Semantic Web Service Discovery in the WSMO Framework

Uwe Keller\(^*\)\(^1\), Rubén Lara\(^2\), Holger Lausen\(^3\) and Dieter Fensel\(^4\)

\(^*\) Corresponding author

\(^1\) Digital Enterprise Research Institute (DERI), Leopold-Franzens-Universität Innsbruck, Austria
Technikerstrasse 21a, ICT Building, A-6020 Innsbruck, Austria
phone: +43 512 507 6468
fax: +43 512 507 9872
eMail: uwe.keller@deri.org

\(^2\) Grupo Analistas, Tecnología, Información y Finanzas, Madrid, Spain
Españalet, 19 - 28010 Madrid
phone: +34 91 520.04.05
fax: +34 91 520.01.67
eMail: rlara@afi.es

\(^3\) Digital Enterprise Research Institute (DERI), Leopold-Franzens-Universität Innsbruck, Austria
Technikerstrasse 21a, ICT Building, A-6020 Innsbruck, Austria
phone: +43 512 507 6464
fax: +43 512 507 9872
eMail: holger.lausen@deri.org

\(^4\) Digital Enterprise Research Institute (DERI), Leopold-Franzens-Universität Innsbruck, Austria
Technikerstrasse 21a, ICT Building, A-6020 Innsbruck, Austria
phone: +43 512 507 6485
fax: +43 512 507 9872
eMail: dieter.fensel@deri.org

Chapterdraft (4th April 2006) for
Semantic Web Services: Theory, Tools and Applications
SEMANTIC WEB SERVICE DISCOVERY
IN THE WSMO FRAMEWORK

Abstract

The Web Service Modeling Ontology (WSMO) provides a conceptual framework for semantically describing Web services and their specific properties. In this chapter we discuss how WSMO can be applied for service discovery. We provide a proper conceptual grounding by strictly distinguishing between service and Web service discovery and then present different techniques for realizing 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.
# Contents

1 Introduction ............................................. 4

2 Basics ................................................. 6
   2.1 The Web Service Modeling Ontology .................. 6
      2.1.1 Design Principles and Approach .................. 6
      2.1.2 Top Level Elements .............................. 7
   2.2 Web Service vs. Service .............................. 9
   2.3 Web Services at various Levels of Abstraction ........ 12
   2.4 Goal of the WSMO Discovery Framework ............... 17

3 A Conceptual Model for Service Discovery .................. 18
   3.1 Heuristic Classification .............................. 18
   3.2 Service Discovery .................................... 19
   3.3 Relation between Discovery and Mediation ............ 22

4 Web Service Discovery .................................... 23
   4.1 Keyword-based Discovery ............................... 23
   4.2 Discovery based on simple semantic Descriptions ..... 23
   4.3 Discovery based on rich semantic Descriptions ........ 34

5 Related Work ............................................ 39

6 Conclusions ............................................. 40
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. [Fensel, 2003]).
- 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. [Alonso et al., 2003]). 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. [Fensel and Bussler, 2002]). OWL-S [The OWL Services Coalition, 2004] and WSMO [Roman et al., 2005] 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. [Akkiraju et al., 2003], [Li and Horrocks, 2003], [Paolucci et al., 2002].
[Benatallah et al., 2003], [Gonzlez-Castillo et al., 2001], [Verma et al., 2004], [Sycara et al., 2002], [Zein and Kermarrec, 2004]). 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 semantic-based discovery of web services. Stepwise we will enrich the scenario we are able to support. In this chapter, we focus on the essential principles underlying semantic-based discovery based on a simple formal model, that is independent of any particular knowledge representation language. For a discussion of an implementation of these principles in particular logics, we refer to [Keller et al., 2004].

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.

**Overview of the Chapter.** This chapter is organized as follows: Section 2 provides some the necessary background on the Web Service Modelling Ontology (WSMO) Framework and the relevant notions for discovery therein. Section 3 identifies three major conceptual phases in service discovery. Section 4 discusses in detail the issues around semantic-based discovery of web services. Finally, Section 6 gives some conclusions about the WSMO Discovery Framework at the end of the chapter.
2 Basics

In the following, we briefly overview the Web Service Modeling Ontology (WSMO) [Roman et al., 2005] and discuss the notions that are relevant for discovery. In particular, WSMO distinguishes the notions of Web Service and Service in the context of discovery. Furthermore, we summarize what WSMO eventually aims at in regard of Web Service Discovery.

2.1 The Web Service Modeling Ontology

Taking the Web Service Modeling Framework (WSMF) as its conceptual basis [Fensel and Bussler, 2002], the WSMO project is an ongoing research and development initiative\(^1\) for defining a capacious framework for Semantic Web services.

2.1.1 Design Principles and Approach

Semantic Web services aim at realizing the vision of the Semantic Web. Therefore, WSMO is based on the following design principles that integrate Web design principles, Semantic Web design principles, as well as design principles for distributed, service-oriented computing for the Web.

**Web Compliance:** WSMO inherits the concept of IRIs (Internationalized Resource Identifier) for unique identification of resources as the essential design principle of the Web. Moreover, WSMO adopts the concept of Namespaces for denoting consistent information spaces, and supports XML as well as other W3C Web technology recommendations.

**Ontology-Based:** Ontologies are used as the data model throughout WSMO, meaning that all resource descriptions as well as all data interchanged during service usage are based on ontologies. Following the idea of the Semantic Web, this allows semantically enhanced information processing as well as support for semantic interoperability.

**Goal-driven Architecture:** User requests are formulated as goals independently of available Web services. Thereby, the underlying epistemology of WSMO differentiates between the desires of clients and available Web services.

**Strict Decoupling:** Each WSMO resource is specified independently, without regard to possible usage or interactions with other resources. This complies with the open and distributed nature of the Web.

**Centrality of Mediation:** Mediation addresses the handling of heterogeneities that naturally arise in open environments like the Web. As a complementary design principle to strict decoupling, WSMO recognizes the importance of mediation for the successful deployment of Web services by making mediation a first class component of the framework.

**Description versus Implementation:** WSMO differentiates between the description and the implementation of Web services. The former denotes the unambiguous description of Web services that is needed for automated usage of Web services; the latter is concerned with the internal implementation of the Web Service which is not of interest for Semantic Web service technologies.

---

\(^1\)WSMO is a working group of the SDK-Cluster, a joint initiative of European research and development efforts around the Semantic Web and Web services (www.sdk-cluster.org). All specifications and related information are available at the WSMO homepage: http://www.wsmo.org.
The design principles are reflected in the four WSMO top level elements shown in Figure 1. Ontologies provide the formal terminology definitions that are used as the data model throughout WSMO; Goals are formal specifications of objectives that a client aims to achieve by using Web services, realizing a goal-driven approach that ontologically decouples requesters and providers; WSMO Web Services are formal descriptions needed for automated service handling and usage, whereby the internal implementation of a Web service is not of interest; finally, Mediators are the top level element for handling heterogeneity.

![Diagram of WSMO Top Level Elements]

While referring to the WSMO specification for detailed definitions [Roman et al., 2005], the following explains the WSMO elements with regard to their purpose and constitutive description elements.

### 2.1.2 Top Level Elements

**Ontologies**  
In compliance to the vision of the Semantic Web, WSMO uses ontologies as the underlying data model for Semantic Web services. This means that all resource descriptions and all information interchanged during collaboration execution is based on ontologies, thereby providing the basis for semantically enhanced information processing and ensuring semantic interoperability between Semantic Web services.

In accordance to the AI-theory of ontologies [Staab and Studer, 2004], WSMO ontologies consists of the following elements: **Concepts** describe the entities of a domain that are characterized by **Attributes**; **Relations** describe associations between concepts, whereby subsumption and membership relationships define the taxonomic structure of an ontology. An **Instance** is a concrete individual of a concept, and **Axioms** define constraints and complex aspects of the domain in terms of logical expressions. Regarding engineering methodologies developed for the Semantic Web [Fensel, 2003], ontology design in WSMO demands and supports **modularization**, i.e. small-sized and concise ontologies, **decoupling**, i.e. distributed and multi-party ontology development, and **ontology mediation** for resolving possibly occurring mismatches between loosely coupled ontologies for a specific usage scenario.

