

Automatic Location of Services

Uwe Keller¹, Rubén Lara², Holger Lausen¹, Axel Polleres¹, and Dieter Fensel¹

¹ Digital Enterprise Research Institute (DERI) Innsbruck, Austria

`<firstname>.<lastname>@deri.org`

² Tecnología, Información y Finanzas, Madrid, Spain

`rlara@afi.es`

Abstract. The automatic location of services that fulfill a given need is a key step towards dynamic and scalable integration. In this paper we present a model for the automatic location of services that considers the static and dynamic aspects of service descriptions and identifies what notions and techniques are useful for the matching of both. Our model presents three important features: ease of use for the requester, efficient pre-filtering of relevant services, and accurate contracting of services that fulfill a given requester goal. We further elaborate previous work and results on Web service discovery by analyzing what steps and what kinds of descriptions are necessary for efficient and usable automatic service location. Furthermore, we analyze intuitive and formal notions of match that are of interest for locating services that fulfill a given goal. Although having a formal underpinning, the proposed model does not impose any restrictions on how to implement it for specific applications, but proposes some useful formalisms for providing such implementations.

1 Introduction

Current Web service technologies around SOAP [22], WSDL [3] and UDDI [1], only address the syntactical aspects of a Web services and, therefore, only provide protocols and interface descriptions for services in a rigid way that cannot adapt to changing environments without human intervention. The human programmer has to be kept in the loop and scalability as well as economic value of Web services are limited [5]. The vision of semantic Web services is to describe the various aspects of a Web service using explicit, machine-understandable semantics, enabling the automatic location, combination and use of Web services. Semantic Web technologies are being applied to Web services in order to keep the intervention of the human user to the minimum. The basic idea is to add semantic markup that can be exploited to automate the tasks of discovering services, executing them, composing them and enabling seamless interoperation between them [4], thus enabling intelligent Web services.

The description of Web services in a machine-understandable fashion is expected to have a great impact in areas of e-Commerce and Enterprise Application Integration (EAI), as it can enable dynamic and scalable cooperation between different systems and organizations.

An important step towards dynamic and scalable integration, both within and across enterprise boundaries, is the mechanization of service discovery. Automatically locating available services to perform a given business activity can considerably reduce the

cost of making applications and businesses work together and can enable a flexible integration, where providers are dynamically selected based on semantic descriptions of their capabilities and maybe other non-functional criteria such as trust, security, etc.

Scope of the Paper. In this paper we will address the dynamic location of services that can fulfill a given request. Hereby, we concentrate on the most fundamental aspect of service descriptions for the problem at hand: the service capability, i.e. what *functionality* the service provides. Approaches to automatic service location must precisely analyze what kind of service descriptions can be used for capturing the static and dynamic aspects of a given 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. [2, 6, 20, 14, 17], 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, service discovery, and service contracting. We will focus on service discovery and service contracting, analyzing the relevant notions of match. Although different notions of match have been studied in the literature (e.g. [14, 23, 17]), some issues involved in the identification of such notions have not been addressed and will be discussed in this paper.

The remainder of this paper is structured as follows: Section 2 discusses static and dynamic aspects of service descriptions and our assumptions on the problem domain providing a general model for the automatic location of services. Service discovery will be discussed in Section 3 where the relevant notions of match are presented. Service contracting will be addressed in Section 4. Related works in the areas of Web service discovery and software component retrieval will be briefly discussed in Section 5. Finally, we conclude the paper and outline future work in Section 6.

2 A Model for the Automatic Location of Services

A workable approach to automatic service location 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 [18], as well as our assumptions on the elements involved in the location process.

Definition of Service. Recently, it has been pointed out in [18] 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 this document, we use the following interpretation for the term *service*, as described in the conceptual architecture for semantic Web services presented in [18]: *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.

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 \mathcal{A}_P of services. Following the terminology from [18], 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, \dots, i_n will determine what *concrete service* [18] $S \in \mathcal{A}_P$ has to be provided by P .

Description of Requester Needs. Following the approach taken by the Web Service Modeling Ontology (WSMO) [13], a client specifies its needs in terms of what he wants to achieve by a concrete service $S \in \mathcal{A}_P$ of some provider P . Our assumption is that a user will in general care about *what* he wants to get from P , but not about how it is achieved. The conceptual element which formally reflects a desire in WSMO called *goal*. Goals describe what kind of outputs and effects are expected by the client.

