WSML Deliverable

# D5.1 v0.1
# WSMO Web Service Discovery

WSML Working Draft – November 12, 2004

**Authors:**
Uwe Keller, Rubén Lara, Axel Polleres, Ioan Toma, Michael Kifer, and Dieter Fensel

**Editors:**
Uwe Keller, Rubén Lara, and Axel Polleres

**This version:**
http://www.wsmo.org/2004/d5/d5.1/v0.1/20041112
**Latest version:**
http://www.wsmo.org/2004/d5/d5.1/v0.1/
**Previous version:**
http://www.wsmo.org/2004/d5/d5.1/v0.1/20041008

# Abstract

The *Web Service Modeling Ontology* (WSMO) provides the conceptual framework for semantically describing web services and their specific properties. The *Web Service Modeling Language* (WSML) is a formal language for annotating web services with semantic information, which is based on the WSMO conceptual framework.

Service discovery, i.e., the automatic location of services that fulfill a user goal is a popular research topic. However, a proper conceptual setting for this task is largely missing. In this document we try to fill in this gap and discuss a number of conceptual issues related to service discovery. First, we propose a distinction between service discovery and Web service discovery. Then we identify three major steps in service discovery and note that only one of these steps has to do with Web service discovery proper. Second, we discuss different techniques for web service discovery using this as a means for a better understanding of the dialectic relationship between discovery and mediation. Third, we will discuss the logical foundations of web service discovery. In order to cover the complete range of scenarios that can appear in practical applications, several approaches to achieve the automation of web service discovery are presented and discussed. They require different levels of semantics in the description of web services and requests, and have different complexity and precision.

WSMO, WSML, and the discovery model presented in this paper constitute the necessary theoretical basis for automatic location of services in different scenarios, ranging from natural language descriptions of web services and discovery requests to precise logical definitions of the functionality of web services and the requester's objectives.

# Contents

# 1 Introduction

The web is a tremendous success story. Starting as an in-house solution for exchanging scientific information it has become, in slightly more than a decade, a world wide used media for information dissemination and access. In many respects, it has become the major means for publishing and accessing information. Its scalability and the comfort and speed in disseminating information has no precedent. However, it is solely a web for humans. Computers cannot "understand" the provided information and in return do not provide any support in processing this information. Two complementary trends are about to transform the web, from being for humans only, into a web that connects computers to provide support for human interactions at a much higher level than is available with current web technology.

- The semantic web is about adding machine-processable semantics to data. The computer can "understand" the information and therefore process it on behalf of the human user (cf. [Fen03]).
- Web services try to employ the web as a global infrastructure for distributed computation, for integrating various applications, and for the automatization of business processes (cf. [ACKM03]). The web will not only be the place where human readable information is published but the place where global computing is realized.

The semantic web promises to make information understandable to a computer and web services promise to provide smooth and painless integration of disparate applications. Web services offer a new level of automatization in eWork and eCommerce, where fully open and flexible cooperation can be achieved, on-the-fly, with low programming costs. However, the current implementations of web service technology are still far from reaching these goals, as integrating heterogeneous and dynamically changing applications is still a tremendous task.

Eventually, semantic web services promise the combination of semantic web with web service technology in order to overcome the limitations of current web services by adding explicit semantics to them. The exploitation of such semantics can enable a fully mechanized web for computer interaction, which would become a new infrastructure on which humans organize their cooperations and business relationships (cf. [FB02]). OWL-S [The04] and WSMO [RLK04] are the major proposals for providing semantic annotations on top of a web service infrastructure.

An important step for fully open and flexible eCommerce would be the mechanization of service discovery. As long as human intervention is required in service discovery the potential costs of establishing a new eCommerce link may outrange the potential savings and advantages. Open, flexible, on-the-fly creation of new supply chains is essentially based on full or nearly full automatization of this process. Therefore, it is not surprising that automatic web service discovery is a popular research topic and many papers are published on it (cf. [AGDR03], [LH03a], [PKPS02a], [BHRT03], [GCTB01], [VSSP04], [SWKL02], [ZK04]). Still, many of these papers discuss discovery in the setting of multi-agent systems or in the setting of description logic based reasoning and none of them really seems to take a look at the actual conceptual and pragmatic issues that are involved in service discovery by using web services.

Therefore, we provide an in-depth analysis of the major conceptual issues that are involved in service discovery via web services.

- First, we strictly distinguish between service and web service discovery,

identifying three major steps in service discovery where only one of them is about web service discovery.

- Second, we discuss different techniques for web service discovery using this as a means for achieving a better understanding of the dialectic relationship between discovery and mediation. In this context, we discuss the mediation support needed for different approaches to web service discovery.
- Third, we discuss in detail logic based discovery of web services. Stepwise we will enrich the scenario we are able to support.

In conclusion, we provide a conceptual model for service discovery and different approaches to one of the steps of such model, web service discovery, which can be realized by WSMO and related efforts.

This document is organized as follows. Section 2 identifies three major conceptual phases in service discovery. Section 3 analyzes the relationship between discovery and mediation based on different techniques to achieve web service discovery. Section 4 discusses in detail the issues around logic-based discovery of web services. Finally, Section 5 concludes the paper.

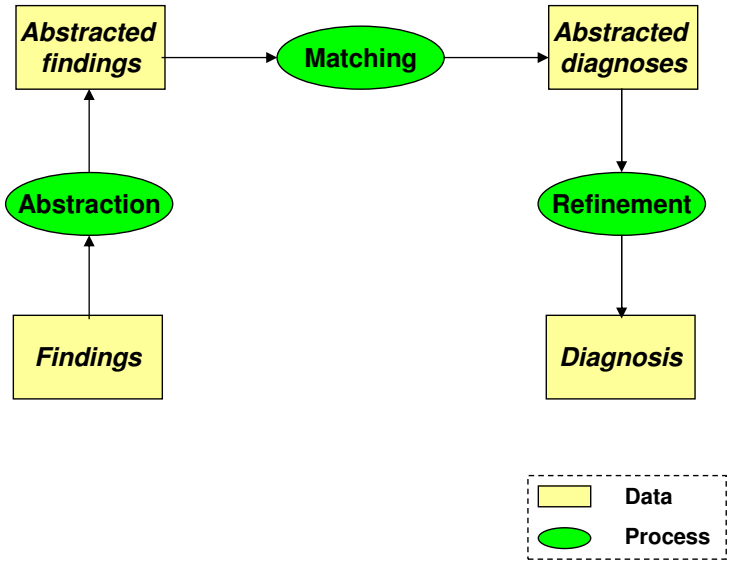



Figure 1: The three major processes of heuristic classification

# 2 A Conceptual Model for Service Discovery

In this section, we will first briefly introduce the gist of the matter of heuristic classification. This model is then applied to service discovery.

## 2.1 Heuristic Classification

[Cla85] provided a land marking analysis in the area of experts systems. Based on an analysis of numerous rule-based systems for classification and diagnosis he extracted a pattern of three inference steps that helped to understand the various production rules implemented in the various systems[1]. The problem-solving method he called *heuristic classification* separates abstraction, matching, and refinement as the three major activities in any classification and diagnosis task (see Figure 1).

**Abstraction.** Abstraction is the process of translating concrete description of a case into features that can be used for classifying the case. For example, the name of a patient can be ignored when making a diagnosis, his precise age may be translated into an age class, and his precise body temperature may be translated into the finding "low fever". The process is about extracting classification relevant features from a concrete case description.

**Matching.** Matching is the process of inferring potential explanation, diagnoses, or classifications from the extracted features. It matches the abstracted case description with abstract categories describing potential solutions.

**Refinement.** Refinement is the process of inferring a final diagnosis explaining the given findings. This process may include the acquisition of new features

[1] In [OF01] we already used heuristic classification as a model for improving the translation process between different XML schema dialects

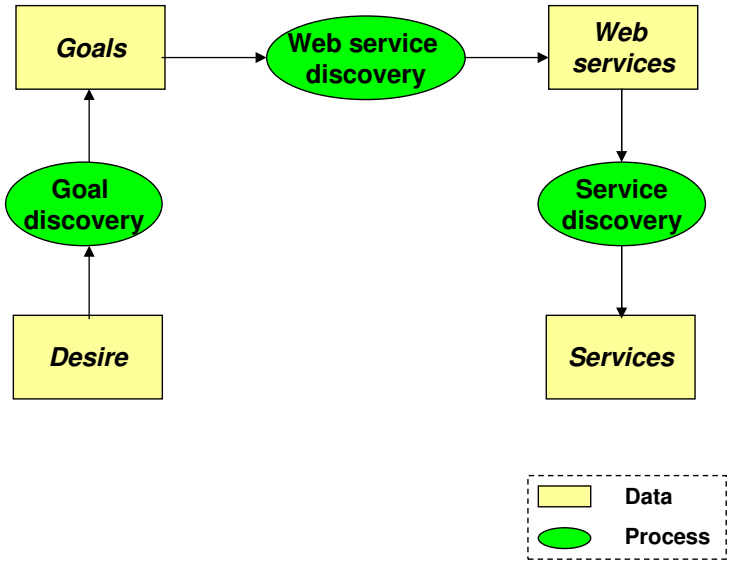



Figure 2: The three major processes in service discovery

describing the given case. However, it is now the potential solution that guides the acquisition process of these features.

As the latest step indicates, the entire process can be executed in an iterative fashion. Instead of acquiring all potential findings an initial set can be used to derive intermediate potential explanation that can be further used to guide the next iteration of the process.

## 2.2 Service Discovery

Now, what has this to do with web service or service discovery? We strongly believe that a scalable and workable service discovery approach has to follow the same pattern (see Figure 2).

**Abstracting goals from user desires.** Users may describe their desires in a very individual and specific way that makes immediate mapping with service descriptions very complicated. Therefore, each service discovery attempt requires a process where user expectations are mapped on more generic goal descriptions. Notice that this can be hidden by the fact that a discovery engine allows the user only to select from predefined goals. However, then it is simply the user who has to provide this mapping i.e. who has to translate his specific requirements and expectations into more generic goal descriptions. This step can be called **goal discovery**, i.e., the user or the discovery engine has to find a goal that describes (with different levels of accuracy) his requirements and desires. In the current literature on service and web service discovery this step is mostly neglected.

An example of such a user desire would be to buy a train ticket from Innsbruck to Venice, on December 12th 2004, and leaving Innsbruck between 15:00 and 17:00. It can be seen that this is a very concrete and detailed desire, while goals are intended to be **generic and reusable**, e.g. buy train tickets or buy train tickets in Europe. Therefore, a mapping from the user desire to generic goals becomes necessary.

In order to fully understand the difference between matching and refinement in the service discovery context we have to introduce the difference between services and web services (cf. [Pre04]) and, in consequence, between service and web service discovery. Let us take again a travelling scenario as a means to illustrate the difference. A customer may want to travel from Innsbruck to Venice and he is looking for a service that provides this to him. The service may be provided by an airline or a train company. This is the service he is looking for. In order to find (and buy) the service he is accessing a web service, i.e., a software artifact. This software artifact will not provide him the service to travel from Innsbruck to Venice (for this he needs a plane or a train) but it may help him to find this service. He will find a suitable web service based on the semantic annotations of available web services.

Actually, web services are means to find (and buy) services, i.e., they are to a large extent **service discovery engines and a discovery engine should not try to replace or duplicate this functionality**[2]. Using the previous travelling example, when somebody wants to travel from Innsbruck to Venice on a given date and with some time constraints, he looks for a web service which is offering travelling services, based on the semantic annotations of the web service, and then he consults the web service to check whether this trip can be done by train or plane, on the given date, and with the given time constraints. Taking the analogy with databases as illustration, web service discovery is about searching for databases that may contain instance data we are looking for, while service discovery is about finding the proper instance data by querying the discovered databases. The same analogy can be extended to consider services that imply some effects in the real world. Web service discovery is about searching for web services that can be used to e.g. buy tickets. Service discovery is about checking whether the ticket sellers offering such web services can really provide the concrete requested ticket.

Unfortunately, most approaches around service and web service discovery neglect this distinction leading to non-pragmatically assumptions and non-workable proposals for the discovery process e.g. [LH03a]. Assuming to find services based on the semantic annotation of web services requires complete and correct meta descriptions of all the services offered by a web service e.g. a train ticket seller has to include in the web service description information about all the available departures and destinations, on what dates and at what times, with what price, etc. This would imply that the semantic annotation and the discovery process duplicate most of the actual service of the web service. It is no longer necessary to execute a web service to find a service, rather the semantic annotation and the related reasoning provide this support.

We do not think that complete and correct descriptions of all the services offered by a web service are a realistic assumption; it would make web services no longer necessary (at least for the discovery process); and we wonder whether logical reasoning would scale under this conditions where the execution of an efficient program accessed via the web service is simulated by reasoning over its semantic annotation.

Alternatively one could assume to directly query the web service during the web service discovery process. However, this may lead to network and server overload and it makes a very strong assumption: in addition to data mediation, protocol and process mediation for the web service must be in place before the discovery process even starts. We do not think that this is a realistic assumption as we will further discuss in Section 3. In consequence we think it is essential

[2]Spoken more precisely, web services are the interface to a software artifact that may help to find and buy services (cf. [Pre04]), however, we neglect this distinction here and try the web service interface and the accessed software artifact identically.

to distinguish between web service and service discovery for coming up with a workable approach that scales and makes realistic assumptions in a pragmatic environment.

**Web service discovery.** Web service discovery is based on matching abstracted goal descriptions with semantic annotations of web services. This discovery process can only happen on an ontological level, i.e., it can only rely on conceptual and reusable. For this, two processes are required: a) the concrete user input has to be generalized to more abstract goal descriptions, and b) concrete services and their descriptions have to be abstracted to the classes of services a web service can provide. We believe that this twofold abstraction is essential for lifting web service discovery on an ontological level that is the prerequisite for a scalable and workable solution for it. In Section 4, different approaches to web service discovery will be discussed in detail.