**Web Services**  
WSMO defines a description model that encompasses those information needed for automatically determining the usability of a Web service. As shown in Figure 2, a WSMO Web service description is comprised of four elements: (1) **non-functional properties**, (2) a **capability** as the functional description of the service; summarized as service interfaces, (3) a **choreography** that describes the interface for service consumption by a client (i.e. how to interact with the Web Service), and (4) an **orchestration** that describes how
the functionality of the service is achieved by aggregating other Web services. These notions describe the functionality and behavior of a Web service, while its internal implementation is not of interest.

While the non-functional properties contain descriptions like quality of service, financial information, and versioning information, the functional service description elements are the Capability and the Service Interfaces. The former describes the functionality of a Web service from a black box perspective for supporting automated functional discovery, meaning to determine whether a Web service can be used to satisfy a user request on basis of its capability. The Service Interfaces describe the interaction behavior of the Web service for consuming, respectively achieving its functionality: a client that wants to utilize the Web service needs to comply with its Choreography Interface; similar, a Web service that realizes it functionality by aggregating other Web services in its Orchestration - which is a main objective of Web service technology - needs to consume these via their respective Choreography Interfaces.

As outlined above, WSMO differentiates two Service Interfaces that are concerned with the interaction behavior of the Web service. The Choreography Interface describes the behavior of the Web Service for consuming its functionality in terms of the information interchange expected, and the Orchestration describes how the Web service interacts with other Web services in order to achieve its functionality.

In contrast to several existing Web service technologies that focus on describing the interaction execution in detail - like WSDL, BPEL4WS, and WS-CDL (see [Barros et al., 2005] for an extensive discussion) - the aim of WSMO Service Interface descriptions is to provide the foundation for determining automatically whether the interactions between a Web service, its clients, and other aggregated Web services that need to be performed for achieving its functionality. Therefore, WSMO defines a formal model for Service Interface descriptions that supports ontologies as the underlying data model as is based on the Abstract State Machine (short: ASM) framework, a high-level, abstract technique for validating complex systems or programs and provide a highly expressive, flexible, and formally sound means for representing dynamics [Boerger and Staerk, 2003].

In the remainder of the book we will focus on the functional description contained in the capability and will not further consider behavioral aspects due to space limitations.
Goals  In order to facilitate automated Web service usage and support ontological separation of user desires, service usage requests, and Web service descriptions, Goals in WSMO allow specifying objectives that clients - which can be humans or machines - wish to achieve.

The general structure of WSMO Goal descriptions is similar to Web service descriptions. The client can specify the functionality expected in a requested capability. Also, a Goal can carry information on the expected behavior of an acceptable Web service in so-called requested interfaces that can define the excepted communication behavior for consuming a Web service with respect to its Choreography Interface as well as restrictions on other Web services aggregated in the orchestration of an acceptable Web service (e.g. only Web services are accepted that utilize a trusted payment facility). It is important to remark that Goal descriptions are defined from the client perspective, thereby decoupled from Web service descriptions.

Mediators  Mediation is concerned with handling heterogeneity, i.e. resolving possibly occurring mismatches between resources that ought to be interoperable. Heterogeneity naturally arises in open and distributed environments, and thus in the application areas of Semantic Web services. Hence, WSMO defines the concept of Mediators as a top level notion.

Mediator-orientated architectures as introduced in [Wiederhold, 1994] specify a mediator as an entity for establishing interoperability of resources that are not compatible a priori by resolving mismatches between them at runtime. The aspired approach for mediation relies on declarative description of resources whereupon mechanisms for resolving mismatches work on a structural, semantic level, in order to allow generic, domain independent mediation facilities as well as reuse of mediators. Concerning the needs for mediation within Semantic Web services, WSMO distinguishes three levels of mediation:

1. **Data Level Mediation** - mediation between heterogeneous data sources; within ontology-based frameworks like WSMO, this is mainly concerned with ontology integration.

2. **Protocol Level Mediation** - mediation between heterogeneous communication protocols; in WSMO, this mainly relates to choreographies of Web services that ought to interact.

3. **Process Level Mediation** - mediation between heterogeneous business processes; this is concerned with mismatch handling on the business logic level of Web services (related to the orchestration of Web services).

2.2  Web Service vs. Service

A workable approach to automating service discovery must precisely define its conceptual model and the particular assumptions underlying the proposed solution. For this purpose, we start by providing a common understanding of what a service is and the levels of abstraction in its description based on [Preist, 2004], as well as our assumptions on the elements involved in the process of locating suitable services for service requesters.
What is a Service? It has been pointed out in [Preist, 2004] that the notion of service is semantically overloaded. Several communities have different interpretations which makes it difficult to understand and relate single approaches and exchange ideas and results. In order to reach a common understanding of the problem we address here, we need to precisely define the term service and, therefore, what kind of entities we aim at locating in principle.

In this chapter, we use the following interpretation for the term service, as described in the conceptual architecture for semantic Web services presented in [Preist, 2004]: Service as provision of value in some domain. This definition regards a service as a provision of value (not necessarily monetary value) in some given domain, independently of how the supplier and the provider interact. Examples of services in this sense are the provision of information about flight tickets or the booking of a trip with certain characteristics by a tourism service provider.

Abstract Services & Web Services. Usually, a service provider $P$ does not only provide one particular service $S$, but a set of coherent and logically related services. For instance, a hotel usually does not only provide the possibility to book a particular room at a particular date for a given number of nights, but instead it will offer the general service of booking rooms. Thus, a provider will be interested in advertising all the services it is able to provide, i.e. a set $A_P$ of services. Following the terminology from [Preist, 2004], we call this collection of services an abstract service offered by a provider. The smallest unit of advertisement is considered to be an abstract service.

In order to deliver a service, a service provider $P$ usually needs certain information from the requester. For instance, a hotel might require the name of the person booking the room, the requested room features, and a valid credit card number as input information in order to book a room. This input data $i_1, \ldots, i_n$ will determine what concrete service [Preist, 2004] $S \in A_P$ has to be provided by $P$.

Finally, Web Services are computational entities using standardized interfaces that allow clients to interact with a provider $P$ to access, explore and consume concrete services $S \in A_P$. A service requester technically needs to interact with the Web Service to get what he actually aims for: a concrete service or more generally, a set of concrete services that fulfill the requesters goal, i.e. an abstract service. Hence, we will not distinguish between Web Services and Abstract Services of a provider throughout this document and treat both notions as synonyms. Moreover, we can observe that a concrete service $S \in A_P$ is being delivered in the course of or as a result of a particular execution of the Web Service by invocation with specific data $i_1, \ldots, i_n$ for the required input parameters.

Dynamics of Abstract Services over time. In the real-world, we have a further complication of matters in regard of discovery and Web Service description: in general, an abstract service $A_P$ offered by some provider $P$ does not stay the same, but changes over time. For instance, a hotel will not be able to book a room with a single bed on a specific date if all such rooms in the hotel are already booked on this date.

Since clients are basically interested in finding abstract services which actually can solve their problem at hand (as specified in a WSMO goal), discovery
in general needs to take into account this dynamics in order to create accurate results. This basically means, that purely static descriptions of Web services are not sufficient in the general case. In applications where highly accurate results are requested, Web Service descriptions will have to consist of a dynamic component as well.