A formal Model for Services and Goals. We use a state-based perspective to formalize the concepts involved in the process of automatically locating services. A state $w \in \mathcal{U}$ (where \mathcal{U} is the set of all possible states) determines the properties of the real-world and of the available information at some point in time e.g. the number of rooms currently available in a given hotel. An abstract service \mathcal{A} is considered as a set of state transformations i.e. a relation on the state space \mathcal{U} . Each concrete service $S \in \mathcal{A}$ represents a concrete state transformation $S = (w, w')$, with $w, w' \in \mathcal{U}$. In particular, the delivery of a service S determines the outputs and effects which can be observed by the requester; both can be considered as sets of objects (from some universe U), that are attached to w' . Formally, we denote these sets $out_S(w') \subseteq U$ and $eff_S(w') \subseteq U$.

As mentioned above, whether the provision of a service S is possible depends on some information i_1, \dots, i_n provided by the service requester. What information i_1, \dots, i_n is needed is different for each provider P and each abstract service \mathcal{A}_P provided by P . Hence, \mathcal{A} can be considered as a family of relations $\mathcal{A}(i_1, \dots, i_{n_{\mathcal{A}}}) \subseteq \mathcal{U} \times \mathcal{U}$ where each relation of the family is determined by the concrete input information $i_1, \dots, i_{n_{\mathcal{A}}}$ that the service requester provides i.e. the service description must specify *what* can be delivered by the provider *under which circumstances*. Goals can be formally represented as two distinct sets of objects denoting the required set of outputs $out(\mathcal{G}) \subseteq U$ and effects $eff(\mathcal{G}) \subseteq U$.

Eventually, a service requester is interested in finding service providers P that advertised an abstract service $\mathcal{A}(i_1, \dots, i_{n_{\mathcal{A}}})$ such that there is a concrete service $S = (w, w') \in \mathcal{A}(i_1, \dots, i_{n_{\mathcal{A}}})$ that actually resolves the requesters goal \mathcal{G} , i.e. the

service S achieves a (final) state $w' \in \mathcal{G}$ in which the sets of requested and provided outputs $out_S(w')$, $out(\mathcal{G})$ as well as the respective effects $eff_S(w')$, $eff(\mathcal{G})$ match. What this precisely means is described in detail in Section 3.

Since each element $S = (w, w')$ of $\mathcal{A}(i_1, \dots, i_{n_A})$ is determined by the respective initial state $w \in dom(\mathcal{A}(i_1, \dots, i_{n_A}))$ ¹ and the input information i_1, \dots, i_{n_A} , whether a provider can serve a given concrete service request cannot be determined without knowing $dom(\mathcal{A})$ and i_1, \dots, i_{n_A} ². Unfortunately, we cannot assume that $\mathcal{A}(i_1, \dots, i_{n_A})$ (and therefore $dom(\mathcal{A}(i_1, \dots, i_{n_A}))$) is static over time. In general, the set will dynamically evolve over time: 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 i.e. this concrete service contained in $\mathcal{A}(i_1, \dots, i_{n_A})$ cannot be provided.

We identify two sources of dynamics for the set \mathcal{A} of concrete services that can be provided by a given service provider: (1) the input information i_1, \dots, i_{n_A} that the requester is able and willing to provide and (2) the respective set of concrete services $\mathcal{A}(i_1, \dots, i_{n_A})$ that can currently be delivered for this input. Such dynamics must be considered for determining matches between \mathcal{A} and a user goal \mathcal{G} . Due to these dynamics, some interaction between the parties involved in the matching process – service requester and provider – will be needed in general to determine if a concrete service S fulfilling the requester goal can be provided. This interaction involves communication between the parties and, thus, can be expected to be rather costly. However, it is this communication which enables location results with high precision. We believe that a scalable framework for finding suitable services must address this problem. Our solution to this problem is to split the process into two successive steps, as done in [18]: A first step identifies possible candidate services using less accurate and *static* descriptions of abstract services (so-called *abstract capabilities*), and a second step which applies precise (and possibly dynamic) service descriptions (so-called *contracting capabilities*) and the costly checks (involving communication) of the candidates identified in the first step. We call the first step *service discovery*, whereas the second step is called *service contracting*.