**Service discovery.** Service discovery is based on the usage of web services for discovering actual services. Web service technology provides automated interfaces to the information provided by software artifacts that is needed to find, select, and eventually buy a real-world service or simply find the piece of information somebody is looking for. Service discovery requires strong mediation and wrapping, since the specific needs of a choreography[3] of a web service have to be met in order to interoperate with it. Notice that automatization of service discovery defines significant higher requirements on mediation than web service discovery, as it also requires protocol and process mediation. In a sense, the role of web service discovery can be compared with the role of an internet search engine like Google, and service discovery with the process of extracting the actual information from the retrieved web sites[4].

[3] See [LRK04] for a definition of choreography.

[4] Please notice that Froogle (http://froogle.google.com/) also extract information from the web sites. However, here we refer here to the general distinction between finding web sites and accessing their content.

# 3 Relation between Discovery and Mediation

In a distributed environment, different users and web services can use different terminologies, which leads to the need for mediation in order to allow heterogeneous parties to communicate. In this section we analyze the relation between discovery and mediation, identifying what kind of mediation is required in different scenarios.

## 3.1 Assumptions on Mediation

One could assume that web services and goals are described by the same terminology. Then no data mediation problem exists during the discovery process. However, it is unlikely that a potentially huge number of distributed and autonomous parties will agree before-hand in a common terminology.

Alternatively, one could assume that goals and web services are described by completely independent vocabularies. Although this case might happen in a real setting, discovery would be impossible to achieve. In consequence, only an intermediate approach can lead to a scenario where neither unrealistic assumptions nor complete failure of discovery has to occur. Such an scenario relies in three main assumptions:

- Goals and web services most likely use different vocabularies, or in other words, we do not restrict our approach to the case where both need to use the same vocabulary.
- Goals and web services use controlled vocabularies or ontologies to describe requested and provided services.
- There is some mediation service in place. Given the previous assumption, we can optimistically assume that a mapping has already been established between the used terminologies, not to facilitate our specific discovery problem but rather to support the general information exchange process between these terminologies.

Under these assumptions, we do not simply neglect the mapping problem by assuming that it does not exist and, at the same time, we do not simply declare discovery as a failure. We rather look for the minimal assumed mediation support that is a pre-requisite for successful discovery.

Notice that this has also been the approach taken in IBROW (cf. [BPM+98]), a project in the area of internet-based matchmaking of task descriptions and competence definitions of problem-solving methods. Both tasks and methods used different ontologies to describe their requests and services. However, both description ontologies were grounded in a common basic ontology that allowed theorem proving to rewrite the terms until equality could be proven.

## 3.2 Mediation requirements

In the following, we will discuss different technological scenarios for the discovery process and how this reshapes our assumption on the underlying mediation support that needs to be in place.

### 3.2.1 Natural Language Processing and Keywords

Processing written natural language input of a human user is commonly used to derive information for computer consumption. Stemming, part-of-speech tag-

ging, phrase recognition, synonym detection etc. can be used to derive machine-processable semantics from service requests and service descriptions. In the end, a set of keywords extracted from a service request are matched against a set of keywords extracted from a service description, or both requester and provider can use keywords to describe their requests and offers.

Stemming and synonym recognition already try to reduce different words to their joined underlying conceptual meaning. Still, this mediation support is very generic and only relies on general rules of language processing. Mediation can also be required for translating from a given human language into another one. Also in this case the mediation support required is generic and it can be assumed that it will be in place.

### 3.2.2 Controlled vocabularies & Ontologies

A different scenario is to assume that requester and provider use (not necessarily the same) controlled vocabularies[5]. In order to illustrate such scenario we will refer to controlled vocabularies for products and services. Efforts like UNSPSC[6] or eCl@ss[7] (among others) provide controlled vocabularies for describing products and services with around 15,000 concepts each. A service is described by a reference to a class in one of the classification schemas and a web service is described by the set of the classes that are used to describe the services that can be accessed through it. Similarly, a goal is described by a concept taken from a controlled vocabulary.

A goal discovers a web service if its concept is an element of the set of concepts describing the web service. Reasoning over hierarchical relationships may be required in case of taxonomies.

Mediation service is needed in case the requester and provider use different vocabularies. Since they use controlled instead of ad-hoc vocabularies, requiring such mediation service to be in place is not a completely unrealistic assumption. Notice that we do not assume a customized mediation service for our specific discovery task but rather generic alignments of generic business terminologies that may fulfill general information exchange needs between services and processes using these terminologies[8].

The border between controlled vocabularies and ontologies is thin and open for a smooth and incremental evolvement. Ontologies are consensual and formal conceptualizations of a domain (cf. [Gru93]). Controlled vocabularies organized in taxonomies and with defined attributes like eCl@ss resemble all necessary elements of an ontology. Ontologies may simply add some logical axioms for further restricting the semantics of the terminological elements. Notice that a service requester or provider gets these logical definition "for free". He can select a couple of concepts for annotating his service/request, but he does not need to write logical expression as long as he only reuses the ones already included in the ontology.

This scenario looks quite appealing since it adds semantic web facilities to the web service discovery eliminating two major risk factors of logic-based techniques:

[5]Notice that this does not necessarily imply that service provider and requester use directly these controlled vocabularies. A goal discovery process and an analogous process of discovering generic service descriptions can provide this service. Trials to mechanize such a process are described in [DKO+02].

[6]http://www.unspsc.org/

[7]http://www.eclass.de/

[8]Establishing this alignments is not an easy and straight-forward task as discussed in [Fen03].

- Effort in writing logical expressions. Writing down correct logical formulas is a very cumbersome process. A discovery approach that is assuming this on a large scale for requesters and provides leads to scalability problems.
- Effort in reasoning about logical expressions. Reasoning over complex logical statements is computationally very hard in case the formal language provides a certain level of expressivity. Therefore, it is important to decouple the reasoning process from the actual discovery process. With the discussed approach this is possible, as the goal as well as the service descriptions are abstracted from concrete specifications and inputs to a generic description at the level of an ontology. The logical relationship between these concepts can be derived independent from a concrete service request and the materialized inferences can simply be reused during a discovery process. Doing reasoning in the back-end mostly or totally decoupled from the actual discovery request ensures that reasoning does not become a bottleneck.

Similar to Section 3.2.2, mediation support is needed in case the requester and provider use different ontologies. However, since generic ontologies can be used, such an existing mediation service is not a completely unrealistic assumption. Again, we do not assume customized meditation service for our specific discovery task but rather generic alignments of generic ontologies[9].

### 3.2.3 Full-fledged logic

Simply reusing existing concept definitions as described in the previous section has the advantage of the simplicity in annotating services and in reasoning about them. However, this approach has limited flexibility, expressivity, and grain-size in describing services and request. Therefore, it is only suitable for scenarios where a more precise description of requests and services is not required. Furthermore, describing the functionality of a Web Service requires an expressivity which goes beyond the capabilities of current ontology languages. For these reasons, a full-fledged logic is required when a higher precision in the results of the discovery process is required.

Mediation can only be provided if the terminology used in the logical expressions is grounded in ontologies. Otherwise, it is impossible to prove the given logical relation between requests and service descriptions and discovery breaks down. Therefore, the mediation support required is the same as for ontology-based discovery (see Section **??**).

[9] See [dMRME04] for a survey on Ontology mapping and alignment.

# 4 Web Service Discovery

In this chapter we discuss different approaches to web service discovery in the WSMO framework which require different effort in annotation and description of both goals and services and deliver discovery results of different accuracy. Our focus in this document is primarily on *semantic-based approaches* to web service discovery. We believe that the different techniques altogether help to create a workable solution to the problem of web service discovery which addresses practical requirements and is based on realistic assumptions; our final goal here is thus ensure that the WSMO framework and its discovery component is adequate for a wide range of application scenarios with rather different requirements. As explained in Section 3, the single approaches can require different techniques for mediation. The most important technique for us certainly is ontology merging and alignment. The creation of mediators itself is outside the scope of this deliverable.

## 4.1 Keyword-based Discovery

The keyword-based discovery is a basic ingredient in a complete framework for semantic web service discovery. By performing a keyword base search the huge amount of available services can be filtered or ranked rather quickly. The focus of WSMO discovery is not in keyword-based discovery but we consider this kind of discovery a useful technique in a complete semantic web service discovery framework.

In a typical keyword-based scenario a keyword-based query engine is used to discover services. A query, which is basically a set of keywords, is provided as input to the query engine. The query engine match the keywords from to query against the keywords used to describe the service. A query with the same meaning can be formulated by using a synonyms dictionary, like WordNet [10] [Fel98]. The semantic of the query remains the same but because of the different keywords used, synonyms of previous ones, more services that possible fulfill user request are found. Moreover, by using dictionaries like WordNet as well as natural language processing techniques an increase of the semantic relevance of search results (wrt. to the search request) can principally be achieved [RS95]; nonetheless, such techniques are inherently restricted by the ambiguities of natural language and the lack of semantic understanding of natural language descriptions by algorithmic systems.

The services descriptions are provided in the terms of advertisements along with the keywords for categorization.

## 4.2 Discovery based on simple semantic Descriptions of Services

Although keyword-based search is a widely used technique for information retrieval, it does not use explicit, well-defined semantics. The keywords used to retrieve relevant information do not have an explicit formalization and, therefore, do not allow inferencing to improve the search results.

For these reasons, as a second approach we consider the use of controlled vocabularies with explicit, formal semantics. Ontologies, which offer a *formal, explicit specification of a shared conceptualization of some problem domain* [Gru93], are excellent and prominent conceptual means for this purpose. They provide an explicit and shared terminology, explicate interdependencies between

[10] The WordNet homepage: http://www.cogsci.princeton.edu/~wn/

single concepts and thus are well-suited for the description of web services and requester goals. Moreover, Ontologies can be formalized in logics which enables the use of inference services for exploiting knowledge about the problem domain during matchmaking and discovery.

In Section 4.2.1 we present a formal modelling approach for web services and goals which is based on set theory and exploits ontologies as formal, machine-processable representation of domain knowledge. We discuss web service discovery based on this approach for simple semantic descriptions. In Section 4.2.2 we finally discuss how to implement the set-based model in the formal framework of logic.

The framework that we present in this section for web service discovery based on simple semantic annotations is in fact similar to a model that has recently been proposed in [GMP04] for discovery in an e-Business setting. In some respects our approach is a generalization of the model discussed there.

**A running example.** Let us illustrate this approach with a few of examples from the travelling domain: we model some client goals and web service capabilities related to information about flights and train connections. We want to define the following informal sample goals and assume the requester wants to find a adequate web services:

$\mathcal{G}_1$: I want to know about all flights from any place in Ireland to any place in Austria.

$\mathcal{G}_2$: I want to know about some flights from any place in Ireland to any place in Ireland which is different form the starting location.

$\mathcal{G}_3$: I want to know about all flights from Galway (Ireland) to Dublin (Ireland).

$\mathcal{G}_4$: I want to know about some flights from Innsbruck (Austria) to some place in Ireland (the client does not necessarily care which one).

$\mathcal{G}_5$: I want to know about some train connections from Galway (Ireland) to Dublin (Ireland).

For the formalization of the example in our set-based approach we assume that an appropriate set of ontologies $\mathcal{O}$ for geographical data and travelling are in place. In particular, we have a relation $in(x, y)$ which states that object $x$ is geographically located in object $y$, a relation $flight(f, s, e)$ which states that $f$ is a flight from location $s$ to location $e$ and a relation $train(t, s, e)$ which states that $t$ is a train connection from location $s$ to location $e$.

Let us further assume four available web services $\mathcal{W}_1, \ldots, \mathcal{W}_4$ exposing the following informal capabilities:

$\mathcal{W}_1$ offers information about all flights for any place in Europe to any place in Europe.

$\mathcal{W}_2$ offers information about flights from all places in Ireland to all other places in Ireland, but not necessarily information about all such flights.

$\mathcal{W}_3$ offers information about all flights from all places in Ireland to Innsbruck (Austria).

$\mathcal{W}_4$ offers information about all train connections from Galway (Ireland) to Dublin (Ireland).

### 4.2.1 A set-based Modelling Approach

Frame-based or object-oriented modelling techniques have become popular principle approaches to system and domain modelling in both academia as well as the industry in recent years. One main characteristic of these approaches is that the problem domain (or the *universe*) is understood as a set of objects and single objects can be grouped together into sets (or *classes*). Each class captures common (syntactic and semantic) features of their elements. Features can be inherited between classes by defining class hierarchies. This way, a problem domain can be structured as classes of objects and is basically understood as a collection of classes (or sets of things). In particular, ontologies are a popular knowledge-representation technique which usually exploit the very same modelling paradigm.

In such modelling approaches the main semantic properties that one is interested in are certain relationships between such sets or objects of the universe. Establishing and checking such relationships is the main reasoning task which allows agents to exploit knowledge formalized in the domain model (or Ontology). From a knowledge representation perspective, this is a very natural and simple modelling approach for human beings.

**How to model Web Services and Goals?** A *web service* can be seen as computational object which can be invoked by a client. The invocation itself is based on web service technologies like SOAP and WSDL whereas the technical details of invocation are not relevant for discovery based on functional specifications. The resulting execution of the web service generates (wrt. a set of input values provided by the client) certain information as an output and achieves certain changes of the state of the world. An output as well as an effect can be considered as objects which can be embedded in some domain ontology, that means a formal conceptualization of the problem domain under consideration.

Hence, a service description should capture the objects that can be delivered by the service in principle. Obviously, one has two options here: Either one only considers the objects which can be delivered in the course of a *single execution* of the web service or the objects which can be delivered by executing the web service *an arbitrary number of times*. For the moment, we want to abstract from concrete executions of web services (and hence inputs) and thus only consider the latter case, that means services describe abstractly what they can deliver (independent of any input and number of executions by the user) as output and effects at all. We will come back on this issue in Section 4.3.1 where we discuss how to model the semantics of a service in a more detailed and precise manner as a set of executions.