On a description level there are various options to achieve the proper representation of dynamically changing abstract services of a provider: (1) Use a purely static description of the abstract service and change the description in its entirety every time the abstract service changes or (2) use a static description where some parts refer to a well-defined resource that reflects that currently valid information of the dynamic part (for instance a database table with all available rooms of a hotel at a certain point in time). Clearly, the first approach is not really scalable since constant changes of a stored abstract service description are needed, whereas the second approach is a viable one, as we can see with dynamically generated Web pages of online stores like Amazon.com that give an up-to-date perspective on prices of single books. In the latter case, we simply externalize the details of the changing aspects of an abstract service and provide a reference in the remaining part of the Web service description. Hence, the abstract service description including the reference does not need to change over time when the abstract service of the provider changes and Web service descriptions get more compact.

Nonetheless, the latter approach requires communication with the provider of an abstract service (in some way, for instance via a Web service that accesses a certain database table) and thus can potentially be a costly step in terms of computation time. Hence, there is a clear trade-off between accuracy of discovery results and efficiency of the discovery process, that has to be considered for any specific application that leverages Web Service discovery.

**Knowledge Representation Languages in the Framework.** Before we explain in more detail the model of our framework, we want to stress that the framework, as it is presented in the following is language-independent in the first place and stresses the essential aspects of Web services, matchmaking and Web service discovery on the basis of a mathematical model. The choice of a particular knowledge representation language for the model is in no way essential for the WSMO Discovery Framework; however, when giving the examples of Web service descriptions for the various levels of our model in the following, we use a specific description (or knowledge representation language) for illustration purposes: the WSML language\(^2\)[de Bruijn, 2005].

In general we want to note the following observation, since it embodies distinct feature as well as a clear and essential contribution of the WSMO working group to the Semantic Web Service community in regard of semantic Web service description and discovery: traditionally\(^3\), people in various communities, especially the Semantic Web community as well as the Description Logic community had a heavily language-centered perspective when studying the semantic description of Web services as well as matchmaking. Because of this over-

\(^2\)WSML (Web Service Modelling Language) is a formal language that implements the WSMO ontology in a particular representation language with formal semantics. In particular, WSML can be used to describe Ontologies and supports various widely-used knowledge representation principles in a single family of languages: Description Logic-based, Rule-based as well First-order-based Modelling.

\(^3\)i.e. before the appearance of WSMO Working Group
emphasized (and rather narrow) language focus the community unfortunately overlooked until recently the most essential aspect of any proper description framework: A clear understanding of what the object actually is, that one needs to describe (i.e. Web service). In a sense, a fundamental principles underlying formal logics have been ignored, namely that a language is tailored towards the domain its applied in and that the semantics of a description language usually is defined in model-theoretic terms. The latter requires that one needs some proper mathematical model of Web Services and enables a clear understanding of the matter. Subsequently, matching of Web service advertisements and requests (and thus service discovery) can then be considered in a proper way with respect to this mathematical model in a completely language independent manner. One can focus on the essential aspects rather than a concrete language (which only is of secondary importance) and get a far better understanding of the overall matters.

During the work on the WSMO Discovery Framework this prevalent misconception has been understood and properly addressed (see in particular [Lausen, 2005; Keller et al., 2004]): discovery in the first place is not a matter of using a specific language, but understanding a mathematical model and developing matching based on the model. Later, on can address the problem of deriving a suitable formalization in a concrete language. Eventually, there might be many different languages that one can use for this purpose. That is one of the definite strengths of the WSMO Discovery Framework. In the following, we use therefore the universal framework of set theory to describe and discuss Web services as well as Web Service Discovery.

2.3 Web Services at various Levels of Abstraction
The descriptions of Web services as published by service providers and the goals of service requesters need to be compared to each other during the discovery process in order to identify matching Web services and goals. In essence, matching in the discovery process is about finding common elements of these descriptions.

In the model that we discussed above, an abstract service $A$ of a provider essentially is a set of elements that we call services. Services themselves are abstract entities in this model. Depending on how detailed we consider these entities, we end up with a model of Web services on different levels of abstraction that is depicted in Fig. 3.

On the most fine-grained level, we can consider services as concrete state-transitions $(s, s')$ from a state $s$ to another state $s'$ in the world. The notion of state here refers to an abstract object that describes all the relevant properties of the world at some specific moment in time. The states $s$ and $s'$ precisely determine, how the world looks like before and after the service has been provided. One a more abstract level, we can ignore the detailed structure of services (i.e the respective states, their interdependency, and the respective state-transition)

\footnote{For instance, the almost religious use of OWL ontology language (as well as underlying Description Logics) despite any consideration of adequacy of the language for the purpose.}

\footnote{For our purposes, it is not really relevant what a state actually is. It is only relevant, what kind of information we can attach to (and read off) a specific state and that it represents the world at some moment in time. For instance, this could be a interpretation of a non-logical signature in the context of some logic. It can be more than this as well. For instance, see [Lausen, 2005] for some discussion.}
and understand them purely as abstract objects and characterize their specific properties. The objects themselves have no longer an elaborate structure. Relevant information about the single services is captured by simple properties. In terms of ontologies we would then consider services (epistemologically) as instances and abstract services or Web services as (complex) concepts. Eventually, we can abstract even more in the description of abstract services by ignoring the description of the possible elements of the set. On this level, we would simply use a term or keyword for describing the abstract service and neglect any information about fine-grained structure. Web services on this level are merely considered as atomic objects without any explicit structure (i.e. as symbols). Essentially, we consider keyword-based descriptions here. In the simplest case, one can use free text (sets of words or keywords) here to describe the advertised or requested set of services. In this case, no explicit machine-processable semantics of the description is available for agents performing matching between requests and advertisements; natural language processing techniques can be used to (heuristically) guess the correct meaning of the description. Instead of free text, one could use simple keywords from controlled vocabularies as well to still have a representation with very shallow (i.e. non-explicit), but yet machine-processable semantics. Such a representation is necessarily based on very strong agreements between service requesters and service providers on how to describe and model Web services and goals.

To illustrate the different levels of abstractions in our model, we consider a simple example of a Web service of the financial domain and see how it could be described at the single levels. Consider the following scenario:

ABank is a specialized German bank that offers domestic wire transfers to arbitrary branches of other Austrian Banks, whereby money can only be transferred in European currencies, and a transfer may not exceed 100,000 Euros for private customers and may not exceed 2,000,000 Euros for companies. For the successful execution of the transfer, a (customer-specific) minimum balance of the customer’s account is required. The minimum balance is computed as follows: [...] The respective business service is provided as a Web service.
How can we describe this scenario on the single levels identified in Fig. 3, i.e. how does a description $\mathcal{A}$ of the respective Web service look like?

**Syntactic Level:** In case of a natural language free text, we could literally use the scenario description given above, i.e.

$$\mathcal{A} = \{ \text{ABank is a specialized German bank that offers international wire transfers to arbitrary branches of Austrian Banks, whereby money can only be transferred in European currencies, and a transfer may not exceed 100.000 Euros for private customers and may not exceed 2.000.000 Euros for companies. For the successful execution of the transfer, a (customer-specific) minimum balance of the customer’s account is required. The minimum balance is computed as follows:} \ldots \} \$$

or a more condensed description based on a list of keywords, e.g.

$$\mathcal{A} = \{ \text{Bank Wire Transfer, ABank, in Germany, to Branch of Austrian Bank, European Currencies only, not more than 100.000 Euros for private customers, no more than 2.000.000 Euros for companies, suitable minimum balance needed} \} \$$

Such descriptions are easily processable for humans (with respective domain knowledge), however they are hard to deal with for computers: natural language processing techniques must be applied to heuristically find out relevant relations between the single information chunks, for instance that “100.000” refers to an upper limit of the transfer and not the transferred amount itself, that this is only applies to the case of private customers but not to companies (where the limit is significantly higher), that Austrian branch refers to the target accounts of the wire transfer (and not the source account) etc.