The abstract capability of a service is defined as the set of states that can *potentially* be reached by the provision of such service, independently of the afore-mentioned dynamic factors. It describes only *what* an advertised service can provide but no longer under which circumstances a concrete service S can actually be provided. The contracting capability describes what concrete services can be delivered under what circumstances. It fully describes the family of relations $\mathcal{A}(i_1, \dots, i_{n_A})$. This might involve interaction between both parties for determining if the input available from the requester side can indeed lead to a state w' fulfilling the requester goal.

Assumptions. In order to define a model for the overall location process (including service discovery and contracting), we need to make clear our assumptions on the domain from which we derive the model. Such assumptions are discussed below:

¹ Here $dom(\mathcal{A}(i_1, \dots, i_{n_A}))$ denotes the domain of the state-space relation $\mathcal{A}(i_1, \dots, i_{n_A})$

² More generally, we should consider only input information i_1, \dots, i_{n_A} the requester is able to provide and *willing to disclose* to the provider of an abstract service \mathcal{A} . As discussed in [16], this might involve the use of information disclosure policies and a trust negotiation process.

Pre-defined goals. Human service requesters are not expected to have the required background to formalize their goals. Thus, either goals can be expressed in a familiar language (such as natural language for requesters) or appropriate tools should be available which can support requesters to express their precise needs in a simple manner. Hence, we expect that pre-defined, generic, formal and reusable goals will be available to the requester, defining generic objectives requesters may have. They can be refined (or parameterized) by the requester to reflect his concrete needs, as requesters are not expected to write formalized goals from scratch. We assume that there will be a way for requesters to easily locate such pre-defined goals, e.g. by keyword matching.

Abstract capabilities. Abstract capabilities will abstract contracting capabilities in the sense that they abstract from the input information that is provided by the requester, as well as from the dynamics of the available set of concrete services for this input and at a specific point in time. Abstract capabilities are expected to be complete but not always correct [18]: every concrete service S that can be provided will be a model of the description, but there might be concrete services that are models of the description but cannot be provided by P . For example, a tourism service that provides flights within Europe (but not all possible flights) will describe its abstract service as being able to provide any flight within Europe. However, there might be flights that are a model of this description i.e. they are flights within Europe, but that cannot be provided by P for some reason. This incorrectness is a consequence of the abstraction necessary to make descriptions manageable and the matching of candidate services efficient. Therefore, whether a concrete service can indeed be provided will be determined during the contracting phase i.e. during the contracting phase only providers that can actually provide a suitable concrete service S will be matched.

Contracting capabilities. A service provider will describe the concrete services he can provide by describing its *contracting capability*. The contracting capability will also include the description of what conditions have to be fulfilled for a successful service provision, as well as the relation of the required input to the results of the service. The abstract capability might be automatically derived from the contracting capability and both must be consistent with each other.

It is assumed that the requester goal resulting from refining a pre-defined goal will include the information necessary for contracting, such as the input information the requester can or is willing to offer to a provider. We do not impose that this (possibly big) set of information has to be listed for every goal, but it can be made available to the discovery process by other means e.g. an additional service that provides the information that the requester has available and is willing to disclose. A service will not be selected if the requester is not able to provide all the information required by the provider to actually deliver the required concrete service.

Finally, the communication between requesters and providers will be transparent to us in the descriptions of contracting capabilities i.e. we will not describe and deal with service choreographies but only with a logic representation of the communication act.

Conceptual Model for Service Location. Based on our formal model for services and goals, and the assumptions on the domain given above, we provide a conceptual model for the semantic-based location of services that includes the reuse of pre-defined goals,

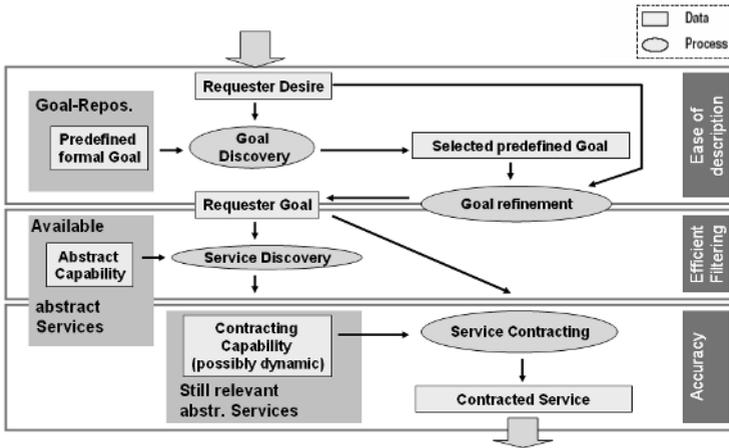


Fig. 1. A conceptual Model for the Discovery Process

the discovery of relevant abstract services, and the contracting of concrete services to fulfill a requester goal. Figure 1 depicts such conceptual model. The different steps of the overall process are:

(1) *Goal Discovery*: Starting from a user desire (expressed using natural language or any other means), goal discovery will locate the pre-defined goal that fits the requester desire from the set of pre-defined goals, resulting on a selected pre-defined goal. Such a pre-defined goal is an abstraction of the requester desire into a generic and reusable goal. (2) *Goal Refinement*: The selected pre-defined goal is refined, based on the given requester desire, in order to actually reflect such desire. This step will result on a formalized requester goal. (3) *Service Discovery*: Available services that can, according to their abstract capabilities, potentially fulfill the requester goal are discovered. As the abstract capability is not guaranteed to be correct, we cannot assure at this level that the service will actually fulfill the requester goal. (4) *Service Contracting*: Based on the contracting capability, the abstract services selected in the previous step will be checked for their ability to deliver a suitable concrete service that fulfills the requester's goal. Such services will eventually be selected.

Let us take as an example a requester who wants to find information about flights from Innsbruck to Madrid on December 21st, 2004. Such requester can express his desire as a text of the form "Search information about flights from Innsbruck to Madrid on December 21st, 2004". This text can be used to perform keyword-based matching of existing pre-defined goals, such as a pre-defined goal for searching flight information.

Once such formal pre-defined goal has been located, it will be refined to reflect the concrete origin and destination given by the requester, as well as the date. This refinement can be done manually (supported by appropriate tools) or automatically from the textual desire.

If a tourism service provider is available and it describes that it can provide flights from any place in Austria to any other place in Europe (as its abstract capability), this service will be considered in the contracting phase. Notice that at this level the dynamics

of the service provision e.g. availability of seats is not considered. Furthermore, we do not expect the service provider to accurately describe all the actual flights it can provide information for, as in general it is not realistic to expect flight information providers to replicate their flight databases in the service description. They will, instead, provide an abstraction of what they can provide.

During the contracting phase it will be checked whether the selected service \mathcal{A} can indeed provides the requested flight information: it will be tested if \mathcal{A} can provide information about flights from Innsbruck to Madrid on the given date. For that purpose, the contracting capability of the service will be used. It might include logic predicates that will actually query the database of the provider to check whether the requested flight information is available. In addition, in case the provider requires extra information from the requester to deliver its service (e.g. personal data of the customer) it will be checked whether the requester can provide such information.

If all the above criteria are fulfilled, the service will be selected and eventually the concrete service provided. In the following sections we concentrate on the notions of match involved in service discovery and service contracting. A more detailed analysis of goal discovery and refinement is beyond the scope of this paper.

3 Service Discovery

As we have sketched in Section 2, the capability of an abstract service can be considered on various levels of abstraction: The most fine-grained perspective on an abstract service \mathcal{A} is to consider it as a family of relations on a state space \mathcal{U} . In our discussion, we identified the problem of dynamics in abstract service descriptions when performing accurate and efficient matching and proposed a solution based on a separation of concerns in the overall location process: a service discovery phase based on more abstract and less accurate capability descriptions to identify possible candidates, followed by a service contracting step based on precise capability descriptions which might involve interaction between service requester and provider.

In this section, we discuss the description of abstract services to be used during the first step of the location process, namely *abstract capabilities* of services. An abstract capability of an abstract service 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 abstract capability describes only *what* an advertised abstract service \mathcal{A} can potentially deliver but no longer under which circumstances the single services $S \in \mathcal{A}$ can be actually provided.

The proposed modelling of abstract capabilities has been designed in a way such that it provides a formal yet comprehensive model of the description of service capabilities and goals. One particular design goal has been the independence of the formal framework from specific logics. For this reason, we chose a set-based approach for the description of abstract services and goals. How to ground this modelling and discovery approach in logics is shown in [10] (using a slightly different terminology).