*Goals* specify the desire of a client that he wants to have resolved after invoking a web service, that means they describe the information the client wants to receive as output of the web service execution as well as the effects on the state of the world that the client intends to achieve by invoking the web service. This desire can be represented as sets of elements which are relevant to the client as the outputs and the effects of a service execution. Thus and in line with the WSMO model [LRK04], goals refer to the state of the world which is desired by executing some web service.

According to this view, web services and goals are represented as *sets of objects* in the approach described here. The single descriptions of these sets refer to ontologies that capture general knowledge about the problem domains under consideration. Hence, the objects described in some web service description and the objects used in some goal description can or might be interrelated in

some way by ontologies. Eventually, such an interrelation is needed to establish a match between goals and services. In the general case, this interrelation has to explicated by a suitable mediator that resolves heterogenities amongst the terms used in goal and web service descriptions.

This way, web service and goal descriptions are become largely decoupled and modelers have reasonable flexibility when describing web services and goals. In particular, they do not have to know in detail about how the corresponding matching elements (i.e. services or goals) have precisely been described to actually ensure that a match can formally be established between compatible goals and web services – in fact, they do should never have to care about this question – but instead they only refer to domain ontologies which are not specific to a particular or a particular web service. Ontologies and ontology mediation provide a reasonable framework for decoupling of web service and goal descriptions as well as a flexible semantic-based matching which are considered as major desiderata for a matching engines in [PKPS02b].

In important observation in our approach is that the description of a set of objects for representing a goal or a web service actually can be interpreted in different ways and thus the description by means of a set is not semantically unique: A modeler basically might want to express that either *all* of the elements that are contained in the set are requested (in case of a goal description) or can be delivered (in case of a web service description), or that only *some* of these elements are requested (or can be delivered).

Clearly, a modeler has some specific intuition in mind when specifying such a set of relevant objects for a goal or web service description and this intention essentially determines whether we consider two descriptions to match or not. Thus, these intuition should be stated explicitly in the descriptions of service requests or service advertisements.

**Examples.** We consider the following goal and web service

$\mathcal{G}_4$: I want to know about some flights from Innsbruck (Austria) to some place in Ireland (the client does not necessarily care which one).

$\mathcal{W}_1$ offers information about all flights for any place in Europe to any place in Europe.

For the set-based specification, we refer to an appropriate set of ontologies $\mathcal{O}$ for geographical data and travelling are in place. Hence, we can use the following definitions for the set of relevant objects as well as the respective intentions:

| Goal / Web service | Set $R$ of relevant Objects | Intention of $R$ |
|---|---|---|
| $\mathcal{G}_4$ | $\{f \mid f$ is a flight starting at Innsbruck in Austria and ending at any city $c$ located in Ireland $\}$ | existential ($\exists$) |
| $\mathcal{W}_1$ | $\{f \mid f$ is a flight starting at city $s$ and ending at city $e$, $s$ any city in Europe, $e$ any city in Europe $\}$ | universal ($\forall$) |

**Remark.** For the sake of simplicity, we will consider in the following only outputs of a service and do not treat effects explicitly. The separation of effects and outputs in WSMO is a conceptual one and effects can basically be dealt with in

the very same way. Nonetheless, it is useful to distinguish both since they are conceptually different and we believe that it is beneficial for users to have the ability to apply *different criteria for matching* to outputs as well as effects in a service discovery request. To augment the model discussed here accordingly is a straightforward endavour.

**How to model semantic Matching between Web Services and Goals?** The semantics of goal as well as web service descriptions $\mathcal{D}$ is represented by a set of objects $R_{\mathcal{D}} \subseteq U$ (in a common universe $U$) which represent the set of *relevant objects* for the description as well as an explicit specification about the corresponding intention $I_{\mathcal{D}} \in \{\forall, \exists\}$ of the set.

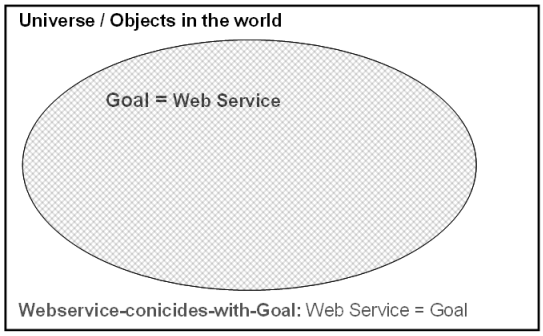
In order to consider a goal $\mathcal{G}$ and a web service $\mathcal{W}$ to match on a semantic level, the sets $R_{\mathcal{G}}$ and $R_{\mathcal{W}}$ describing these elements have to be interrelated somehow; precisely spoken, we expect that some set-theoretic relationship between $R_{\mathcal{G}}$ and $R_{\mathcal{W}}$ has to exist. The most basic set-theoretic relationships that one might consider are the following:

- Set equality: $R_{\mathcal{G}} = R_{\mathcal{W}}$
- Goal description subset of web service description: $R_{\mathcal{G}} \subseteq R_{\mathcal{W}}$
- Web service description subset of goal description: $R_{\mathcal{W}} \subseteq R_{\mathcal{G}}$
- Common element of goal and web service description: $R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset$
- No common element of goal and web service description: $R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset$

These set-theoretic relationships basically provide the basic means for formalizing our *intuitive understanding of a match* between goals and web services in the real-world. For this reason, they have been considered to some extent already in the literature, for instance in [LH03b] or [PKPS02b] in the context of service matchmaking based on Description Logics. The terminology for matching notions in these papers have been inspired by work done in the context of component matching based on component specifications [ZW97].

On the other hand, we have to keep in mind that in our model these sets actually only capture *one part of the semantics* of goals and service description, namely the relevant objects for the service requestor or service provider. The intentions of these sets in the semantic description of the goal or web service is not considered but clearly affects whether a certain existing set-theoretic relationship between $R_{\mathcal{G}}$ and $R_{\mathcal{W}}$ is considered to actually correspond (or formalize) a match in the intuitive sense. Hence, we have consider the intentions of the respective sets as well. In the following we will discuss the single set-theoretical relations as well as their interpretation in detail:

- **Set equality**: $R_{\mathcal{G}} = R_{\mathcal{W}}$



Here the objects that are advertised by the service provider (and which thus can potentially be delivered delivered by the described web service $\mathcal{W}$) and the set of relevant objects for the requester as specified in the goal $\mathcal{G}$ perfectly match, i.e. they coincide.

In other words, the service might be able to deliver all relevant objects (depending on the respective intention of of the $R_{\mathcal{W}}$). In any case, it is guaranteed that no irrelevant objects will be delivered by the service.

When considering the possible combinations of intentions $I_{\mathcal{W}}$ and $I_{\mathcal{G}}$ for $R_{\mathcal{W}}$ and $R_{\mathcal{G}}$, we get the following intuitive interpretations of the set-theoretic relationship in a real-world context:

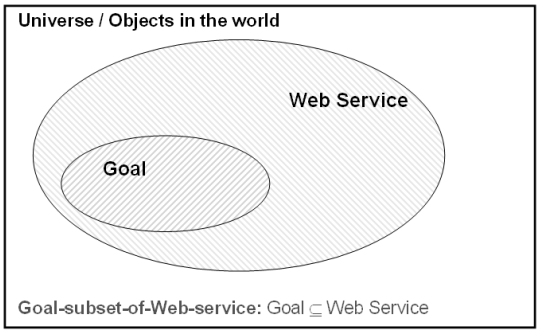
1. The service requester wants to get all of the objects that he has specified as relevant ($I_{\mathcal{G}} = \forall$), whereas the service provider claims that the web service is able to deliver all the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \forall$): In this case, the requester needs and the advertised service capability match perfectly.
2. The service requester wants to get some of the objects that he has specified as relevant ($I_{\mathcal{G}} = \exists$), whereas the service provider claims that the web service is able to deliver all the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \forall$): In this case, the requester needs and the advertised service capability again match perfectly. In a sense, the service even over-satisfies the needs of the requester.
3. The service requester wants to get all of the objects that he has specified as relevant ($I_{\mathcal{G}} = \forall$), whereas the service provider claims that the web service is able to deliver only some the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \exists$): In this case, the requester needs can not be fully satisfied by the service. At best, the service can contribute to resolve the desire of the client. Thus, we consider this case as a *partial match* only. Nonetheless, it is guaranteed that the service does not deliver objects which are irrelevant for the requester.
4. The service requester wants to get some of the objects that he has specified as relevant ($I_{\mathcal{G}} = \exists$), whereas the service provider claims that the web service is able to deliver only some the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \exists$): In this case, the requester needs and the advertised service capability again match intuitively. This time, the web service does not necessarily over-satisfy the needs of the requester.

In [LH03a] the situation where $R_{\mathcal{G}} = R_{\mathcal{W}}$ can be established is called *exact match*. However, in our model we do not necessarily consider the goal and the web service as matching.

- **Goal description subset of web service description**: $R_{\mathcal{G}} \subseteq R_{\mathcal{W}}$

  Here the relevant objects that are advertised by the service provider form a superset of the set of relevant objects for the requester as specified in the goal $\mathcal{G}$.

  In other words, the service might be able to deliver all relevant objects (depending on the respective intention of of the $R_{\mathcal{W}}$). Moreover, there is no guarantee that no irrelevant objects will be delivered by the service, i.e. it is possible that the service delivers objects that are irrelevant for the client as well.




When considering the possible combinations of intentions $I_{\mathcal{W}}$ and $I_{\mathcal{G}}$ for $R_{\mathcal{W}}$ and $R_{\mathcal{G}}$, we get the following intuitive interpretations of the set-theoretic relationship in a real-world context:

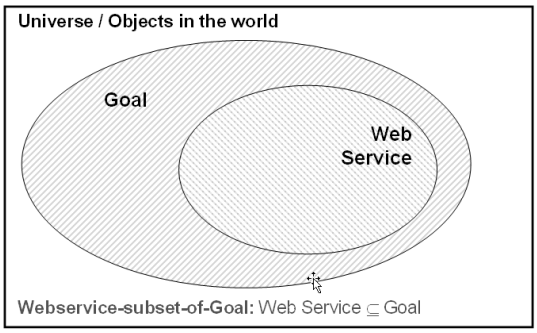
1. The service requester wants to get all of the objects that he has specified as relevant ($I_{\mathcal{G}} = \forall$), whereas the service provider claims that the web service is able to deliver all the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \forall$): In this case, the requester needs are fully covered by the web service.
2. The service requester wants to get some of the objects that he has specified as relevant ($I_{\mathcal{G}} = \exists$), whereas the service provider claims that the web service is able to deliver all the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \forall$): In this case, the requester needs are fully covered by the web service. In a sense, the service even over-satisfies the needs of the requester, since it actually is able to deliver all relevant objects for the client.
3. The service requester wants to get all of the objects that he has specified as relevant ($I_{\mathcal{G}} = \forall$), whereas the service provider claims that the web service is able to deliver only some the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \exists$): In this case, we can not determine whether there actually is a match, since we do not know which elements of $R_{\mathcal{W}}$ are actually delivered. It might happen, that the actually delivered set contains all the elements of $R_{\mathcal{G}}$ (in that case we would have indeed a match), whereas this Hence, we consider in this case the web service and the goal as *non-matching*.
4. The service requester wants to get some of the objects that he has specified as relevant ($I_{\mathcal{G}} = \exists$), whereas the service provider claims that the web service is able to deliver only some the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \exists$): In this case, the same holds as for the previous case and we consider this situation as a non-match as well.

In [LH03a] the situation where $R_{\mathcal{G}} \subseteq R_{\mathcal{W}}$ can be established is called *plugin match*. However, in our model we do not necessarily consider the goal and the web service as matching.

- **Web service description subset of goal description**: $R_{\mathcal{W}} \subseteq R_{\mathcal{G}}$

  Here the relevant objects that are advertised by the service provider only form a subset of the set of relevant objects for the requester as specified in the goal $\mathcal{G}$.

  In other words, the service in general is not able to deliver all objects that are relevant objects for the requester. But, there is a guarantee that no irrelevant objects will be delivered by the service.




When considering the possible combinations of intentions $I_{\mathcal{W}}$ and $I_{\mathcal{G}}$ for $R_{\mathcal{W}}$ and $R_{\mathcal{G}}$, we get the following intuitive interpretations of the set-theoretic relationship in a real-world context:

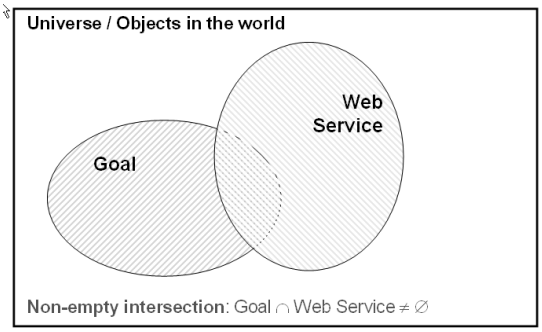
1. The service requester wants to get all of the objects that he has specified as relevant ($I_{\mathcal{G}} = \forall$), whereas the service provider claims that the web service is able to deliver all the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \forall$): In this case, the requester needs can not be fully satisfied by the service. At best, the service can contribute to resolve the desire of the client. Thus, we consider this case as a *partial match* only.
2. The service requester wants to get some of the objects that he has specified as relevant ($I_{\mathcal{G}} = \exists$), whereas the service provider claims that the web service is able to deliver all the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \forall$): In this case, the requester needs are fully covered by the web service. Additionally, the client will only receive objects which are relevant for him.
3. The service requester wants to get all of the objects that he has specified as relevant ($I_{\mathcal{G}} = \forall$), whereas the service provider claims that the web service is able to deliver only some the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \exists$): In this case, the requester needs can not be fully satisfied by the service but the service can at least contribute to resolve the desire of the client. Thus, we consider this case as well as a *partial match*.
4. The service requester wants to get some of the objects that he has specified as relevant ($I_{\mathcal{G}} = \exists$), whereas the service provider claims that the web service is able to deliver only some the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \exists$): In this case, the same holds as for the second case and we consider this situation as a match as well.