Finally, one could use a term of an agreed upon shared vocabulary to regain machine-processability, e.g. a term from the eClass system that denotes “Account Management”:

$$\mathcal{A} = \{ 25 - 15 - 13 - 04 [AJZ69300201] \}$$

However, in the last case, many relevant information about the Web service that one might want to specify can not be expressed in such simple representation frameworks as controlled vocabularies because of the expressivity of the vocabulary. The quality of the results that are computed in a discovery process are thus of limited quality, and strong agreement between service providers and service requesters are absolutely necessary.

**Light Semantic Level:** Here, the set of (atomic) objects which represent the Web service under consideration could be captured by a formula $\mathcal{A} := \phi(?)$ formally representing the definition of an unary predicate (and thus a set of objects in some domain). In particular, the formula $\phi(?)$ is based on a suitable ontologies describing concepts that are relevant in the financial domain (such as bank transfer, wire transfer, branch, account, currency, private customer etc.) or geographic ontologies (defining elements like Germany, Austria). Hence, as the description $\mathcal{A}$ we could use the

---

6eClass is a standardized classification system for product groups and product properties that is commonly used in Germany. See [http://www.eclass.de/](http://www.eclass.de/) for further information.
following formula φ (with free variable ?t representing the objects under investigation, i.e. transfers with specific properties):

```prolog
?t memberOf BankTransfer and
?t memberOf WireTransfer and
exists (?F, ?T, ?A, ?C ( (?t fromAcc hasValue ?F,
toAcc hasValue ?T,
amount hasValue ?A,
currency hasValue ?C )
and (?F.owner memberOf PrivateCustomer implies
(?A < convertCurrency(100000, ?C, Euro))
and (?F.owner memberOf CompanyCustomer implies
(?A < convertCurrency(2000000, ?C, Euro))
and ?F.bank = ABank
and ?F.branch.locatedIn = Germany
and ?T.branch.locatedIn = Austria
and isEuropeanCurrency(?C)
and ?F.balance > = requiredMinBalance(?F, ?F.owner)

Because of the use of a simple, yet formal language and the use of domain ontologies, such descriptions allow machines to exploit the explicit representation of semantics without any “guessing” based on natural language processing and similar heuristic techniques.

The symbols used to construct such formulas (such as `requiredMinBalance`, `isEuropeanCurrency`, `convertCurrency`, `<`) are defined in ontologies representing domain-specific (or even entity-specific) knowledge.

One can observe that on this level, we can not really represent, what actually happens to the world (i.e. the actual state-changes that one can observe when the executing the Web service: what happens to the balances of accounts ?F and ?T ?) or what the obligations of the requester really are, in terms of the input that is to be provided. For instance, the expression specifies that for any suitable tuple of values (?F, ?T, ?A, ?C), a respective transfer ?t can be delivered by the service, neglecting the dependency on the input of the invoker. It is not clear from the description above, that ?F.balance in the very last part of φ refers to the balance of account ?F before the execution of the Web service (i.e. in the pre-state) and not to the balance of the account afterwards. Such things can thoroughly be represented within a state-based framework, such as the one underlying the most-detailed level of Web services in our model.

**Rich Semantic Level:** On this level, we exploit a state-based perspective on the world and Web services acting therein, and consequently are really able to distinguish between properties of the pre-state of the Web service execution and properties of the respective final state. Furthermore, we can thus clearly point out the obligations of the requester when invoking the Web service (in terms of the inputs that needs to be provided).

To represent our example on this level, one could use a description `A` similar to the following pair of formulae `φ_pre(?F, ?T, ?A, ?C)` and `φ_post(?F, ?T, ?A, ?C)` where `φ_pre` describes the state of the world as well as the requirements on the input values before the execution of the Web service, `φ_post` describes the state of the world after the execution of the Web service as well as the output, and the variables ?F, ?T, ?A, ?C represent the inputs that must be provided by the user, when invoking the Web service, namely the account ?F from which the transfer is initiated, the target account ?T, the amount ?A to be transferred and the currency ?C to which the amount refers:
\[ \phi_{\text{pre}}(\?F, \?T, \?A, \?C) = ( \?F.\text{owner memberOf PrivateCustomer implies} \] 
\[ \?A < \text{convertCurrency}(100000, \?C, \text{Euro}) \] 
\[ \text{and} \ ( \?F.\text{owner instanceOf CompanyCustomer implies} \] 
\[ \?A < \text{convertCurrency}(200000, \?C, \text{Euro}) \] 
\[ \text{and} \ ?F.\text{bank} = \text{ABank} \] 
\[ \text{and} \ ?F.\text{branch.locatedIn} = \text{Germany} \] 
\[ \text{and} \ ?T.\text{branch.locatedIn} = \text{Austria} \] 
\[ \text{and} \ \text{isEuropeanCurrency}(\?C) \] 
\[ \text{and} \ ?F.\text{balance} >\text{= requiredMinBalance}(\?F, \?F.\text{owner}) \] 
\[ \phi_{\text{post}}(\?F, \?T, \?A, \?C) = \exists \ ?t \ ( \] 
\[ \?t \text{ memberOf BankTransfer and} \] 
\[ \?t \text{ memberOf WireTransfer and} \] 
\[ \?t\{ \text{fromAcc hasValue } \?F, \text{ toAcc hasValue } \?T, \text{ amount hasValue } \?A, \text{ currency hasValue } \?C \} \] 
\[ \text{and} \] 
\[ \?F.\text{balance} = \?F.\text{balance@pre} - \text{convertCurrency}(\?A, \?C, \?F.\text{currency}) \] 
\[ \text{and} \] 
\[ \?T.\text{balance} = \?T.\text{balance@pre} + \text{convertCurrency}(\?A, \?C, \?T.\text{currency}) \] 

In particular, the last part of \( \phi_{\text{post}}(\?F, \?T, \?A, \?C) \) describes in detail what happens to the respective account balances after the execution \( (?F.\text{balance}) \) in regard of their values before the execution \( (?F.\text{balance@pre}) \) and the specifically provided input \( (?A, \?C, \?F) \).

In contrast to the last example, we will use a little different and more specific way to model Web service on this most fine-grained level of abstraction in the context of this chapter which is a little closer to the level of light semantic descriptions and eventually allows us to establish a mathematically well-defined relationship between the the level of rich semantic descriptions and the next higher level of abstraction (see Section 4.2 for details). Eventually, this enables semantic interoperability between descriptions on different levels of abstraction (e.g. a description on the rich semantic level and one at the level of light semantic descriptions). A deeper and more general discussion of the level of rich semantic descriptions can be found in [Lausen, 2005].

Each of these levels of abstraction imposes different descriptions of Web services, ranging from detailed characterizations of possible state-transition, less detailed descriptions as (complex) concepts in an ontology to simple unstructured keywords. Consequently, the achievable accuracy of result in the discovery process varies significantly, since more or less structure is actually reflected in the descriptions. On the other hand, the ease of providing the descriptions varies on the single levels as well drastically: Whereas simple keywords are easy to provide, the descriptions of concepts is still not hard but requires more effort. The provision of detailed state-based descriptions definitely requires more elaborate skill of the people creating the formal descriptions. Eventually, the more fine-grained information the descriptions reveal, the more complex algorithms are needed to deal with these descriptions.

Therefore, there is an interesting trade-off between the possible achievable accuracy and the ease of creating the descriptions as well as the potential computational efficiency of the discovery process.
2.4 Goal of the WSMO Discovery Framework

During the design of the WSMO Discovery Framework the following aspects were desired goals that we aimed to achieve:

- It should be a general framework that supports a wide variety of application scenarios. Amongst the supported scenarios should be applications for which efficiency is far more important than very accurate results, as well as applications that require highly accurate results of the discovery process, for instance when aiming at full automation of processes where suitable Web services need to be integrated on the fly.

- The framework should put specific focus on pragmatic aspects that allow workable and scalable solutions for discovery in industrial and end-user applications.

- It should provide an understanding of the trade-offs between ease of use and simplicity of description, efficiency of computation and accuracy of the results and try to balance these aspects of a discovery framework.

- Following the spirit of the WSMO Framework in general, it should facilitate inter-operation between various communities and user groups rather than defining only a specific solution that is suitable for a particular application scenario.

Scope of the chapter. For the sake of space, we do not consider the WSMO Framework in its entirety in this article. In the following, we focus instead on a specific aspect of the overall model of Web Service and Goal descriptions, which can be seen as the most fundamental aspect of such descriptions, namely, the capability description of Goals and Web Services.

