explained the relationship of Δ-relations to semantic matchmaking, in particular that each matchmaking degree correlates to a specific structure of the Δ-relation between the functional descriptions of the goal and the Web service. We have demonstrated this within an exhaustive example, along with a technique for formulae simplification as well as inference rules for applying Δ-relations for goal refinement and assumption explication.

After discussing the properties and formal semantics of functional descriptions that are defined in terms of preconditions and effects on basis of the Abstract State Space model, we have shown that such a functional description $\mathcal{D}$ can be represented by conventional formulae $\phi^{\mathcal{D}}$ in a formal language with model-theoretic semantics so that $\phi^{\mathcal{D}}$ semantically corresponds to $\mathcal{D}$. In consequence, the definitions and techniques for Δ-relations on conventional logical formulae are straight forward applicable for explicating the semantic difference between formal functional descriptions. Finally, we have outlined the integration of Δ-relations into the WSMO mediation framework as a technique for handling not precisely matching requested and provided functionalities.

Therewith, we have specified a technique for explicating conditions and impacts of using a Web service for solving a goal that arise under not-exact matchmaking degrees. Inspired by previous work within the UPML framework, such a technique does not yet exist for the context of Web service detection. As future work, we plan to integrate Δ-relations into semantic matchmakers developed in the course of WSMO, and extending the definitions towards languages with minimal model semantics in order to also cover the LP branch of ontology languages that are under development for the Semantic Web.

Acknowledgments This work has greatly benefited from interesting discussions with Dieter Fensel. The authors like to thank Stijn Heymans for extensive review and feedback to the presented work. 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).

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] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[3] S. Battle, A. Bernstein, H. Boley, B. Grosof, 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] V. R. Benjamin, D. Fensel, and R Straatman. Assumptions of Problem-Solving Methods and their Role in Knowledge Engineering. In *Proceedings of the European Conference on Artifical Inteliigence (ECAI 1996), Budapest, Hungary*, 1996.

[5] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language WSML. Deliverable D16.1 final draft 05 Oct 2005, WSML Working Group, 2005. Available at: http://www.wsmo.org/TR/d16/d16.1/v0.2/.

[6] E. Cimpian and A. Mocan. WSMX Process Mediation Based on Choreographies. In *Proceedings of the 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management at the BPM 2005, Nancy, France*, 2005.

[7] I. Constantinescu, B. Faltings, and W. Binder. Large Scale, Type-Compatible Service Composition. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04), San Diego, California, USA*, 2004.

[8] de Bruijn. J., editor. *Ontology Languages around FOL and LP*. 2005. Working Draft 9-Dec-2005.

[9] D. Fensel and A. Schnegge. Inverse Verification of Problem-Solving Methods. *International Journal of Human-Computer Studies*, 49(4):339–361, 1998.

[10] D. Fensel et al. The Unified Problem Solving Method Development Language UPML. *Knowledge and Information Systems Journal (KAIS)*, 5(1), 2003.

[11] M. Fitting. *First Order Logic and Automated Theorem Proving*. Springer, 2 edition, 1996.

[12] B. N. Grosof, I. Horrocks, R. Volz, and Decker S. Description Logic Programs: Combining Logic Programs with Description Logic. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003*, pages 48–57, 2003.

[13] C. A. R. Hoare. An Axiomatic Basis for Computer Programming. *Communincations of the ACM*, 12(10):576–580, 1969.

[14] A. Horn. On Sentences Which Are True of Direct Unions of Algebras. *Journal of Symbolic Logic*, 16:14–21, 1951.

[15] U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel. Automatic Location of Services. In *Proceedings of the 2nd European Semantic Web Conference (ESWC 2005), Crete, Greece*, 2005.

[16] U. Keller, R. Lara, and A. Polleres (eds.). WSMO Web Service Discovery. Deliverable D5.1, WSML Working Group, 2004. Available at: http://www.wsmo.org/TR/d5/d5.1/.

[17] 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/.

[18] U. Keller, H. Lausen, and M. Stollberg. On the Semantics of Funtional Descriptions of Web Services. In *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006), Montenegro*, 2006.

[19] G. Kifer, M. andLausen and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *JACM*, 42(4):741–843, 1995.

[20] 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/.

[21] 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.

[22] D. Martin. OWL-S: Semantic Markup for Web Services. W3C Member Submission 22 November 2004, 2004. online: http://www.w3.org/Submission/OWL-S/.

[23] J. McCarthy. Situations, Actions and Causal Laws. Technical report, Stanford University, 1963.

[24] 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/.

[25] A. Mocan (ed.). WSMX Data Mediation. WSMX Working Draft D13.3, 2005. available at: http://www.wsmo.org/TR/d13/d13.3/v0.2/.

[26] 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.

[27] A. Riazanov and A. Voronkov. The Design and Implementation of VAMPIRE. *AI Communications*, 15(2):91–110, 2002. Special Issue on CASC.

[28] D. Roman, H. Lausen, and U. Keller. Web Service Modeling Ontology (WSMO). Working Draft D2, WSMO Working Group, 2005. final version v1.2, 13 April 2005, online at: http://www.wsmo.org/TR/d2/v1.2/.

[29] F. Scharffe and J. de Bruijn. A language to specify mappings between ontologies. In *Proc. of the Internet Based Systems IEEE Conference (SITIS05)*, 2005.

[30] R. M. Smullyan. *First Order Logic*. Springer, 1968.

[31] M. Stollberg, E. Cimpian, and D. Fensel. Mediating Capabilities with Delta-Relations. In *Proceedings of the First International Workshop on Mediation in Semantic Web Services, co-located with the Third International Conference on Service Oriented Computing (ICSOC 2005), Amsterdam, the Netherlands*, 2005.

[32] M. Stollberg, E. Cimpian, A. Mocan, and D. Fensel. A Semantic Web Mediation Architecture. In *Proceedings of the 1st Canadian Semantic Web Working Symposium (CSWWS 2006), Quebec, Canada*, 2006.

[33] M. Stollberg and M. Hepp. Goal Description Ontology. Deliverable D3.10, DIP, 2006.

[34] 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.

[35] M. Stollberg and F. Rhomberg. Survey on Goal-driven Architectures. DERI Technical Report, 2006. online: ... (to appear).

[36] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I. Computer Science Press, 1988.