In [LH03a] the situation where $R_{\mathcal{W}} \subseteq R_{\mathcal{G}}$ can be established is called *subsumes match*. However, in our model we do not necessarily consider the goal and the web service as matching.

- **Common element of goal and web service description**: $R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset$

  Here there the set of relevant objects that are advertised by the service provider and the set of relevant objects for the requester have a non-empty intersection, i.e. there is an object (in the common) universe which is declared as relevant by both parties.

  In other words, the service in general is not able to deliver all objects that are relevant objects for the requester, but at least one such element can be delivered. Moreover, there is no guarantee that no irrelevant objects will be delivered by the service.




In a sense, this criterion can be seen as the weakest possible criterion for semantic matching in this set-based modelling approach.

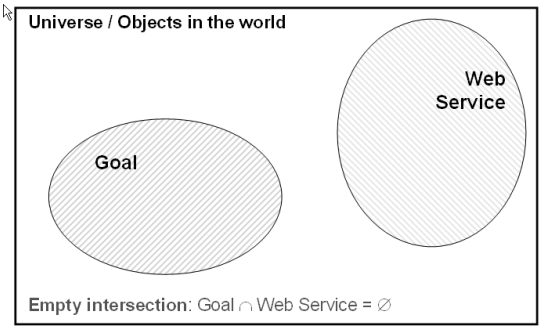
When considering the possible combinations of intentions $I_{\mathcal{W}}$ and $I_{\mathcal{G}}$ for $R_{\mathcal{W}}$ and $R_{\mathcal{G}}$, we get the following intuitive interpretations of the set-theoretic relationship in a real-world context:

1. The service requester wants to get all of the objects that he has specified as relevant ($I_{\mathcal{G}} = \forall$), whereas the service provider claims that the web service is able to deliver all the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \forall$): In this case, the requester needs can not be fully satisfied by the service. At best, the service can (weakly) contribute to resolve the desire of the client. Thus, we consider this case as a *partial match* only.
2. The service requester wants to get some of the objects that he has specified as relevant ($I_{\mathcal{G}} = \exists$), whereas the service provider claims that the web service is able to deliver all the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \forall$): In this case, the requester needs are fully covered by the web service. The requester might as well receive objects which are not relevant for him.
3. The service requester wants to get all of the objects that he has specified as relevant ($I_{\mathcal{G}} = \forall$), whereas the service provider claims that the web service is able to deliver only some the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \exists$): In this case, the requester needs are not fully covered. We are even not able to determine whether the service actually can deliver a common element of $R_{\mathcal{G}}$ and $R_{\mathcal{W}}$ and hence we consider this match as a non-match.
4. The service requester wants to get some of the objects that he has specified as relevant ($I_{\mathcal{G}} = \exists$), whereas the service provider claims that the web service is able to deliver only some the objects specified in $R_{\mathcal{W}}$ ($I_{\mathcal{W}} = \exists$): In this case, the same holds as for the previous case and we consider this situation as a non-match as well.

In [LH03a] the situation where $R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset$ can be established is called *intersection match*. However, in our model we do not necessarily consider the goal and the web service as matching.

- No common element of goal and web service description: $R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset$

  Here, the objects the web service description refers to and the objects the requester goal refers to are disjoint. That means there is no semantic link between both descriptions; they are talking about different things.




Hence, regardless of the corresponding intuitions for $R_{\mathcal{G}}$ and $R_{\mathcal{W}}$, we consider this situation as a non-match.

In [LH03a] the situation where $R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset$ can be established is called *disjointness*.

**Discussion.** Given some goal $\mathcal{G}$ and some web service $\mathcal{W}$, Figure 3 summarizes the discussion and shows under which circumstances the presence of which set-theoretic relationship between $R_{\mathcal{G}}$ and $R_{\mathcal{W}}$ is considered as a match ( Match ), a partial match ( PMatch ) or a non-match ( Nomatch ).

| Intention of $\mathcal{G}$ / $\mathcal{W}$ | $I_{\mathcal{W}} = \forall$ | | $I_{\mathcal{W}} = \exists$ | |
|---|---|---|---|---|
| $I_{\mathcal{G}} = \forall$ | $R_{\mathcal{G}} = R_{\mathcal{W}}$ | Match | $R_{\mathcal{G}} = R_{\mathcal{W}}$ | PMatch |
| | $R_{\mathcal{G}} \subseteq R_{\mathcal{W}}$ | Match | $R_{\mathcal{G}} \subseteq R_{\mathcal{W}}$ | Nomatch |
| | $R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ | PMatch | $R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ | PMatch |
| | $R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset$ | PMatch | $R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset$ | Nomatch |
| | $R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset$ | Nomatch | $R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset$ | Nomatch |
| $I_{\mathcal{G}} = \exists$ | $R_{\mathcal{G}} = R_{\mathcal{W}}$ | Match | $R_{\mathcal{G}} = R_{\mathcal{W}}$ | Match |
| | $R_{\mathcal{G}} \subseteq R_{\mathcal{W}}$ | Match | $R_{\mathcal{G}} \subseteq R_{\mathcal{W}}$ | Nomatch |
| | $R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ | Match | $R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ | Match |
| | $R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset$ | Match | $R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset$ | Nomatch |
| | $R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset$ | Nomatch | $R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset$ | Nomatch |

Figure 3: Interaction between set-theoretic criteria, intentions and our intuitive understanding of matching.

*Inconsistent descriptions for goals and services.* What have not considered so far is the possibility of inconsistent descriptions for goals and web services, that means descriptions where $R_{\mathcal{G}}$ or $R_{\mathcal{W}}$ are empty. Obviously, such descriptions do not make any sense: A requester who is asking for nothing as well as web services that do not deliver anything are simply superfluous and undesired. Nonetheless, they might occur in cases where the descriptions are quite complex or refer to several complex ontologies which are not themselves designed by the modeler.

Additionally, when just being ignored they can have an undesired impact on matching and thus discovery: Consider for example an inconsistent goal description, i.e. $R_{\mathcal{G}} = \emptyset$. If we check $\mathcal{G}$ for matching web services using the Plugin-criterion, i.e. $R_{\mathcal{G}} \subseteq R_{\mathcal{W}}$, then obviously *every* web service matches. For a user (who is not aware that his description is inconsistent, since otherwise he would usually not pose the query) the result would seem rather strange and

even incorrect because all checked services actually will be matched. From a logical perspective this is indeed not wrong[11], one the other hand it does not seem to be the best way to deal with the situation, since the user gets neither a hint that his goal description is inconsistent nor he gets (from his perspective) reasonable results.

There are multiple ways to deal with inconsistent descriptions in matching and discovery:

- *Ignore the possibility of inconsistent descriptions.* This basically means to apply the *Garbage-In-Garbage-Out*-principle and can lead to results which are unexpected by the requester or provider (see the example above).
- *Do not allow the advertise inconsistent goal and web service descriptions.* If somebody advertises a goal or service description which denotes an empty set, then just reject the description before it can take part in any discovery. During matching and discovery we work only with consistent descriptions. One has to be careful when considering the consistency-state of some goal or web service description: the consistency does not depend exclusively on the description itself but as well an all ontologies that the description refers to. Hence, changes to such ontologies potentially can lead to inconsistent goal and web service descriptions. For this reason, one might have to reconsider the consistency of all depending goal and web service descriptions that refer to some ontology when changing the ontology.
- *Check for inconsistency of goal and web service descriptions during a match and return acceptable results.* In this case we do not assume the consistency of any of the descriptions which are involved, but we check the consistency during the matching phase and deliver reasonable results or inform the requester or web service provider that something is wrong with the advertised description. Checking the consistency basically means to determine whether there is some element $e \in U$ such that it satisfies the properties of the given goal (or web service) description. It is not hard to extent the criteria in Figure 3 in a proper way.

In principle, every mentioned approach is possible. One has to decide for a concrete approach when actually implementing a discovery component. Then, one should be aware of the underlying assumptions and consequences of the selected approach. Personally, we would favor the last mentioned approach and check for inconsistency of description in the matching phase.

*Intentions.* In DL based approaches to service discovery like [PKPS02b], [LH03a] the notion of ,,intention" has at present not been reflected explicitly. As we have shown above, intentions capture an important aspect of goal and web service descriptions and affect essentially the situations in which certain set-theoretic criteria represent our intuitive understanding of matches properly.

The DL approaches so far basically can be understood as covering only the lower left part of the table, namely the situation where a goal has an existential intention and a web service has universal intention, i.e. $I_{\mathcal{G}} = \exists, I_{\mathcal{W}} = \forall$.

Figure 3 shows in detail how the overall picture is affected when Intention come into play. In contrast to the DL approaches so far where intentions of goal and web service descriptions are fixed (and thus can not be affected by the modelers), we believe that it is useful to give modelers additional freedom for capturing the precisely their intuitive understanding of their descriptions.

[11] Following the *Garbage-In-Garbage-Out*-principle!

Moreover, we believe that certain pairs of intentions will occur more often in practice than others: web service providers for example have a strong interest in their web services being discovered. If we compare the number of possible matchings with a given goal under existential and universal intentions, it seems most likely that providers tend to use universal intentions, even if the description does not necessarily model the real-world capability of the service accurately and promises too much. However, if a service provider wants to be more accurate with his web service description then in many situations he would have to use the existential intention. Further, please note, that even under existential intention, there are actual matches detectable, when the requester uses goals with existential intention.

For service requesters (in particular in an e-Business setting) we expect that the existential intention will suffice in many situations, however the requester has the freedom to properly express stronger requests than existential goals (using universal intention) if he needs to and thus get more accurate results in these situations.

*Set-theoretic criteria.* As shown in Figure 3, we basically have for each pair of intentions for a goal and a web service *several* formal criteria that capture actual matches, partial matches as well as non-matches. The question arises whether this is needed or what we gain from distinguishing the single notions. In case that indeed such a distinction is not useful or needed, we have to find out which of the several candidate criteria is the ,,right" one. In this paragraph we want to investigate and answer these questions.

According to most elementary Set-theory the single criteria are not completely separated, but the following interdependencies hold:

$$\begin{array}{l}\text{For any descriptions } R_{\mathcal{G}}, R_{\mathcal{W}} \subseteq U: \\ \quad R_{\mathcal{G}} = R_{\mathcal{W}} \Rightarrow R_{\mathcal{G}} \subseteq R_{\mathcal{W}} \\ \quad R_{\mathcal{G}} = R_{\mathcal{W}} \Rightarrow R_{\mathcal{G}} \supseteq R_{\mathcal{W}} \\ \quad R_{\mathcal{G}} \subseteq R_{\mathcal{W}}, R_{\mathcal{G}} \neq \emptyset \Rightarrow R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset \\ \quad R_{\mathcal{G}} \supseteq R_{\mathcal{W}}, R_{\mathcal{W}} \neq \emptyset \Rightarrow R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset \\ \quad R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset, R_{\mathcal{G}} \neq \emptyset, R_{\mathcal{W}} \neq \emptyset \Rightarrow R_{\mathcal{G}} \not\subseteq R_{\mathcal{W}}, R_{\mathcal{G}} \not\supseteq R_{\mathcal{W}}, R_{\mathcal{G}} \neq R_{\mathcal{W}}\end{array} \tag{1}$$

That means that certain formal set-theoretic criteria that we consider here are logically stronger notions than others: if the stronger relationship holds than the weaker relationship must holds as well. Using these properties, we can partially order the set-theoretic criteria: $C_1 \preceq C_2$ iff $C_2$ is logically weaker (or equivalent) than $C_1$, i.e.

$$(R_{\mathcal{G}} = R_{\mathcal{W}}) \quad \preceq \quad (R_{\mathcal{G}} \subseteq R_{\mathcal{W}}), (R_{\mathcal{G}} \supseteq R_{\mathcal{W}}) \quad \preceq \quad (R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset)$$

Given two particular intentions for a goal and a service description, let be $C_1$ a criterion which captures an actual match wrt. the given intentions and $C_2$ be a logically weaker criterion $C_1 \preceq C_2$, then $C_2$ denotes a match as well. Clearly, there is always a weakest criterion which denotes an actual match (wrt. $\preceq$), if there is a criterion which denotes an actual match at all. The weakest criterion for an actual match (wrt. given intentions $I_{\mathcal{G}}$ and $I_{\mathcal{W}}$ does not have to be the only criterion denoting an actual match.

Thus, if $C_1 \preceq C_2$, $C_1$ represents more semantic information about the match than $C_2$, in other words it provides additional knowledge about the specific kind of relationship between matching goal and web service descriptions (besides mere matching in the sense of $C_2$). If a specific criterion $C_1$ is used during the matching phase which is *not* the weakest criterion for an actual match and a

match can be detected, then additional properties on the kind of interrelation besides the mere fact that there match.

By requesting the use of a particular criterion for the matching between goal and web service descriptions, a service requester basically could exploit this property during a discovery process in order to ensure certain convenient properties from the discovered web services. We will investigate this aspect briefly:

- The *Intersection-Match* reflects *no special additional information* for the search request (beyond what is given in the goal description $R_{\mathcal{G}}$ and its corresponding intention $I_{\mathcal{G}}$). In particular, in case that this criterion denotes an actual match (wrt. the given Intentions) there is no weaker criterion which denotes a match as well.