Modelling Abstract Services by Means of Abstract Capabilities. A (concrete) *service* S (of an abstract service $\mathcal{A}(i_1, \dots, i_{n_{\mathcal{A}}})$) 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 $eff_S(w, i_1, \dots, i_{n_A})$ and outputs $out_S(w, i_1, \dots, i_{n_A})$ can be seen as sets of objects depending on the initial state w and the input information i_1, \dots, 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, \dots, 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 01233.

If we abstract the description of an *abstract service* \mathcal{A} from the dependency on the contained concrete services, on the provided inputs i_1, \dots, i_{n_A} , and on the particular initial states $w \in dom(\mathcal{A}(i_1, \dots, i_{n_A}))$, the description will only specify which objects we can expect from the abstract service as effects $eff_{\mathcal{A}}$ and as outputs $out_{\mathcal{A}}$. 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 abstract capability, but there might be services which are models of the abstract capability but cannot be delivered as part of the abstract service \mathcal{A} by the provider (since we abstract from the circumstances under which a service can be provided). More formally, we assume $\bigcup_{i_1, \dots, i_{n_A}} \bigcup_{w \in dom(\mathcal{A}(i_1, \dots, i_{n_A}))} eff_S(w, i_1, \dots, i_{n_A}) \subseteq eff_{\mathcal{A}}$ and $\bigcup_{i_1, \dots, i_{n_A}} \bigcup_{w \in dom(\mathcal{A}(i_1, \dots, i_{n_A}))} out_S(w, i_1, \dots, i_{n_A}) \subseteq out_{\mathcal{A}}$. Abstracting further beyond the unions over sets for the single initial states w and input values i_1, \dots, i_{n_A} 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. 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 [13], 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 *sets of objects* during the 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.

An important observation in our approach is that the description of a set of objects for representing a goal or an abstract capability can be interpreted in different ways and, thus, the description by means of a set is *not* semantically unique: A modeler

might want to express that either *all* of the elements that are contained in the set are requested (goal) or can be delivered (abstract capability), or that only *some* of these elements are requested (or can be delivered). For this reason, a modeler has to explicitly specify his 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 $\mathcal{D} = (R_{\mathcal{D}}, \mathcal{I}_{\mathcal{D}})$ where $R_{\mathcal{D}}$ is the set of objects which are considered as relevant for the description and $\mathcal{I}_{\mathcal{D}} \in \{\forall, \exists\}$ is the respective (universal or existential) intention. 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 \mathcal{G} and an abstract service \mathcal{A} to match on a semantic level, the sets $R_{\mathcal{G}}$ and $R_{\mathcal{A}}$ describing these elements have to be interrelated; precisely spoken, we expect that some set-theoretic relationship between $R_{\mathcal{G}}$ and $R_{\mathcal{A}}$ exists. The most basic set-theoretic relationships that might be considered are the following: $R_{\mathcal{G}} = R_{\mathcal{A}}$, $R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$, $R_{\mathcal{A}} \subseteq R_{\mathcal{G}}$, $R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$, $R_{\mathcal{G}} \cap R_{\mathcal{A}} = \emptyset$.

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 [14] or [17], 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 \mathcal{D} , namely the relevant objects for the service requester or service provider. The *intentions* of these sets in the semantic descriptions \mathcal{D} is not considered but clearly affects whether a certain existing set-theoretic relationship between $R_{\mathcal{G}}$ and $R_{\mathcal{A}}$ 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 2 gives an overview of the single

Intention of $\mathcal{G} / \mathcal{A}$	$\mathcal{I}_{\mathcal{A}} = \forall$		$\mathcal{I}_{\mathcal{A}} = \exists$	
$\mathcal{I}_{\mathcal{G}} = \forall$	$R_{\mathcal{G}} = R_{\mathcal{A}}$	Match	$R_{\mathcal{G}} = R_{\mathcal{A}}$	PossMatch
	$R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$	Match	$R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$	PossMatch
	$R_{\mathcal{G}} \supseteq R_{\mathcal{A}}$	ParMatch	$R_{\mathcal{G}} \supseteq R_{\mathcal{A}}$	ParMatch
	$R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$	ParMatch	$R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$	PossParMatch
	$R_{\mathcal{G}} \cap R_{\mathcal{A}} = \emptyset$	Nonmatch	$R_{\mathcal{G}} \cap R_{\mathcal{A}} = \emptyset$	Nonmatch
$\mathcal{I}_{\mathcal{G}} = \exists$	$R_{\mathcal{G}} = R_{\mathcal{A}}$	Match	$R_{\mathcal{G}} = R_{\mathcal{A}}$	Match
	$R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$	Match	$R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$	PossMatch
	$R_{\mathcal{G}} \supseteq R_{\mathcal{A}}$	Match	$R_{\mathcal{G}} \supseteq R_{\mathcal{A}}$	Match
	$R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$	Match	$R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$	PossMatch
	$R_{\mathcal{G}} \cap R_{\mathcal{A}} = \emptyset$	Nonmatch	$R_{\mathcal{G}} \cap R_{\mathcal{A}} = \emptyset$	Nonmatch