In particular, that we do not explore the behavioral part of descriptions (i.e. the Service Interface) as well as the non-functional properties, which clearly are important and interesting aspects in the context of discovery as well. The techniques that are need to address these aspects can be very different to the ones that are used when considering capabilities. However, the aspects themselves are orthogonal to each other, thus, the solutions to the single problems can be combined in a modular way.

---

7Following the principle of a „good-enough“ answer
3 A Conceptual Model for Service Discovery

Approaches to automating discovery of suitable web services must precisely analyze what kind of descriptions can be used for capturing the static and dynamic aspects of a given Web service, and how such descriptions can be exploited for efficiently and accurately locating a requested service. While a number of proposals are available in our area of interest e.g. [Benatallah et al., 2003; Gonzalez-Castillo et al., 2001; Sycara et al., 2002; Li and Horrocks, 2003; Paolucci et al., 2002], none of them has precisely discussed these aspects, but they mainly focused on some specific description languages and frameworks, partly neglecting overall needs. Therefore, we will first define a model that takes into account pragmatic considerations and defines the border line between different steps involved in the process of locating services, namely: goal discovery, goal refinement, web service discovery, and service discovery.

In this section, we will first briefly introduce the gist of the matter of heuristic classification. Then we show how this model is applied in WSMO to structure the service discovery process as the underlying conceptual pattern.

3.1 Heuristic Classification

[Clancey, 1985] provided a landmarking 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. 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 4).

Figure 4: The three major processes of heuristic classification

**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.

---

8In [Omelayenko and Fensel, 2001] we already used heuristic classification as a model for improving the translation process between different XML schema dialects
Refinement. Refinement is the process of inferring a final diagnosis explaining the given findings. This process may include the acquisition of new features 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.

3.2 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 should follow the same pattern (see Figure 5).

Figure 5: The three major processes in service discovery

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 typically 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 distinguish between services and web services (cf. [Preist, 2004]) 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. 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. [Li and Horrocks, 2003]. 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 in the general case as we will further discuss in Section 3.3. In consequence we think it is essential to distinguish between web service and service discovery for coming up with a workable approach that scales and makes realistic assumptions.

9Spoken more precisely, web services are the interface to a software artifact that may help to find and buy services (cf. [Preist, 2004]), however, we neglect this distinction here and try the web service interface and the accessed software artifact identically.
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 (widely) reusable elements. 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 more 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 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.
3.3 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.

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 beforehand 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. [Benjamins et al., 1998]), 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.
4 Web Service Discovery

In this section, 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. Each approach addresses a different level of abstraction in Web service descriptions as discussed in Section 2.3. Our interest 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.

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-based search the huge amount of available services can be filtered or ranked rather quickly. The focus of WSMO Web Service 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 Web services. A query, which is basically a list of keywords, is provided as input to the query engine. The query engine match the keywords from the query against the keywords used to describe the Web service. A query with the same meaning can be formulated by using a synonyms dictionary, like WordNet [Felbaum, 1998]. The meaning 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 [Richardson and Smeaton, 1995]; 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.

Web service descriptions on this level at least consist of a dedicated list of keywords for categorization and indexing. Furthermore, they additionally could include richer, semantic-based elements.

4.2 Discovery based on simple semantic Descriptions

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 [Gruber, 1993], are excellent and prominent conceptual means for this purpose.

---

They provide an explicit and shared terminology, explicate interdependencies between 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 this section, we discuss the description of Web services on the intermediary level of abstraction, where they are understood as sets of (unstructured) elements (that are called services), i.e. as concepts from an ontological perspective. A capability of an abstract service on this level of abstraction is a description which does not depend on dynamic factors, i.e. the current state of the world as well as the requester input needed by the provider. The capability describes only what an advertised abstract service \( A \) can potentially deliver but not under which circumstances (that means preconditions) the single services \( S \in A \) can be actually provided.

The presentation that we use here provides a formal yet comprehensive model of the description of service capabilities and goals. In particular, it is independent from specific knowledge representation languages such as logics. To achieve this, we chose a set-based approach for the description of abstract services and goals. How to ground this modelling and discovery approach eventually in some specific logic is demonstrated in [Keller et al., 2004].

**Abstracting from the basic formal model.** On the most detailed level of abstraction, a (concrete) service \( S \in A \) corresponds to a state transformation on the state space \( \mathcal{U} \): when starting in a specific state \( w \in \mathcal{U} \) we end up in a state \( w' \in \mathcal{U} \) where the world has changed (some effects are observable) and some output has been provided to the user. Both effects \( \text{eff}_S(w, i_1, \ldots, i_{n_A}) \) and outputs \( \text{out}_S(w, i_1, \ldots, i_{n_A}) \) can be seen as sets of objects depending on the initial state \( w \) and the input information \( i_1, \ldots, i_{n_A} \) which has been provided to the service provider by the service requester in \( w \). The circumstances under which a service \( S \) can be delivered by the provider are represented by \( w \) and \( i_1, \ldots, i_{n_A} \). For example, the description of a concrete service provided by a European airline could be that a business-class flight is booked for the male passenger James Joyce on January 5th, 2005 from Dublin to Innsbruck, and 420 Euros are charged on a MasterCard with number #120127933.

If we abstract the description of an abstract service \( A \) from the dependency of the contained concrete services on the provided inputs \( i_1, \ldots, i_{n_A} \), and on the particular initial states \( w \in \text{dom}(A(i_1, \ldots, i_{n_A})) \), the description will only specify which objects we can expect from the abstract service as effects \( \text{eff}_A \) and as outputs \( \text{out}_A \). For example, an abstract description of a European airline could state that the airline provides information about flights within Europe as well as reservations for these flights, but not what input has to be provided and how this input will determine the results of the service provision. In general, we expect completeness but not necessarily correctness of the abstract capability: every concrete service provided by an abstract service should be covered by the capability (on this intermediate level of abstraction), but there might be services which are models of capability but can actually not be delivered as part of the abstract service \( A \) by the provider (since we abstract from the circumstances under which a service can be provided). More formally, we assume

\[
\bigcup_{i_1, \ldots, i_{n_A}} \bigcup_{w \in \text{dom}(A(i_1, \ldots, i_{n_A}))} \text{eff}_S(w, i_1, \ldots, i_{n_A}) \subseteq \text{eff}_A
\]
and 
\[ \bigcup_{i_1, \ldots, i_n, w \in \text{dom}(\mathcal{A}(i_1, \ldots, i_n, A))} \bigcup \text{out}_S(w, i_1, \ldots, i_n, A) \subseteq \text{out}_A \]