- The *Subsumes-Match* reflects the additional property that *only relevant elements* will be delivered. This holds regardless of the intention which applies to a web service description. Thus, if the user insists on getting only web services which do not deliver items that are not of interest for resolving his goal then he should request explicitly that for matching a criterion is used which represents an actual match and is $\preceq$-smaller than the Subsumes-match.

- The *Plugin-Match* on the other hand reflects the additional property that *all relevant elements* can be delivered by a service (in case of universal intention of a web service) or (in case of an existential intention a lot weaker) that all elements that are relevant for the requester are relevant for the service as well. There might be objects which are relevant for the web service description but not for the service requestor.

  If the service request has existential intention then obviously this property is not interesting for the requester, since otherwise he would have chosen a universal intention. In the case of a universal intention of the service request, this property is automatically guaranteed, since the weakest criterion for an actual match is the Plugin-Match.

  Hence, it does not make any sense for the user to use this criterion for detecting matches if a weaker criterion (wrt. given intentions) applies as well.

- Finally, the *Exact-Match* precisely combines the *Subsumes-* and the *Plugin-criterion* and thus specifies that objects the web service description refers to and the objects the requester refers to precisely match; In particular, it holds (independent of the intention of the web service description) that irrelevant objects will not be delivered by the service. For the property that is represented by the Plugin-Match part, the same argument as for the Plugin-Match holds. Hence, the corresponding semantic property is irrelevant and the Exact-Match basically coincides (in the context of our discussion in this paragraph) with the Subsumes match.

To sum up, we have seen that there are cases where a client could benefit from exploiting the additional semantics captured by matching criteria that are stronger (i.e. $\preceq$-smaller) than the weakest (i.e. $\preceq$-maximal) criterion which represents an actual match. Hence, it makes sense to not only allow the use of the weakest (i.e. $\preceq$-maximal) criterion that actually denotes a match (for the respective intentions of the goal and the web service) to be applied for matching but to allow the user to manually ,,raise" the semantic requirements that are captured by the criterion to apply and thus to reflect his interest faithfully.

We have seen as well that in our general framework there is only one such additional property that actually can be considered as useful, namely the property of a web service to not deliver objects that are irrelevant to the user.

In order to ensure that this property is satisfied when matching, the discovery component has to apply a $\preceq$-maximal criterion, which is $\preceq$-smaller than the Subsumes criterion $R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ (that means either $R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ itself or $R_{\mathcal{G}} = R_{\mathcal{W}}$) and represents an actual match (wrt. the given intentions of the goal and the web service).

In case that the requester does not want to have this specific property reflected in the discovery result, the appropriate criterion to apply for matching clearly is one which is $\preceq$-maximal among the criteria that represent an actual match, since it is the weakest that formalizes our intuition understanding of a real-world match and thus does not restrict the set of possible matches unnecessarily.

Basically the same holds if we want to check whether a web service partially matches a given goal: we have to apply the criterion which is $\preceq$-maximal among the criteria that represent a partial match.

Finally, we have to for detecting non-matches we simply check for matches or partial matches. If neither criterions is satisfied, we have established a non-match.

Figure 4 represents the result of the discussion in case that the requester just gives a goal description, whereas Figure 5 describes the situation when additionally the requester wants to avoid web services that might deliver irrelevant objects.

| Intention of $\mathcal{G}$ / $\mathcal{W}$ | $I_{\mathcal{W}} = \forall$ | | $I_{\mathcal{W}} = \exists$ | |
|---|---|---|---|---|
| $I_{\mathcal{G}} = \forall$ | Match | $R_{\mathcal{G}} \subseteq R_{\mathcal{W}}$ | Match | **Not possible** |
| | PMatch | $R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset$ | PMatch | $R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ |
| | Nomatch | $R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset$ | Nomatch | $\neg R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ |
| $I_{\mathcal{G}} = \exists$ | Match | $R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset$ | Match | $R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ |
| | PMatch | **Not possible** | PMatch | **Not possible** |
| | Nomatch | $R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset$ | Nomatch | $\neg R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ |

Figure 4: Which formal criteria should be used for checking different degrees of matching.

| Intention of $\mathcal{G}$ / $\mathcal{W}$ | $I_{\mathcal{W}} = \forall$ | | $I_{\mathcal{W}} = \exists$ | |
|---|---|---|---|---|
| $I_{\mathcal{G}} = \forall$ | Match | $R_{\mathcal{G}} = R_{\mathcal{W}}$ | Match | **Not possible** |
| | PMatch | $R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset$ | PMatch | $R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ |
| | Nomatch | $R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset$ | Nomatch | $\neg R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ |
| $I_{\mathcal{G}} = \exists$ | Match | $R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ | Match | $R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ |
| | PMatch | $R_{\mathcal{G}} \cap R_{\mathcal{W}} \neq \emptyset$ | PMatch | **Not possible** |
| | Nomatch | $R_{\mathcal{G}} \cap R_{\mathcal{W}} = \emptyset$ | Nomatch | $\neg R_{\mathcal{G}} \supseteq R_{\mathcal{W}}$ |

Figure 5: Which formal criteria should be used for checking different degrees of matching when a requester insists on services delivering relevant objects only.

*Matching scenario.* During the discovery process the scenario for matching between goal and web service descriptions in general is as follows: A requester

specifies his goal by means of a set of relevant objects and the respective intention. Moreover, he might specify that he is only interested in web services which deliver only objects that are relevant for his goal. Furthermore, the requester can indicate in his request whether he is interested in partial matches, in case that no actual matches can be detected.

If the discovery request of the client contains only $(R_{\mathcal{G}}, I_{\mathcal{G}})$, then we check for a match using the respective criterion for matching under intentions $(I_{\mathcal{G}}, I_{\mathcal{W}})$ from Figure 4. In case of a detected match, we store the web service in a list with actual matches. On the other hand, if a match has not been detected and the discovery request indicates that the user is interested in partial matches, we check for a partial match using the corresponding criterion from the same table. If a partial match has been detected we store the web service in a list with partial matches. Eventually, we return the list of actual matches and (if the request indicates that) the list of partial matches.

If the discovery request of the client specifies besides $(R_{\mathcal{G}}, I_{\mathcal{G}})$ that only web services are requested that deliver relevant objects only, then proceed in the very same way, but apply the criterions with respect to Figure 5 instead.

The discussion shows that during discovery and matching intentions can be dealt with on a meta-level (in comparison to the set-theoretic notions), i.e. they do not directly affect single set-theoretic criterions and the respective checks themselves. Hence, we just need an implementation of the different set-theoretic criteria in order to realize a system for matchmaking. We will discuss a logic based approach as a general and elegant candidate approach for the implementation of these formal criteria in Section 4.2.2.

*Concluding remarks.* It should be clear that a detected match in this framework for web service discovery based is based on simple semantic annotation only and thus can not provide very strong guarantees on the actual accuracy of the results in each case: A detected match between a goal and a web service actually does not ensure that the web service can really be used for resolving the goal in the real-world, since important information that affect this possibility is not specified in the descriptions: Is the requester actually able to satisfy the requirements of the web service when invoking and interaction with the service, namely the preconditions as well as a prescribed choreography.

Nevertheless, the approach is based on a formal semantic model and uses formal domain knowledge to detect matches. Thus, the achievable accuracy of the approach (although being inherently limited) in general will be a lot higher than with keyword-based approaches.

In this respect, it still can be considered as a semantic-driven heuristic for locating web service which might can resolve the goal of a service requester.

### 4.2.2 Realization of the Approach in Logic

In the previous section we discussed an approach for goal and web service modelling as well as discovery that is conceptually grounded in Set-theory. This lead to a simple yet semantic model for modelling and matching of goals and services. Nonetheless, the question arises how to implement the framework in a way that allows machines to exploit the respective descriptions and establish semantic-based matchings between web services and goals. We will provide an adequate answer to this question in this section by showing how to implement the set-based modelling approach from Section 4.2.1 in the framework of logic.

Description Logics (DLs) [BCM+03] are important examples for formal languages that are equipped with a direct set-theoretic semantics: expressions in

DL languages represent sets of objects. Sets can be constructed from several other sets using specific (set-theoretic) operators. Formulae in Description Logics basically describe set-theoretic relations such as set-inclusion.

Hence, they are a natural target language for the implementation of the approach discussed in the previous section. There are already numerous publications (e.g. [PKPS02b], [LH03a], [GMP04]) that have investigated the use of DL for a discovery framework. Most of them, do not clearly separate between a conceptual model and a concrete logical framework that implements the conceptual model. A very nice exception is [GMP04]. Description Logic languages in general restrict the way one can characterize the set of instances of a concept (i.e. its *extension*). This restriction is mainly based on the interest to have certain guarantees wrt. computational behaviour: The relevant inferencing tasks in Description Logics, such as subsumption or satisfiability checking, are decidable.

In the formalization of our simple modelling approach that is outlined in this paragraph we basically aim to exploit the same modelling style for goal and web services as DLs but do not enforce any specific restriction on the concept language, though; we basically allow a rich (first-order) language for describing sets of elements. For concrete applications it is then up to the designers to choose an appropriate language (and a corresponding reasoning system) that fits their needs wrt. expressivity and computational characteristics. This could be for example a Description Logic like $\mathcal{SHIQ}$ or $\mathcal{SHIQ}(\mathbf{D})$ [BCM+03, HST00], a full first-order language like WSML-Full [dB04] or restricted sublanguages thereof.

There are some reasons that justify the decision to consider a first-order framework for our discussion here:

- There are *several elements* involved in checking a formal criterion which enables us to detect matches, namely a web service description, a goal description, ontologies and perhaps auxilliary definitions that are not relate to any ontology. As soon as one one of these elements can not be formalized in DLs properly, the whole approach is breaks and is not applicable. We want to avoid this situation. Actually, [GMP04] gives a concrete and simple example where common DLs as modelling languages lack sufficient expressiveness: they give a specific web service and a specific goal which we would expect to match, but using plain DL it is not possible to formalize them in such a way that the desired formalization of the matching criterion actually detects the intuitive match.

  On the other hand, manifold applications of logics for formal modelling provide enough evidence, that first-order languages are expressive enough for most applications.

- A first-order framework can provide and integration platform for descriptions in various languages. As long as there is a semantics-preserving mapping into the first-order framework, descriptions in several different languages can be combined with each other. That means matching across different modelling languages becomes basically possible.

- In our opinion, the proof obligations that formalize the checks of the single set-theoretic criteria discussed in Section 4.2.1 are more comprehensive.

- In case that for a specific application a less expressive language and logical framework is sufficient, it should not be too hard to transfer the proof obligation given below into the respective framework and thus to carry over the results presented in this section.

**How to model Web Services and Goals logically?** In the simple set-based approach that we presented in the previous Section we described web services and goals in terms of sets $R_{\mathcal{W}}$ and $R_{\mathcal{G}}$[12]. Formally, we represent the sets used for describing the semantics of a web service by means of *unary predicates* in a first-order language. All elements of the universe which satisfy the predicate (according to its definition) are considered to be elements of the set.

The precise semantics of the predicate symbol $ws_{\mathcal{W}}(x)$ is defined by means of a set $\mathcal{W}$ of first-order formulae. In general, we assume that the formulae in $\mathcal{W}$ refer some set of ontologies. Hence, these ontologies have to be formally represented as well, in order to properly define the semantics of the predicate symbol $ws_{\mathcal{W}}(x)$ in a formal way. The formal representation of these ontologies as a logical theory in our first-order framework is denoted here by $\mathcal{O}$.

The set $\mathcal{W}$ of defining formulae must be chosen in such a way that under every interpretation $I$ which is a model of $\mathcal{W}$ as well as the ontologies $\mathcal{O}$ to which the definition refers to, the predicate is interpreted as the respective set of relevant objects, i.e.

$$I \models \mathcal{W} \Leftrightarrow I(ws_{\mathcal{W}}) = R_{\mathcal{W}} \tag{2}$$

for every interpretation $I$.

In this formal sense, $\mathcal{W}$ (and $\mathcal{O}$) determine the interpretation of the symbol $ws_{\mathcal{W}}$ in the intended way. Obviously, the symbol $ws_{\mathcal{W}}$ has to occur in $\mathcal{W}$.

In the simplest case, we can just use a single formula of the form

$$\mathcal{W} : \forall x.(\psi(x) \leftrightarrow ws_{\mathcal{W}}(x)) \tag{3}$$

where $\psi(x)$ is an arbitrary first-order formula with exactly one free variable $x$.

For goals $\mathcal{G}$ we do the same and define an unary predicate symbol $g_{\mathcal{G}}$ by means of a set $\mathcal{G}$ (closed) first-order formulae and the and the formalization $\mathcal{O}$ of the set of ontologies which the definition refers to. Again, the definition $\mathcal{G}$ has to have the satisfy the property

$$I \models \mathcal{G} \Leftrightarrow I(g_{\mathcal{G}}) = R_{\mathcal{G}} \tag{4}$$

for every interpretation $I$.

Again, in the simplest case $g_{\mathcal{G}}(x)$ can be defined by a single formula of the form

$$\mathcal{G} : \forall x.(\phi(x) \leftrightarrow g_{\mathcal{G}}(x)) \tag{5}$$

where $\phi(x)$ is an arbitrary first-order formula with exactly one free variable $x$.

**How to model the set-theoretic criteria logically?** Given two logical definitions $W$ of a web service and $\mathcal{G}$ of a goal, we now want to show how to represent the set-theoretic criteria from Section 4.2.1 in our logical framework.

Similar to [LH03b], we distinguished five types of criteria:

- **Exact-Match.** Here the sets of relevant objects of the web service description and the goal description coincide: $R_{\mathcal{G}} = R_{\mathcal{W}}$. Each element $e \in U$ of the universe $U$ which is in $R_{\mathcal{G}} \subseteq U$ is as well in $R_{\mathcal{W}} \subseteq U$ and vise versa.