Fig. 2. Interaction between set-theoretic criteria, intentions and our intuitive understanding of matching

set-theoretical relations as well as their interpretation³ as matches when considering the request and provider intentions. In the table we distinguish several forms of matches: A match (**Match**) means that \mathcal{A} completely satisfies \mathcal{G} , a partial match (**ParMatch**) means that \mathcal{A} partially satisfies \mathcal{G} and additional abstract services would be required to completely satisfy the request, a possible match (**PossMatch**) means that there might be an actual match given a more detailed description (at contracting time) of the abstract service, a possible partial match (**PossParMatch**) means that there might be a partial match given more detailed description (at contracting time) of the abstract service or a non-match (**Nonmatch**). Due to space restrictions, we only briefly discuss some entries from the table. A detailed discussion can be found in [10]: (1) $I_{\mathcal{G}} = \forall, I_{\mathcal{A}} = \forall, R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$: The requester wants to get all the objects specified in $R_{\mathcal{G}}$ ($I_{\mathcal{G}} = \forall$), whereas the provider claims that he is able to deliver all the objects specified in $R_{\mathcal{A}}$ ($I_{\mathcal{A}} = \forall$). In this case, the requester needs are fully covered by the \mathcal{A} since all the requested objects $R_{\mathcal{G}}$ can be delivered by the abstract service according to its abstract capability. (2) $I_{\mathcal{G}} = \forall, I_{\mathcal{A}} = \forall, R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$: The requester wants to get all the objects in $R_{\mathcal{G}}$, whereas the provider claims that \mathcal{A} is able to deliver all the objects specified in $R_{\mathcal{A}}$. In this case, the requester needs cannot be fully satisfied by \mathcal{A} . At best, the \mathcal{A} can contribute to resolve the desire of the client. Thus, we consider this case as a *partial match*. (3) $I_{\mathcal{G}} = \forall, I_{\mathcal{A}} = \exists, R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$: The requester wants to get all the objects in $R_{\mathcal{G}}$, whereas the provider claims he is only able to deliver some of the objects in $R_{\mathcal{A}}$. In this case, we cannot determine from the given descriptions whether there is a match or not. However, it might turn out when examining a more detailed description there is a match. Such detailed description is available during service contracting (see Section 4). Hence, we consider this as a *possible match*.

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 in a logical framework [10]. 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 contracting. This idea is further elaborated in [12].

³ Please note, that when assigning the intuitive notions we assume that the listed set-theoretic properties between $R_{\mathcal{G}}$ and $R_{\mathcal{A}}$ are the *strongest* ones that actually hold between $R_{\mathcal{G}}$ and $R_{\mathcal{A}}$.

4 Service Contracting

In this section we present service contracting, the last step in the conceptual model presented in Section 3. For service contracting, only the services discovered as discussed in the previous section will be considered. As mentioned above, service contracting will exploit the contracting capability of such services, interpreted as a family of relations $\mathcal{A}(i_1, \dots, i_{n_A}) \subseteq \mathcal{U} \times \mathcal{U}$ where i_1, \dots, i_{n_A} is the input information that has to be made available by the requester.