Abstracing further beyond the unions over sets for the single initial states \( w \) and input values \( i_1, \ldots, i_n \) might in particular be helpful for a provider to simplify the description of abstract capabilities further, since it allows to skip some details on specific constraints of the delivered objects. For instance an online book store like \texttt{amazon.com} could just advertise to sell any book, i.e. use the concept “BookSale” to (incorrectly, but completely) specify its provided set of services. However, the more abstraction is used beyond these unions (e.g. the airline only specifies to provide tickets for flights all over the world), the less accurate the descriptions of what the service provider is actually able to provide become. Goals specify the desire of a client that he wants to have resolved after consuming a service. They describe the information the client wants to receive as output of the service as well as the effects on the state of the world that the client intends to achieve by using the 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 provision. According to the WSMO model [Lausen et al., 2004], goals refer to the state which is desired to be reached by service execution.

According to this view, abstract services and goals are both represented as \textit{sets of objects} during the Web service discovery step. 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 abstract service description and the objects used in some goal description can or might be interrelated in some way by ontologies. Eventually, such interrelation is needed to establish a match between goals and services.

This way, abstract service and goal descriptions 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 but instead they only refer to domain ontologies which are not specific to a particular goal 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 matching engines in [Paolucci et al., 2002].

An important observation\footnote{Again, this observation has not been made by the Semantic Web community before, because of a strict language focus.} in our approach is that the description of a set of objects for representing a goal or a capability can be interpreted in different ways and, thus, the description by means of a set is \textit{not} semantically unique: A modeler might want to express that either \textit{all} of the elements that are contained in the set are requested (goal) or can be delivered (abstract capability), or that only \textit{some} of these elements are requested (or can be delivered). For this reason, a modeler has to explicitly specify his \textit{intention} when describing the set of relevant objects for a goal or abstract capability. This intention will strongly affect if we consider two descriptions to match. Therefore, goals as well as abstract capabilities are pairs \( \mathcal{D} = (R_{\mathcal{D}}, I_{\mathcal{D}}) \) where \( R_{\mathcal{D}} \) is the set of objects
which are considered as relevant for the description (e.g. a specific set of flights in the travel scenario) and $\mathcal{I}_D \in \{\forall, \exists\}$ is the respective (universal or existential) intention.

**Example.** Consider the following goal and web service

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

$W$: 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 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:

<table>
<thead>
<tr>
<th>Goal / WS</th>
<th>Set $R$ of relevant Objects</th>
<th>Intention of $R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td>${f</td>
<td>f$ is a flight starting at Innsbruck in Austria and ending at any city $c$ located in Ireland $}$</td>
</tr>
<tr>
<td>$W$</td>
<td>${f</td>
<td>f$ is a flight starting at city $s$ and ending at city $e$, $s$ any city in Europe, $e$ any city in Europe $}$</td>
</tr>
</tbody>
</table>

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 is conceptual and effects can 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 outputs and effects in a service discovery request. Augmenting the model discussed here accordingly is a straightforward endeavor.

**Semantic Matching.** In order to consider a goal $G$ and an abstract service $A$ to match on a semantic level, the sets $R_G$ and $R_A$ describing these elements have to be interrelated; precisely spoken, we expect that some set-theoretic relationship between $R_G$ and $R_A$ exists. The most basic set-theoretic relationships that might be considered are the following: $R_G = R_A$, $R_G \subseteq R_A$, $R_A \subseteq R_G$, $R_G \cap R_A \neq \emptyset$, $R_G \cap R_A = \emptyset$.

These set-theoretic relationships provide the basic means for formalizing our intuitive understanding of a match between goals and abstract services. For this reason, they have been considered to some extent already in the literature, for instance in [Li and Horrocks, 2003] or [Paolucci et al., 2002], in the context of Description Logics-based service matchmaking.

On the other hand, we have to keep in mind that in our model these sets only capture part of the semantics of goal and service descriptions $D$, namely the relevant objects for the service requester or service provider. The intentions of these sets in the semantic descriptions $D$ is not considered but clearly affects whether a certain existing set-theoretic relationship between $R_G$ and $R_A$ is considered to actually correspond to (or formalize) our intuitive understanding of a match in the real-world. Therefore, we have to consider the intentions of the respective sets as well. Figure 6 gives an overview of the single set-theoretical re-
Figure 6: Interaction between set-theoretic criteria, intentions and our intuitive understanding of matching.

<table>
<thead>
<tr>
<th>Intention of $G/A$</th>
<th>$I_A = \forall$</th>
<th>$I_A = \exists$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R_G = R_A$</td>
<td>$R_G = R_A$</td>
</tr>
<tr>
<td></td>
<td>$R_G \subseteq R_A$</td>
<td>$R_G \subseteq R_A$</td>
</tr>
<tr>
<td></td>
<td>$R_G \supseteq R_A$</td>
<td>$R_G \supseteq R_A$</td>
</tr>
<tr>
<td></td>
<td>$R_G \cap R_A \neq \emptyset$</td>
<td>$R_G \cap R_A \neq \emptyset$</td>
</tr>
<tr>
<td></td>
<td>$R_G \cap R_A = \emptyset$</td>
<td>$R_G \cap R_A = \emptyset$</td>
</tr>
</tbody>
</table>

Due to space restrictions, we only briefly discuss some entries from the table. A detailed discussion can be found in [Keller et al., 2004]:

1. **Match** – $I_G = \forall$, $I_A = \forall$, $R_G \subseteq R_A$: The requester wants to get all the objects specified as relevant ($I_G = \forall$), whereas the provider claims that he is able to deliver all the objects specified in $R_A$ ($I_A = \forall$). In this case, the requester needs are fully covered by the abstract service since all the requested objects $R_G$ can be delivered by the abstract service according to its abstract capability $A$.

**Example.**

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

$A$: offers information about all flights from any place in Europe to any place in Europe.

12Please note, that when assigning the intuitive notions we assume that the listed set-theoretic properties between $R_G$ and $R_A$ are the strongest ones that actually hold between $R_G$ and $R_A$. 


<table>
<thead>
<tr>
<th>Goal / WS</th>
<th>Set $R$ of relevant Objects</th>
<th>Intention of $R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{G}$</td>
<td>${ f \mid f$ is a flight starting at Innsbruck in Austria and ending at any city $c$ located in Ireland $}$</td>
<td>existential ($\exists$)</td>
</tr>
<tr>
<td>$\mathcal{A}$</td>
<td>${ f \mid f$ is a flight starting at city $s$ and ending at city $e$, $s$ any city in Europe, $e$ any city in Europe $}$</td>
<td>universal ($\forall$)</td>
</tr>
</tbody>
</table>

(2) **ParMatch** – $I_{\mathcal{G}} = \forall$, $I_{\mathcal{A}} = \forall$, $R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$: The requester wants to get all the objects that he has specified as relevant, whereas the provider claims that the abstract service is able to deliver all the objects specified in $R_{\mathcal{A}}$. However, the two sets of reference objects do only overlap. In this case, the requester needs cannot be fully satisfied by the abstract service. At best, the service can contribute to resolve the desire of the client. Thus, we consider this case as a **partial match**.

**Example.**

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

$\mathcal{A}$: offers information about all flights from any place in Europe to any place in Europe.

<table>
<thead>
<tr>
<th>Goal / WS</th>
<th>Set $R$ of relevant Objects</th>
<th>Intention of $R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{G}$</td>
<td>${ f \mid f$ is a flight starting at Innsbruck in Austria and ending at any city $c$ located in any country of the world. $}$</td>
<td>universal ($\forall$)</td>
</tr>
<tr>
<td>$\mathcal{A}$</td>
<td>${ f \mid f$ is a flight starting at city $s$ and ending at city $e$, $s$ any city in Europe, $e$ any city in Europe $}$</td>
<td>universal ($\forall$)</td>
</tr>
</tbody>
</table>

(3) **PossMatch** – $I_{\mathcal{G}} = \forall$, $I_{\mathcal{A}} = \exists$, $R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$: The requester wants to get all the objects that he has specified as relevant, whereas the provider claims he is only able to deliver some of the objects specified in $R_{\mathcal{A}}$. Finally, the set of relevant objects to the service requester is a subset of the set of reference objects advertised by the service provider. In this case, we cannot determine from the given descriptions whether there is a match or not, since we don’t know which (non-empty) subset of $R_{\mathcal{A}}$ the provider actually can deliver. However, it might turn out when examining a more detailed description (or interacting with the provider at service discovery time) that there is a match. Such detailed description is considered during service discovery. Hence, we consider this as a **possible match**.