[12] Again, for the sake of simplicity we always mention a single set to describe a service (or a goal, resp.) although it should be clear that in WSMO we distinguish between service outputs and effects! To adapt the basic approach is straightforward and thus we skip this detail here to keep the discussion as simple as possible.

Formally, that means that we have to prove the following:

$$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \forall x.(g_{\mathcal{G}}(x) \leftrightarrow ws_{\mathcal{W}}(x)) \tag{6}$$

where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

In this case we write $\mathcal{W} \equiv_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

- **Subsumption-Match.** Here the set of relevant objects for the web service is a subset of the set of relevant objects of the goal: $R_{\mathcal{W}} \subseteq R_{\mathcal{G}}$. Each element $e \in U$ of the universe $U$ which is in $R_{\mathcal{W}} \subseteq U$ is as well in $R_{\mathcal{G}} \subseteq U$.

  Formally, that means that we have to prove the following:

$$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \forall x.(ws_{\mathcal{W}}(x) \rightarrow g_{\mathcal{G}}(x)) \tag{7}$$

  where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

  In this case we write $\mathcal{W} \sqsubseteq_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

- **Plugin-Match.** Here the set of relevant objects for the web service is a superset of the set of relevant objects of the goal: $R_{\mathcal{G}} \subseteq R_{\mathcal{W}}$. Each element $e \in U$ of the universe $U$ which is in $R_{\mathcal{G}} \subseteq U$ is as well in $R_{\mathcal{W}} \subseteq U$.

  Formally, that means that we have to prove the following:

$$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \forall x.(g_{\mathcal{G}}(x) \rightarrow ws_{\mathcal{W}}(x)) \tag{8}$$

  where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

  In this case we write $\mathcal{W} \sqsupseteq_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

- **Intersection-Match.** Here the intersection between the set of relevant objects for the web service and the set of relevant objects of the goal is not the empty set: $R_{\mathcal{W}} \cap R_{\mathcal{G}} \neq \emptyset$. There is an element $e \in U$ of the universe $U$ which is in both $R_{\mathcal{W}} \subseteq U$ and $R_{\mathcal{G}} \subseteq U$.

  Formally, that means that we have to prove the following:

$$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \exists x.(g_{\mathcal{G}}(x) \wedge ws_{\mathcal{W}}(x)) \tag{9}$$

  where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

  In this case we write $\mathcal{W} \sqcap_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

- **Non-Match.** Here the intersection between the set of relevant objects for the web service and the set of relevant objects of the goal is the empty set: $R_{\mathcal{W}} \cap R_{\mathcal{G}} = \emptyset$. There is no element $e \in U$ of the universe $U$ which is in both $R_{\mathcal{W}} \subseteq U$ and $R_{\mathcal{G}} \subseteq U$.

  Formally, that means that we have to prove the following:

$$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \neg\exists x.(g_{\mathcal{G}}(x) \wedge ws_{\mathcal{W}}(x)) \tag{10}$$

  where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

  In this case we write $\mathcal{W} \| _{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of non-match.

*Advantages and Disadvantages of the set-based Modelling Approach.* We briefly summarize the main advantages and drawbacks of the discussed modelling approach:

+ This modelling approach is based on a very simple and intuitive perspective of the world where everything is considered in terms of sets (or concepts)

+ The approach represents a general framework which does not fix the language to be used for describing goals and web services. In particular, it allows in general description which are not possible to express using Description Logics and thus provides increased expressiveness[13] over purely Description Logic-based approaches like [PKPS02c, LH03a].

+ Because of the same conceptual modelling style, this approach potentially allows a seamless integration of descriptions formalized in different languages.

- This approach does not capture the actual relation between service input and the corresponding outputs. Thus, the semantics of a web service is only described in a conceptual manner. In fact, this can be too coarse-grained for enabling the automation of the discovery and later execution of a service.

- The increased expressive power comes of course at a cost: Reasoning with these description is not necessarily guaranteed to be decidable any longer. Nonetheless, this does not have to be a problem in applications. Furthermore, the proposed approach is very flexible and allows to use restrictions[14] on the expressiveness of the allowed definitions for web services and goals such that the main reasoning tasks are decidable too (e.g. formulae in the Guarded-Fragment).

Indeed, the first mentioned disadvantage is not a real issue for people who are interest to model the service capability in detail: We will demonstrate later, that it is even possible to lift this modelling approach from the level of simple semantic annotations to the level of complex semantic descriptions where we describe a service capability by means of the actual relation between input values and generated outputs and effects.

## 4.3 Discovery based on rich semantic Descriptions of Services

Using simple semantic annotations for a service as it it described in the previous section basically adds machine-processable semantic information to service descriptions which allows a discovery mechanism to exploit this semantics during the discovery process and deliver results with high precision and recall.

Nonetheless, the kind of semantic information that can be expressed in that approach is limited wrt. the details of the service characteristics that can be captured. Therefore, the precision that can be achieved by a discovery component is limited as well, if the user could already state more detailed information

[13] Common Description Logics like $\mathcal{SHIQ}(\mathbf{D})$ for instance can only express so-called *tree-like* first-order sentences [GHVD03] (a syntactically restricted form of the first-order language) whereas in the discussed approach arbitrary first-order formulae are allowed

[14] Different restrictions from the ones imposed by Description Logics and perhaps still more expressive.

in the discovery request (and wants to do so). For certain applications and users, in particular, if you aim at a high-degree of automation in a dynamic service-based software system, definitely an approach is needed which allows to model nuances of meaning of the functionality of web services and the goal that a client wants to have resolved.

An elaborate description of the single main entities which are involved in the discovery process, namely service capabilities and goals, can be achieved by refining the pure conceptual level (where services and goals are basically concepts) as described in the Section 4.2. This requires at least a rich modelling language which allows to talk about objects in a world and hence variables, constant symbols and perhaps function symbols. That means, on this level we definitely want to use a First-order language (or possibly some restricted subset of such a language).

However, this is not enough. At this level of description we are interested to describe how outputs and effects created by a service execution actually depend on the concrete input provided by the user when invoking a service. That means to consider a service as a *relation on an abstract state-space* and to capture the functionality provided by a service in these terms. Here, we would consider for services input, output, preconditions, assumptions, postconditions and effects of a service, whereas in WSMO (version 1.0) for goals we only consider the state of the world that is reached after the execution of a service and hence postconditions and effects.

In Section 4.3.1 we show how to extend the set-based modelling approach discussed in Section 4.2 to capture the actual relation implemented by the service as well. This will allow us to exploit the matching notions that we already discussed as well as give us additional notions that might be of interest in some situations.

In Section 4.3.2 we discuss a further approach which is based on Transaction Logic – an extension of First-order Logic that enables to specify the dynamics of logical theories (or knowledge bases) in a declarative way. Here, the state of the world (and the information space) is represented by a logical theory and since the execution of a service changes the state of the world (and/or the information space) it results in an update of the logical theory. In principle, the described approach can be implemented using the $\mathcal{F}$LORA-2 system [KLP+04].

### 4.3.1 A set-based Modelling Approach for Rich Service Descriptions

In Section 4.2, we described the capability of a service by means of a set of objects and interpreted the set as a description of all objects that the service is able to deliver. In particular, we abstracted from concrete executions of the web service and the relation between the pre-state of the web service execution and the corresponding post-state. In other words, a web service description can be seen as a purely ontological concept whose description potentially can be quite detailed and complex.

In this section we show how to extend the set-based modelling approach discussed in Section 4.2 to capture the actual relation implemented by the service as well. That means, we skip the abstraction step that we have decided to take in Section 4.2 and consider service executions explicitly. Thus, we increase the level of detail of our service model.

**The informal Service Model revisited.** According to our discussion in Section 4.2 , a web service can be seen as computational object which can be invoked by a client. At invocation time, the client provides all the information

needed by the web service to identify and deliver the concrete service that has been requested. The resulting execution of the web service generates (wrt. a set of input values) certain information as an output and achieves certain effects on the state of the world. Both, an output and as well as an effect can be considered as objects which can be embedded in some domain ontology.

So far we ignored inputs and their relations to outputs and effects of the web service execution and considered a web service as delivering a *single* set of objects. In fact, when considering actual executions of a web service, the sets describing the outputs and effects of the execution actually *depend on the provided input values*. Hence, a *web service execution* can be described by a set of outputs and a set of effects (for the *specific* values for input parameters), whereas a *web service* is seen as a collection of possible executions and thus should be modelled by a collection of sets of outputs and effects: one pair of such sets for each service execution (or concrete values for the input parameters of the web service).

Figure 6 illustrates the extended service model: When invoking a web service $ws$ with some specific input values $i_1, \ldots, i_n$ in some state of the world (*pre-state*), the execution of the service results in a (different) state of the world (*post-state*) where the service delivers a set of elements as its output ($ws^{out}(i_1, \ldots, i_n)$) as well as a set of effects ($ws^{eff}(i_1, \ldots, i_n)$).

In the WSMO framework the client specifies his desire as a goal. More precisely, the goal description consists of the specification of a set of desired information ($goal^{out}$) as well as a set of desired effects ($goal^{eff}$).

As before, matching mainly is based on checking certain set-theoretic relationships between the sets related to the output ($ws^{out}(i_1, \ldots, i_n)$ and $goal^{out}$) as well as the sets related to effects ($ws^{eff}(i_1, \ldots, i_n)$ and $goal^{eff}$). The interpretation of the various relationships between those sets has already been discussed in detail in Section 4.2. Intentions for goal and web service descriptions are independent of the discussion here, and thus can be dealt with in the very same way as in our model for simple semantic annotations. Since intentions do not affect the logical representation of our set-theoretic relationship and are basically during matching considered on a meta-level on top of the set-theoretic criteria, we do not explain them explicitly here again.

Additionally, we can enrich our set of matching notions given in Section 4.2 by an orthogonal dimension[15]: We can express that we can satisfy a particular matching notion wrt. a *single execution* of a web service as well as wrt. an *arbitrary number of web service executions*. This results in additional matching notions that capture additional semantics in a given discovery request.

We want to illustrate the difference of the two new option by means of a simple example:

Imagine the following goal a user

> ,,I want to know about all sports events in Tyrol today"

and a web service with the following capability

> ,,The service delivers all events for a given city in Austria for today"

Given appropriate domain knowledge about Austria and events the service can not deliver the requested information by a single execution of the web service, but instead it can actually deliver *all* requested information, if it is invoked several times (i.e. for each city in the region ,,Tyrol").

[15]This dimension precisely corresponds to the feature that we added when refining service model from Section 4.2: we can now talk about web service executions instead of a single abstract web service concept.

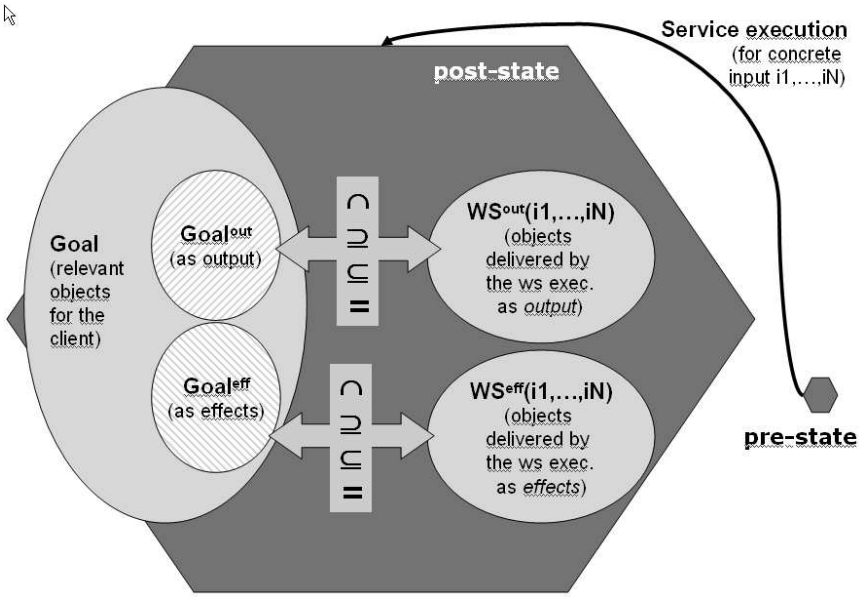



Figure 6: An extended set-based Service Model.

As the examples shows, there might be situations, where such relaxed notions of matching actually can be useful for a user. Thus, we will discuss them here as well.

In the following paragraphs we will show how to formalize the extended web service model and discuss how to extend the matching notions from Section 4.2 to cover service executions. For the sake of simplicity, we will only mention one set of objects in the descriptions of goals and services. Handling the set of outputs and effects separately can be achieved in the very same way.

**Formalizing the extended Service Model.** Instead of using an unary predicate $ws_{\mathcal{W}}(x)$ for describing the capability of a web service $\mathcal{W}$, we have to express the dependency of the objects delivered by a service on the concrete service execution and thus concrete input $i_1, \ldots, i_n$.

Let $\mathcal{W}$ be a web service with input parameters $i_1, \ldots, i_n$, then we formalize the capability of the service by a $n+1$-ary predicate $ws_{\mathcal{W}}(x, i_1 \ldots i_n)$ along with a set $\mathcal{W}$ of first-order sentences defining the semantics of $ws_{\mathcal{W}}(x, i_1 \ldots i_n)$. Here, the basic idea is that $ws_{\mathcal{W}}(x, i_1 \ldots i_n)$ can be understood as the following statement:

> The value $x$ will be delivered by the web service $\mathcal{W}$ if it is invoked with the input values $i_1 \ldots i_n$.