**Example.**

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

$\mathcal{A}$: offers information about some flights from any place in Europe to any place in Europe.
Goal / WS  | Set $R$ of relevant Objects | Intention of $R$
---|---|---
$\mathcal{G}$  | $\{ f | f \text{ is a flight starting at Innsbruck in Austria and ending at any city } c \text{ located in Ireland } \}$ | existential ($\forall$)
$\mathcal{A}$  | $\{ f | f \text{ is a flight starting at city } s \text{ and ending at city } e, \text{ any city in Europe, } e \text{ any city in Europe } \}$ | universal ($\exists$)

(4) **PossParMatch** $- I_G = \forall, I_A = \exists, R_G \cap R_A \neq \emptyset$: The requester wants to get all the objects that he has specified as relevant, whereas the provider claims that the abstract service is able to deliver only some of the objects specified in $R_A$. Additionally, the two sets of reference objects do only overlap (and this is the strongest applicable set-theoretic relation between $R_G$ and $R_A$). In this case, the requester needs cannot be fully satisfied by the abstract service, but at best only partially. However, we cannot determine from the given descriptions whether there is such a partial match or not, since we don’t know which (non-empty) subset of $R_A$ the provider actually can deliver. When examining a more detailed description (or interacting with the provider at service discovery time) it might turn out that there is a partial match. Such detailed description is considered during service discovery. Hence, we consider this as a possible partial match.

**Example.**

$G$: I want to know about all flights from Innsbruck (Austria) to some place in the world (the client does not necessarily care which one).

$A$: offers information about all flights from any place in Europe to any place in Europe.

Goal / WS  | Set $R$ of relevant Objects | Intention of $R$
---|---|---
$\mathcal{G}$  | $\{ f | f \text{ is a flight starting at Innsbruck in Austria and ending at any city } c \text{ located in any country of the world.} \}$ | universal ($\forall$)
$\mathcal{A}$  | $\{ f | f \text{ is a flight starting at city } s \text{ and ending at city } e, s \text{ any city in Europe, } e \text{ any city in Europe } \}$ | universal ($\forall$)

(5) **NoMatch** $- I_G = \exists, I_A = \forall, R_G \cap R_A = \emptyset$: The requester wants to get some of the objects that he has specified as relevant, whereas the provider claims that the abstract service is able to deliver all the objects specified in $R_A$. However, the two sets of reference objects have no common elements. In this case, the requester needs clearly cannot be satisfied by the abstract service and we consider this case as a non-match.

**Example.**

$G$: I want to know about some flights from Innsbruck (Austria) to some place in the world (the client does not necessarily care which one).

$A$: offers information about all train connections from any place in Europe to any place in Europe.
Goal / WS | Set $R$ of relevant Objects | Intention of $R$
---|---|---
$G$ | $\{ f \mid f$ is a flight starting at Innsbruck in Austria and ending at any city $c$ located in any country of the world. $\}$ | universal ($\exists$)
$A$ | $\{ t \mid t$ is a train connection starting at city $s$ and ending at city $e$, $s$ any city in Europe, $e$ any city in Europe $\}$ | universal ($\forall$)

**Further Discussion of the set-theoretic Criteria.** As shown in Figure 6, 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 this question.

According to most elementary set theory the single criteria we considered above are not completely separated, but the following interdependencies hold:

For any descriptions $R_G, R_W \subseteq U$:

- $R_G = R_W \Rightarrow R_G \subseteq R_W$
- $R_G = R_W \Rightarrow R_G \supseteq R_W$
- $R_G \subseteq R_W, R_G \neq \emptyset \Rightarrow R_G \cap R_W \neq \emptyset$
- $R_G \supseteq R_W, R_W \neq \emptyset \Rightarrow R_G \cap R_W \neq \emptyset$
- $R_G \cap R_W = \emptyset, R_G \neq \emptyset, R_W \neq \emptyset \Rightarrow R_G \not\subseteq R_W, R_G \not\supseteq R_W, R_G \not= R_W$ (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.

$$(R_G = R_W) \preceq (R_G \subseteq R_W), (R_G \supseteq R_W) \preceq (R_G \cap R_W \neq \emptyset)$$

This partial order actually represents a lattice structure. 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 (i.e. $C_1 \not\preceq C_2$), 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. But the weakest criterion for an actual match (wrt. given intentions $I_G$ and $I_W$) does not have to be the only criterion denoting an actual match.

Thus, if $C_1 \preceq C_2$ and $C_2$ denotes a match, $C_1$ 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-Criterion ($R_G \cap R_W \neq \emptyset$) reflects no special additional information for the search request (beyond what is given in the goal description $R_G$ and its corresponding intention $I_G$). 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-Criterion ($R_G \supseteq R_W$) 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-Criterion ($R_G \subseteq R_W$) 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 Identity-Criterion ($R_G = R_W$) 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 Web 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 $R_G \supseteq R_W$ (that means either $R_G \supseteq R_W$ itself or $R_G = R_W$) 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, possibly matches or possibly partially matches a given goal: we have to apply the criterion which is $\preceq$-maximal among the criteria that represent a partial match.

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

Figure 7 represents the result of the discussion in case that the requester just gives a goal description, whereas Figure 8 describes the situation when additionally the requester wants to avoid web services that might deliver irrelevant objects.

<table>
<thead>
<tr>
<th>Intent. $G \ / \ W$</th>
<th>$I_W = \forall$</th>
<th>$I_W = \exists$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match</td>
<td>$R_G \subseteq R_W$</td>
<td>Match</td>
</tr>
<tr>
<td>ParMatch</td>
<td>$R_G \cap R_W \neq \emptyset$</td>
<td>ParMatch $R_G \supseteq R_W$</td>
</tr>
<tr>
<td>PossMatch</td>
<td>$-$</td>
<td>PossMatch $R_G \subseteq R_W$</td>
</tr>
<tr>
<td>PossParMatch</td>
<td>$-$</td>
<td>PossParMatch $R_G \cap R_W \neq \emptyset$</td>
</tr>
<tr>
<td>NoMatch</td>
<td>$R_G \cap R_W = \emptyset$</td>
<td>NoMatch $R_G \cap R_W = \emptyset$</td>
</tr>
<tr>
<td>Match</td>
<td>$R_G \cap R_W \neq \emptyset$</td>
<td>Match</td>
</tr>
<tr>
<td>ParMatch</td>
<td>$-$</td>
<td>ParMatch $R_G \supseteq R_W$</td>
</tr>
<tr>
<td>PossMatch</td>
<td>$-$</td>
<td>PossMatch $R_G \cap R_W \neq \emptyset$</td>
</tr>
<tr>
<td>PossParMatch</td>
<td>$-$</td>
<td>PossParMatch $-$</td>
</tr>
<tr>
<td>Nomatch</td>
<td>$R_G \cap R_W = \emptyset$</td>
<td>Nomatch $R_G \cap R_W = \emptyset$</td>
</tr>
</tbody>
</table>

Figure 7: Which formal criteria should be used for checking different degrees of matching.

**Discovery Scenario.** During the discovery process the scenario for matching between goal and Web service descriptions in general can be considered as follows: A requester specifies his goal by means of a set of relevant objects and the respective intention. Moreover, he might additionally specify that he is interested in Web services which deliver objects that are relevant for his goal only (and such raise the semantic requirement for matches). Furthermore, the requester can indicate in his request whether he is interested in partial matches, in case that no actual matches can be detected. A similar option can be used
Discussion. The proposed modelling approach is based on set theory and ontologies for capturing domain knowledge. By abstracting from dynamic aspects of abstract services, we provide static and general abstract capability descriptions. All the information necessary for checking a match is already available when abstract service descriptions are published, and no interaction with any of the involved parties (requester and provider) is needed for this discovery step. On the other hand, the accuracy we can achieve when is limited. Hence, this discovery step based on such simple descriptions allows an efficient identification of candidate abstract services, but does not guarantee that a matched abstract service will deliver a concrete service fulfilling the requester goal. Abstraction can be used as a means to simplify the description of abstract services by the provider. The overall model is simple, comprehensive and can be implemented for possible matches.

If the discovery request of the client contains only a goal description \((R_G, I_G)\) (without the requirement on returning relevant objects only), then we check for a match using the respective criterion for matching under intentions \((I_G, I_W)\) from Figure 7. 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. Similarly, we proceed for possible and possible partial matches. Eventually, we return the list of actual matches and (if the request indicates that) the lists of partial, possible and possible partial matches.

If the discovery request of the client specifies besides \((R_G, I_G)\) 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 8 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, but rather their interpretation as some sort of match. Hence, we just need an implementation of the different set-theoretic criteria in order to realize a system for matchmaking.