The set $\mathcal{W}$ defining $ws_{\mathcal{W}}(x, i_1 \ldots i_n)$ has to be chosen properly. In the simplest case, we achieve that with a single formula as follows:

$$\mathcal{W} : \forall x, i_1 \ldots i_n.(ws_{\mathcal{W}}(x, i_1 \ldots i_n) \leftrightarrow \psi^{pre}(i_1 \ldots i_n) \wedge \psi^{post}(i_1 \ldots i_n, x)) \quad (11)$$

where $\psi^{pre}(i_1 \ldots i_n)$ is an arbitrary first-order formula describing the precondition of the web service and $\psi^{post}(i_1 \ldots i_n, x)$ is an arbitrary first-order formula describing the postcondition. In $\psi^{post}(i_1 \ldots i_n, x)$ the variable $x$ refers to the output value(s) generated by the service execution.

Goals are just described in the very same way as in Section 4.2.

**Adapting the formal Matching Notions.** Since we adapted the way we describe web services, we have to adapt the formal criteria for our matching criteria as well. In the following we will show how to adapt the single notions ($\equiv_{\mathcal{O}}$, $\sqsubseteq_{\mathcal{O}}$, $\sqsupseteq_{\mathcal{O}}$, $\sqcap_{\mathcal{O}}$, $\|_{\mathcal{O}}$) accordingly and give a definition for the case in which we only consider *single executions* as well as the case of considering *multiple executions*.

This way, in principle we end up with (almost) 8 different notions of matching which potentially could be used by a client to specify his desire in a service request.

- **Exact-Match ($\mathcal{W} \equiv^1_{\mathcal{O}} \mathcal{G}$, $\mathcal{W} \equiv^+_{\mathcal{O}} \mathcal{G}$).**

  If we consider Exact-match under the assumption that the service is executed only once we have, we have to formalize the following statement: there are input values $i_1, \ldots, i_n$ such that the sets of objects $R_{\mathcal{W}}(i_1, \ldots, i_n)$ that the web service claims to deliver[16] when being invoked with input values $i_1, \ldots, i_n$ coincides with the set of objects which are relevant for the requester.

  Formally, that means that we have to prove the following :

  $$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \exists i_1, \ldots, i_n.(\forall x.(g_{\mathcal{G}}(x) \leftrightarrow ws_{\mathcal{W}}(x, i_1 \ldots i_n)) \quad (12)$$

  where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

  In this case we write $\mathcal{W} \equiv^1_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

  If we instead want to consider multiple executions we would have formalize the following statement:

  For each object $x$ in the universe it holds that $x$ can be delivered by service execution on some input values $i_1, \ldots, i_n$ iff $x$ is relevant for the client.

  Thus the proof obligation in this case is:

  $$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \forall x.(\exists i_1, \ldots, i_n.(g_{\mathcal{G}}(x) \leftrightarrow ws_{\mathcal{W}}(x, i_1 \ldots i_n)) \quad (13)$$

  where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

  In this case we write $\mathcal{W} \equiv^+_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

- **Subsumes-Match ($\mathcal{W} \sqsubseteq^1_{\mathcal{O}} \mathcal{G}$, $\mathcal{W} \sqsubseteq^+_{\mathcal{O}} \mathcal{G}$).**

  If we consider Subsumes-match under the assumption that the service is executed only once we have, we have to formalize the following statement: there are input values $i_1, \ldots, i_n$ such that the sets of objects $R_{\mathcal{W}}(i_1, \ldots, i_n)$ that the web service claims to deliver when being invoked with input values $i_1, \ldots, i_n$ is a subset of the set of objects which are relevant for the requester.

  Formally, that means that we have to prove the following :

  $$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \exists i_1, \ldots, i_n.(\forall x.(g_{\mathcal{G}}(x) \leftarrow ws_{\mathcal{W}}(x, i_1 \ldots i_n)) \quad (14)$$

  where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

  In this case we write $\mathcal{W} \sqsubseteq^1_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

[16] Of course, this depends on the respective intention used in the web service description.

If we instead want to consider multiple executions we would have formalize the following statement:

For each object $x$ in the universe it holds that if $x$ can be delivered by service execution on some input values $i_1, \ldots, i_n$ then $x$ is relevant for the client.

Thus the proof obligation in this case is:

$$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \forall x.(\exists i_1, \ldots, i_n.(g_{\mathcal{G}}(x) \leftarrow ws_{\mathcal{W}}(x, i_1 \ldots i_n)) \tag{15}$$

where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

In this case we write $\mathcal{W} \sqsubseteq^{+}_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

- **Plugin-Match ($\mathcal{W} \sqsupseteq^{1}_{\mathcal{O}} \mathcal{G}$, $\mathcal{W} \sqsupseteq^{+}_{\mathcal{O}} \mathcal{G}$).**

  If we consider Plugin-match under the assumption that the service is executed only once we have, we have to formalize the following statement: there are input values $i_1, \ldots, i_n$ such that the sets of objects $R_{\mathcal{W}}(i_1, \ldots, i_n)$ that the web service claims to deliver when being invoked with input values $i_1, \ldots, i_n$ is a superset of the set of objects which are relevant for the requester.

  Formally, that means that we have to prove the following :

$$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \exists i_1, \ldots, i_n.(\forall x.(g_{\mathcal{G}}(x) \rightarrow ws_{\mathcal{W}}(x, i_1 \ldots i_n)) \tag{16}$$

  where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

  In this case we write $\mathcal{W} \sqsupseteq^{1}_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

  If we instead want to consider multiple executions we would have formalize the following statement:

  For each object $x$ in the universe it holds that $x$ can be delivered by service execution on some input values $i_1, \ldots, i_n$ if $x$ is relevant for the client.

  Thus the proof obligation in this case is:

$$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \forall x.(\exists i_1, \ldots, i_n.(g_{\mathcal{G}}(x) \rightarrow ws_{\mathcal{W}}(x, i_1 \ldots i_n)) \tag{17}$$

  where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

  In this case we write $\mathcal{W} \sqsupseteq^{+}_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

- **Intersection-Match($\mathcal{W} \sqcap^{1}_{\mathcal{O}} \mathcal{G}$, $\mathcal{W} \sqcap^{+}_{\mathcal{O}} \mathcal{G}$).**

  If we consider Intersection-match under the assumption that the service is executed only once we have, we have to formalize the following statement: there are input values $i_1, \ldots, i_n$ such that the sets of objects $R_{\mathcal{W}}(i_1, \ldots, i_n)$ that the web service claims to deliver when being invoked with input values $i_1, \ldots, i_n$ has a common element with the set of objects which are relevant for the requester.

  Formally, that means that we have to prove the following :

$$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \exists i_1, \ldots, i_n.(\exists x.(g_{\mathcal{G}}(x) \wedge ws_{\mathcal{W}}(x, i_1 \ldots i_n)) \tag{18}$$

  where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

In this case we write $\mathcal{W} \sqcap^1_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

If we instead want to consider multiple executions we would have formalize the following statement:

There is an object $x$ in the universe such that $x$ that can be delivered by service execution on some input values $i_1, \ldots, i_n$ and $x$ is relevant for the client.

Thus the proof obligation in this case is:

$$\mathcal{W}, \mathcal{G}, \mathcal{O} \models \exists x.(\exists i_1, \ldots, i_n.(g_{\mathcal{G}}(x) \wedge ws_{\mathcal{W}}(x, i_1 \ldots i_n)) \tag{19}$$

where $\mathcal{W}$ is the definition of the web service, $\mathcal{G}$ is the definition of the goal, and $\mathcal{O}$ is a set of ontologies to which both descriptions refer.

In this case we write $\mathcal{W} \sqcap^+_{\mathcal{O}} \mathcal{G}$ to indicate this particular kind of match.

Obviously, the criteria (18) and (19) are logically equivalent and thus are the very same criteria. Thus, we do *not have to distinguish* between the two cases.

**Relation of the extended model to the simple one.** To conclude the discussion of the multiple notions we want to mention the following: The multiple execution notions actually check a set-theoretic relationship between the goal and the *union* of the sets of delivered objects *over all possible (valid) inputs*. Indeed, this can be considered as some sort of *abstraction from concrete executions of a web service* when checking a match and thus is very close to what we have discussed in Section 4.2. The main difference is there we do not consider explicit input parameters of a web service and do not refer to valid inputs only (i.e. refer explicitly to a precondition). Hence, we can consider the matching notions of this type as special cases of the ones that we discussed in Section 4.2.

### 4.3.2 Discovery with Transaction Logic

As discussed in previous sections, a more fine-grained matching requires a more precise description of what services a given web service offers access to, specifically of how the information and real-world effects described by the web service relate to the input provided to it.

In the following, we will denote goal postconditions and effects as $G_{post}$ and $G_{eff}$, respectively. Similarly, we will denote the assumptions, preconditions, postconditions and effects of the capability of a given web service $Serv$ by $C_{ass}(Serv, Input)$, $C_{pre}(Serv, Input)$, $C_{post}(Serv, Input)$, and $C_{eff}(Serv, Input)$. The preconditions, assumptions, postconditions, and effects predicates include a parameter $Input$, meaning that the preconditions and assumptions have to hold for the available input[17], and that the postconditions and effects will depend on the input given to the web service.

**Information available to web service discovery** We identify the following information available to the web service discovery process:

- Web services and goals are described in terms of a set of ontologies $O$, providing formal terminology which, as explained in Section 3, is amenable to automatic mediation. Such ontologies also provide the domain knowledge required to reason about the goal and web service descriptions.

[17] Assumptions might be dependent on the input, but not necessarily.

- Additionally to the domain knowledge available to the web service discovery process, the requester may provide some specific knowledge which he is willing to use as an input to a Web Service to achieve his goal e.g. his particular credit card information. This kind of knowledge cannot be expected in the general case to be part of the set of ontologies $O$, which will only provide general domain knowledge e.g. that Innsbruck is located in Austria. However, this knowledge specific to the requester might be required by the web service to provide the advertised results and, furthermore, it is required in the matching process to check whether the web service has the necessary input information available to produce its results. We denote this requester specific knowledge by $I$.

- Some information might be also provided in the description of the goal itself e.g. if the goal requests a train ticket from Innsbruck to Frankfurt, this information can be used to prove our proof obligation for web service discovery. We denote such information by $In(G)$.

**Proof obligations** If the web service models the relation between its input and its results i.e. if the postconditions and effects delivered by the execution of the web service are dependent on the input information made available by the requester to the web service, we can check whether a given input information will lead to outputs and effects that satisfy the postconditions and effects requested in the goal.

For this purpose, we first need to check that there exists input information $Input$ available to the web service i.e. contained in $I$ or $In(G)$[18] that satisfies the preconditions $(C_{pre}(Serv, Input))$ of a potentially matching web service $Serv$. The assumptions $(C_{pre}(Serv, Input))$ of $Serv$ are assumed to hold and, therefore, not checked.

If the above holds, we can hypothetically assume that the postconditions and effects of the web service for this input ($C_{post}(Serv, Input)$ and $C_{eff}(Serv, Input)$ hold. Under this assumption, we have the effects and postconditions of the candidate web service available; if the goal postconditions and effects hold in this state, it means that the web service provides the desired results for the available input and, therefore, $Serv$ is a match (for $Input$).

This can be formalized by the following proof obligation using transaction logic [BK98], where $\diamond$ is the hypothetical operator, and $\otimes$ is the sequence operator:

$$\begin{aligned} O, I, In(G) \models \exists Serv, \exists_{I,In(G)} Input \\ \diamond(C_{pre}(Serv, Input) \otimes insert\{C_{post}(Serv, Input), C_{eff}(Serv, Input)\} \\ \otimes\ G_{post} \wedge G_{eff}) \end{aligned} \quad (20)$$

In the formula above, $\exists_{In(G)} Input$ means, as explained before, that only information in $I$ or $In(G)$ will be considered as possible input.

Since we use the hypothetical operator ($\diamond$), the assertion of the web service postconditions and effects will be rolled back i.e. retracted from the knowledge base after the checking is finished.

The use of the sequence operator ($\otimes$) means that the preconditions must be tested before the postconditions and effects of the web service can be asserted, and that the goal will only be tested after this. In this way, we distinguish between the pre and post-state in the execution of the web service. This is

[18] Notice that the domain knowledge contained in $O$, although necessary for proving a match, is not necessarily input information explicitly made available by the requester to achieve his goal.

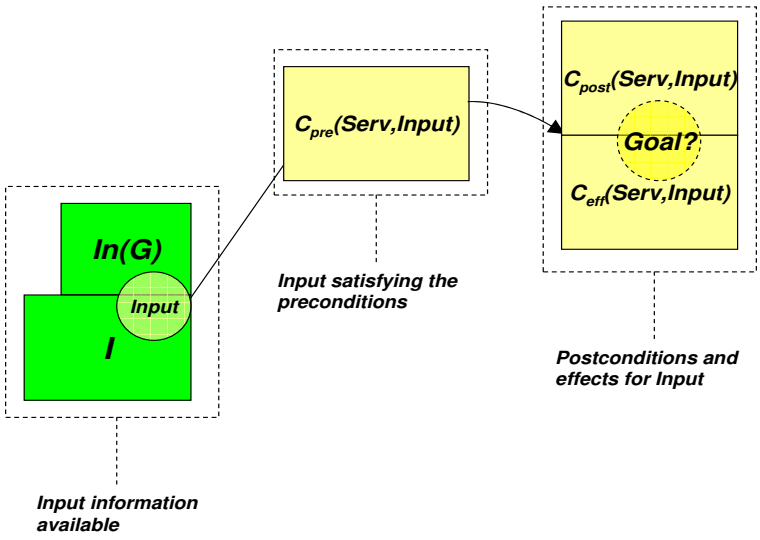



Figure 7: Hypothetically assuming web service postconditions and effects

especially relevant for checking the preconditions, as the assertion of the web service results will change the pre state[19] (both of the information and real-world spaces) and, by definition, the preconditions must be checked before such changes happen. Similarly, the goal must be tested only in the post-state i.e. when the postconditions and effects of the web service have been asserted.

The proof obligation is illustrated in Figure 7. On the left hand side of the picture, we represent the input information available. From this, $Input$ satisfying the web service capability preconditions $C_{pre}(Serv, Input)$ is considered. Depending on this input, we hypothetically assume (represented by an arrow in the figure) the postconditions and effects of the web service. These, represented by the box on the right hand side of the figure, are the results provided by the web service for the input considered. Finally, we check whether the goal is satisfied by these results i.e. if they contain the results requested in the goal.

*Notion of match.* For the description of the goal, we allow the use of both universally quantified goals i.e. all the results satisfying the conditions stated in the goal have to be delivered by the web service, as well as existentially quantified goals i.e. some results satisfying such conditions have to be delivered. Whether the request is existential or universal will be part of the description of the goal itself. If existential quantification is used, web services providing at least one result satisfying the conditions stated in the goal will be matched, while only web services providing all such results will be matched if universal quantification is used. In any case, the proof obligation (20) will provide a guarantee[20] that the selected services can fulfill the goal with the available input.

*Single execution Vs multiple execution.* The proof obligation (20) states that there exists an input ($Input$) that fulfills the web service preconditions and that generate results that satisfy the goal. This means that the web service has to fulfill the goal with a unique tuple of inputs i.e. with a single execution.

[19] This is true for every web service which really provides a service. Otherwise, the web service does not have any interest for the requester and it should not even be considered.

[20] This guarantee is relative to the degree of accuracy in the description of the web service functionality. In the general case, as discussed in section 2, service discovery will be required for a complete guarantee.

If we want to allow a simple form of composition, matching services that can achieve the goal through the use of a set of inputs i.e. a set of invocations, we have to modify our previous proof obligation, which leads to proof obligation (21):

$$\begin{aligned} O, I, In(G) \models \exists Serv \\ \Diamond(\forall Input \\ insert\{C_{post}(Serv, Input), C_{eff}(Serv, Input)|_{I,In(G)} C_{pre}(Serv, Input)\} \\ \otimes\ G_{post} \wedge G_{eff}) \end{aligned} \tag{21}$$

$insert\{C_{post}(Serv, Input), C_{eff}(Serv, Input)|_{I,In(G)} C_{pre}(Serv, Input)\}$ corresponds in the previous formula to a *bulk update* i.e. the assertion of the postconditions and effects will occur for every Input in $I$ or $In(G)$ satisfying the conditions in $C_{pre}(Serv, Input)$. Therefore, we hypothetically assume the results of the service for every possible input that satisfies the web service preconditions, and we test the goal with all such results hypothetically asserted i.e. we test the goal over the union of these results. This allows to match web services that require multiple executions to achieve the goal.

*Considering domain knowledge as available input.* We assumed above that the input available to the web service to provide its results is restricted to the information explicitly provided by the requester i.e. we restricted the input to $I$ and $In(G)$. However, in some cases it might be of interest to also consider as possible input the instance data contained in the domain knowledge provided by the set of ontologies $O$ e.g. that Innsbruck is an instance of *city* and it is *located in* Austria, which is an instance of *country*. In that case, there must exist $Input$ available in $I$, $In(G)$, or $O$ that produces the requested results and, therefore, $\exists_{In(G)} Input$ must be replaced by $\exists Input$ in proof obligation (20), and $|_{I,In(G)}$ by $|$ in proof obligation (21).

As an example, consider as a goal getting the price for a museum pass for Innsbruck. Let us assume that there is a service offering museum pass prices for any city in Austria, and that the web service acting as an interface to the service requires as an input the country for what the information is requested and returns *museum pass prices for all the cities in Austria that offer a museum pass*. This web service, thus, provides the information wanted by the requester, but it requires as an input Austria, not Innsbruck. A match can be proved if we consider that information in $O$ can be used as input to the web service i.e. if we consider that the Austria instance as a possible input to the web service.

Either if we include the information in $O$ as available input or not, the requester will get as the result of the web service discovery process the set of matched web services together with the input necessary to achieve its request for each of such web services. As the results of a web service are dependent, in the general case, on the input provided to it, a complete match result must include not only matched web services but also **for what input the requested results will really be provided by the web service**. These inputs are the ones to be used in service discovery for executing the web service.

*Assumptions.* In proof obligations (20) and (21) we have assumed that the web service assumptions hold. In the following, we revise the previous proof obligations to include the checking of the assumptions of a web service $Serv$ in order to match it. Notice that whether the assumptions are to be assumed or to be checked will depend on the use case and the level of guarantee the requester requires for a match. For checking the assumptions, we have to introduce new

knowledge that might be available to the web service discovery process, $W$, about the current state of the world not covered by neither $I$, $In(G)$, nor $O$[21]. The extended proof obligations for (20, 21) are (22, 23):

$$O, I, In(G), W \models \exists Serv, \exists_{I,In(G)} Input\ C_{ass}(Serv, Input) \wedge \\ \Diamond(C_{pre}(Serv, Input) \otimes insert\{C_{post}(Serv, Input), C_{eff}(Serv, Input)\} \\ \otimes G_{post} \wedge G_{eff}) \tag{22}$$

$$O, I, In(G), W \models \exists Serv \\ \Diamond(insert\{C_{post}(Serv, Input), C_{eff}(Serv, Input)|_{I,In(G)} \\ C_{pre}(Serv, Input) \wedge C_{ass}(Serv, Input)\} \\ \otimes G_{post} \wedge G_{eff}) \tag{23}$$

**Relation to other approaches** The approach presented in this section corresponds to an extension of set based modelling from Section 4.2 to include the relation between the web service results and its input. Roughly speaking, $C_{pre}(Serv, Input)$, $C_{post}(Serv, Input)$ and $C_{eff}(Serv, Input)$ correspond to the parameterization of the sets of results provided by the web service with the input given.

Regarding the approach presented in section 4.3.1, the use of transaction logic introduces the main advantage of explicitly considering the state in the proof obligations for web service discovery. As discussed before, the preconditions of the web service must be checked in the pre-state i.e. before the postconditions and effects of the web service are generated. Not considering the state in the discovery process might lead to problems in some cases e.g. if the web service requires the balance of the requester's account to be higher than a given amount and one of its effects is to decrease the balance of such account. If preconditions and effects are checked in the same state, some expected matches might fail.

The notions of match identified in previous approaches are covered by the different proof obligations introduced in this section. Although partial matches i.e. $\mathcal{W} \sqsubseteq^{\forall}_{\mathcal{O}} \mathcal{G}$ and $\mathcal{W} \sqcap^{\forall}_{\mathcal{O}} \mathcal{G}$ are not directly covered, they can be relaxed to $\mathcal{W} \sqsubseteq^{\exists}_{\mathcal{O}} \mathcal{G}$ and $\mathcal{W} \sqcap^{\exists}_{\mathcal{O}} \mathcal{G}$, respectively. In fact, partially matching in the first two cases is equivalent to relaxing the intention of the requester from universal intention to existential intention. Therefore, we can see that this approach covers all the cases discussed in previous sections.

[21] Notice that in general the ontologies will only include stateless knowledge.

# 5 Conclusions

In this version of the deliverable we have defined a conceptual model for service discovery which avoids unrealistic assumptions and is suitable for a wide range of applications. One of the key features of this model is the explicit distinction between the notions of web services and services. Moreover, the model does not neglect one of the core problems one has to face in order to make discovery work in a real-world setting, namely the heterogeneity of descriptions of requestors and providers and the required mediation between heterogeneous representations. As discussed in Section 3, the discussed approaches are based on using terminologies, controlled vocabularies or rich descriptions which are based on ontologies. For each of them, a working solution to the mediation problem is possible or not unrealistic.

We have outlined various approaches on discovery of web services with different requirements on the description of web services and the discovery request itself. Our main focus has been on semantic-based approaches to web service discovery.

Clearly, this discussion and the comparison of the single approaches will be further elaborated in the next version of the document.

# 6 Acknowledgements

The work is funded by the European Commission under the projects DIP, Knowledge Web, Ontoweb, SEKT, SWWS, Esperonto, and h-TechSight; by Science Foundation Ireland under the DERI-Lion project; and by the Vienna city government under the CoOperate programme.

The editors would like to thank to all the members of the WSMO working group for their advice and input into this document.

# References

[ACKM03] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services*. Springer, 2003.

[AGDR03] Rama Akkiraju, Richard Goodwin, Prashant Doshi, and Sascha Roeder. A method for semantically enhancing the service discovery capabilities of UDDI. In Subbarao Kambhampati and Craig A. Knoblock, editors, *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*, pages 87–92, 2003.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[BHRT03] B. Benatallah, M-S. Hacid, C. Rey, and F. Toumani. Request rewriting-based web service discovery. In *The Semantic Web - ISWC 2003*, pages 242–257, October 2003.

[BK98] A.J. Bonner and M. Kifer. A logic for programming database transactions. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, chapter 5, pages 117–166. Kluwer Academic Publishers, March 1998.

[BPM+98] V. R. Benjamins, E. Plaza, E. Motta, D. Fensel, R. Studer, B. Wielinga, G. Schreiber, Z. Zdrahal, and S. Decker. Ibrow3: An intelligent brokering service for knowledge-component reuse on the world-wide web. In *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW98)*, Banff, Canada, 1998.

[Cla85] William J. Clancey. Heuristic classification. *Artificial Intelligence*, 27(3):289–350, 1985.

[dB04] Jos de Bruijn. The WSML Family of Representation Languages. Working draft, Digital Enterprise Research Insitute (DERI), October 2004. Look at `http://www.wsmo.org/2004/d16/`.

[DKO+02] Ying Ding, M. Korotkiy, Boris Omelayenko, V. Kartseva, V. Zykov, Michael Klein, Ellen Schulten, and Dieter Fensel. Goldenbullet: Automated classification of product data in e-commerce. In *Proceedings of Business Information Systems Conference (BIS 2002)*, Poznan, Poland, 2002.

[dMRME04] Jos de Bruijn, Francisco Martin-Recuerda, Dimitar Manov, and Marc Ehrig. State-of-the-art survey on ontology merging and

aligning v1. Technical report, SEKT project IST-2003-506826, 2004.

[FB02] D. Fensel and C. Bussler. The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[Fel98] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, May 1998.

[Fen03] D. Fensel. *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce, 2nd edition*. Springer-Verlag, Berlin, 2003.

[GCTB01] J. Gonzlez-Castillo, D. Trastour, and C. Bartolini. Description logics for matchmaking of services. In *KI-2001 Workshop on Applications of Description Logics*, September 2001.

[GHVD03] B.N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proc. Intl. Conf. on the World Wide Web (WWW-2003)*, Budapest, Hungary, 2003.

[GMP04] Stephan Grimm, Boris Motik, and Chris Preist. Variance in e-business service discovery. *Semantic Web Services: Preparing to Meet the World of Business Applications, Workshop at ISWC 2004*, November 2004.

[Gru93] T. R. Gruber. A translation approach to portable ontology specification. *Knowledge Acquisition*, 5(2):199–220, 1993.

[HST00] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[KLP+04] Michael Kifer, Ruben Lara, Axel Polleres, Chang Zhao, Uwe Keller, Holger Lausen, and Dieter Fensel. A logical framework for web service discovery. In *ISWC 2004 Workshop on Semantic Web Services: Preparing to Meet the World of Business Applications*, 2004.

[LH03a] Lei Li and Ian 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, May 2003.

[LH03b] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology, 2003.

[LRK04] Holger Lausen, Dumitru Roman, and Uwe Keller (*editors*). Web service modeling ontology - standard (WSMO-Standard), version 1.0. Working draft, Digital Enterprise Research Institute (DERI), 2004. `http://www.wsmo.org/2004/d2/v1.0/`.

[OF01] Boris Omelayenko and Dieter Fensel. An analysis of integration problems of xml-based catalogs for b2b electronic commerce. In *Proceedings of the 9th IFIP 2.6 Working Conference on Database Semantics (DS-9)*, pages 232–246, April 2001.

[PKPS02a] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In I. Horrocks and J. Handler, editors, *1st Int. Semantic Web Conference (ISWC)*, pages 333–347. Springer Verlag, 2002.

[PKPS02b] M. Paolucci, T. Kawamura, T.R. Payne, and K. Sycara. Semantic matching of web services capabilities. In *Proceeding of The First International Semantic Web Conference (ISWC2002)*, Sardinia, Italy, 2002.

[PKPS02c] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara. Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 333–347. Springer-Verlag, 2002.

[Pre04] Chris Preist. A conceptual architecture for semantic web services. In *Proceedings of the International Semantic Web Conference 2004 (ISWC 2004)*, November 2004.

[RLK04] Dumitru Roman, Holger Lausen, and Uwe Keller (*editors*). Web Service Modeling Ontology (WSMO). Working draft, Digital Enterprise Research Insitute (DERI), September 2004. Available from `http://www.wsmo.org/2004/d2/v1.0/`.

[RS95] Ray Richardson and Alan F. Smeaton. Using WordNet in a knowledge-based approach to information retrieval. Technical Report CA-0395, Dublin, Ireland, 1995.

[SWKL02] K. Sycara, S. Widoff, M. Klusch, and J. Lu. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-Agent Systems*, pages 173–203, 2002.

[The04] The OWL Services Coalition. OWL-S 1.1 beta release. Available at http://www.daml.org/services/owl-s/1.1B/, July 2004.

[VSSP04] K. Verma, K. Sivashanmugam, A. Sheth, and A. Patil. Meteor-s wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services. *Journal of Information Technology and Management*, 2004.

[ZK04] O.K. Zein and Y. Kermarrec. An approach for describing/discovering services and for adapting them to the needs of users in distributed systems. In *AAAI Spring Symposium on Semantic Web Services*, March 2004.

[ZW97] Amy Moormann Zaremski and Jeannette M. Wing. Specification matching of software components. *ACM Trans. Softw. Eng. Methodol.*, 6(4):333–369, 1997.