![Figure 8: Which formal criteria should be used for checking different degrees of matching when a requester insists on services delivering relevant objects only.](image-url)
in a logical framework [Keller et al., 2004]. However, the model itself is not based on a specific logical language. The concept of intentions in set-based capability and goal descriptions has not been considered in the literature so far and gives the modeler additional freedom in modelling. Eventually, the use of a set-based model for abstract capabilities can enable the use of Description Logics for classifying and efficiently discovering abstract services to be considered for service discovery. This idea is further elaborated in [Lara et al., 2004].

4.3 Discovery based on rich semantic Descriptions

Using simple semantic annotations for a service as it is 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 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 Web service. That means to consider an abstract 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 for goals we consider the state of the world that is reached after the execution of a service and hence postconditions and effects.

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 9 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: 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”).

\footnote{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.}
As the examples show, 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.

**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 ($\equiv_\emptyset$, $\subseteq_\emptyset$, $\sqsupseteq_\emptyset$, $\sqsubseteq_\emptyset$, $\parallel_\emptyset$) 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** ($W \equiv_\emptyset^1 G$, $W \equiv_\emptyset^+ G$).

  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 $R_W(i_1, \ldots, i_n)$ that the web service claims to deliver when being invoked with input values $i_1, \ldots, i_n$ coincides with the set $R_G$ of objects which are relevant for the requester.

  In this case we write $W \equiv_\emptyset^1 G$ to indicate this particular kind of match.

\[\text{(Of course, this depends on the respective intention used in the web service description.)}\]
If we instead want to consider multiple executions we use the following condition:
For each object \( x \) 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.
In this case we write \( \mathcal{W} \equiv^+ \mathcal{G} \) to indicate this particular kind of match.

- **Subsumes-Match** (\( \mathcal{W} \sqsubseteq^1 \mathcal{G} \), \( \mathcal{W} \sqsubseteq^+ \mathcal{G} \)).
  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 \( R^W(i_1, \ldots, i_n) \) that the web service claims to deliver when being invoked with input values \( i_1, \ldots, i_n \) is a subset of the set \( R^G \) of objects which are relevant for the requester.
  In this case we write \( \mathcal{W} \sqsubseteq^1 \mathcal{G} \) 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 if \( x \) can be delivered by service execution on some input values \( i_1, \ldots, i_n \) then \( x \) is relevant for the client.
  In this case we write \( \mathcal{W} \sqsubseteq^+ \mathcal{G} \) to indicate this particular kind of match.

- **Plugin-Match** (\( \mathcal{W} \sqsupseteq^1 \mathcal{G} \), \( \mathcal{W} \sqsupseteq^+ \mathcal{G} \)).
  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 \( R^W(i_1, \ldots, i_n) \) 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 \( R^G \) which are relevant for the requester.
  In this case we write \( \mathcal{W} \sqsupseteq^1 \mathcal{G} \) 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.
  In this case we write \( \mathcal{W} \sqsupseteq^+ \mathcal{G} \) to indicate this particular kind of match.

- **Intersection-Match** (\( \mathcal{W} \sqcap^1 \mathcal{G} \), \( \mathcal{W} \sqcap^+ \mathcal{G} \)).
  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 \( R^W(i_1, \ldots, i_n) \) 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 \( R^G \) which are relevant for the requester.
  In this case we write \( \mathcal{W} \sqcap^1 \mathcal{G} \) 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.

In this case we write $W \cap_\Theta G$ to indicate this particular kind of match. Obviously, the both criteria $W \cap_1 G$ and $W \cap_2 G$ 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.
5 Related Work

By defining a mathematical model for service, goals and the notion of match-making we provide a basis for applications like semantic Web service repositories and discovery engines. Work in this area has previously leveraged a different (less detailed) formal view on the concept of a Web service: Web services there have been formally mostly considered as sets of objects (describing input, outputs). On a description (language) these sets allow for a natural representation by means of concept expressions in Description Logics. Matching then has been reduced to standard reasoning tasks in the language [Paolucci et al., 2002; Li and Horrocks, 2003], however the dynamics associated with a detailed (state-based) perspective on Web services, can not be represented in such a setting.

Until recently, it seemed to be a common practice in the Semantic Web Community when considering semantic descriptions of Web service, to strictly focus on languages (e.g. Description Logics) rather than an adequate (language-independent) mathematical model of the objects of investigation that underlies such descriptions. The latter question is conceptually interesting and compatible with various concrete representation languages. We consider it as feature and not a back draw to be able to not make commitments on the language levels, but to leave this to the concrete needs of particular implementations.

In the area of software specification, functional descriptions (i.e. the detailed state based perspective) are a well studied phenomena. Hoare [Hoare, 1969] introduced the approach describing a component by its pre- and post-conditions. Numerous systems have been developed since then [Meyer, 1992; Jones, 1990; Spivey, 1992] that follow the same line of description. They have commonalities with the detailed state based perspective of our framework. However our framework is different in two dimensions: (1) we do not fix the underlying language and therefore address the current situation in the Semantic Web with various different languages used in various formalisms, and (2) we explicitly take the existence of background knowledge (represented by some Ontology Ω) and the notion of side effect in the real world modelled into account.

Furthermore there exist several formalisms to specify dynamics such as Transaction Logic (TR) [Bonner and Kifer, 1998] or Situation Calculus [McCarthy, 1963], which also can be used as a model for specification and discovery, however compared to those rather generic approaches we take a minimal approach with respect to the Web service domain, i.e. we only define those aspects in our model that are essential to our domain and do not make unnecessary assumptions on the formalism used. TR for example presents a minimal approach in the sense that it allows describing constraints over any number of states, but does not have specific means for relevant phenomena like input bindings. On the other hand, situation calculi require a certain encoding of states but requires to keep the history of all intermediate states that appears to be too prescriptive in order to serve as a general model.

It is interesting to note, that recently there has been work in the Description Logic Community [Baader et al., 2005b; Baader et al., 2005a] which extends Description Logic with elements of the Situation Calculi (in a decidable manner) and can be seen as work that tries to move decidable languages (Description Logics) that typically can be used for describing Web service on the intermediate level of abstraction in our model a closer to the lowest level in this model.
6 Conclusions

In this chapter we have presented 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 3.3, 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.

For further details, we refer the interested reader to related documents of the WSMO and WSML working groups (cf. www.wsmo.org/TR/), in particular the deliverables [Keller et al., 2004] and [Lausen, 2005].
Acknowledgements

The work summarized in this chapter was carried out in the context of the WSMO and WSML working working groups, and various academic projects.

We want to express our special thanks to Michael Stollberg, Cristina Feier and Dumitru Roman for contributing to Section 2.1. Numerous people have been involved in the discussions and in generating ideas. We especially would like to thank Axel Polleres (Digital Enterprise Research Institute (DERI), University of Innsbruck, Austria) and Michael Kifer (State University of New York (SUNY) at Stony Brook, USA) for their continuous critique and advise. Finally, we would like to thank to all the members of the WSMO, WSML and WSMX working groups for their input and fruitful discussions about the work presented in this chapter.

The presented work has been funded by the European Commission under the projects DIP, Knowledge Web, InfraWebs, SEKT, SWWS, ASG and Esperonto; by Science Foundation Ireland under Grant No. SFI/02/CE1/I131 and the Austrian Federal Ministry for Transport, Innovation, and Technology under the project RW² (FIT-IT contract FFG 809250).

References


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. 22