A contracting capability will describe what input information is required for providing a concrete service and what conditions it must fulfill (i.e. *service preconditions*, denoted by $A_{pre}(i_1 \dots i_{n_A})$), and what conditions the objects delivered fulfill depending on the input given (i.e. *service postconditions*, denoted by $A_{post}(i_1 \dots i_{n_A}, x)$ where x denotes objects that are delivered by a service execution with given input values. We formalize a contracting capability of an abstract service as as follows:

$$\mathcal{A} : \forall x, i_1 \dots i_{n_A}. (as(x, i_1 \dots i_{n_A}) \leftrightarrow A_{pre}(i_1 \dots i_{n_A}) \wedge A_{post}(i_1 \dots i_{n_A}, x)) \quad (1)$$

where $as(x, i_1 \dots i_{n_A})$ represents what objects x the service will deliver for a given input set $i_1 \dots i_{n_A}$. Therefore, the dependency of the service results on the input given is explicitly described. Please note that according to (1) the service is not considered to deliver anything meaningful if the precondition can not be fulfilled by the input provided by the requester. As the dependency on the initial state (e.g. availability of rooms at request time) is dynamic over time, the evaluation of $as(x, i_1 \dots i_{n_A})$ will require interaction with the provider. A goal \mathcal{G} is modelled in terms of a predicate $g(x)$ that describes the relevant objects for the requester.

Single or Multiple Concrete Services. We can enrich the set of matching notions presented in the previous section with an orthogonal dimension: we can express that we can satisfy a particular matching notion wrt. a single concrete service as well as wrt. an arbitrary number of concrete services. This results in additional matching notions that capture additional semantics in a given requester goal. Let us take a requester goal given by the following informal text: "I want to know all flights from Innsbruck to Madrid between 12/10/2004 and 14/10/2004" and an abstract service with the following (informal) capability: "The service provides information about all flights from any place in Austria to any place in Spain on any specific date. Therefore, a single concrete service cannot fulfill the requester goal, but a set of successive concrete services (of the same abstract service) can together fulfill the requester goal: one for each day in the requested period of time. These concrete services correspond to different input information provided by the requester. This can be seen as a simple form of composition, but it can still be captured in our contracting framework and in the definition of the formal proof obligations that have to be checked to determine whether concrete services fulfilling the goal can be contracted.

Extending the Set-Based Modelling. The intuitive notions of match that can be considered at contracting time will be the same as in Figure 2, except for the *possible match* and *possible partial match* notions; as contracting will precisely determine what

concrete services can be provided, we can fully determine whether a service fulfills the requester goal. However, as we introduce the dependency on the input information in the contracting capability, we cannot model our notions of match in terms of set-theoretic relations. Therefore, we replace the set-theoretic relations by the following logical relations, where \mathcal{O} is the set of ontologies that give the terminology used by both descriptions:

1. $R_{\mathcal{G}} = R_{\mathcal{A}}$ for a single concrete service:

$$\mathcal{A}, \mathcal{G}, \mathcal{O} \models \exists i_1, \dots, i_{n_{\mathcal{A}}}. (\forall x. (g(x) \leftrightarrow as(x, i_1 \dots i_{n_{\mathcal{A}}}))) \quad (2)$$

For multiple concrete services:

$$\mathcal{A}, \mathcal{G}, \mathcal{O} \models \forall x. (\exists i_1, \dots, i_{n_{\mathcal{A}}}. (g(x) \leftrightarrow as(x, i_1 \dots i_{n_{\mathcal{A}}}))) \quad (3)$$

2. $R_{\mathcal{G}} \subseteq R_{\mathcal{A}}$ for a single concrete service:

$$\mathcal{A}, \mathcal{G}, \mathcal{O} \models \exists i_1, \dots, i_{n_{\mathcal{A}}}. (\forall x. (g(x) \rightarrow as(x, i_1 \dots i_{n_{\mathcal{A}}}))) \quad (4)$$

For multiple concrete services:

$$\mathcal{A}, \mathcal{G}, \mathcal{O} \models \forall x. (\exists i_1, \dots, i_{n_{\mathcal{A}}}. (g(x) \rightarrow as(x, i_1 \dots i_{n_{\mathcal{A}}}))) \quad (5)$$

3. $R_{\mathcal{A}} \subseteq R_{\mathcal{G}}$ for a single concrete service:

$$\mathcal{A}, \mathcal{G}, \mathcal{O} \models \exists i_1, \dots, i_{n_{\mathcal{A}}}. (\forall x. (g(x) \leftarrow as(x, i_1 \dots i_{n_{\mathcal{A}}}))) \quad (6)$$

For multiple concrete services:

$$\mathcal{A}, \mathcal{G}, \mathcal{O} \models \forall x. (\exists i_1, \dots, i_{n_{\mathcal{A}}}. (g(x) \leftarrow as(x, i_1 \dots i_{n_{\mathcal{A}}}))) \quad (7)$$

4. $R_{\mathcal{G}} \cap R_{\mathcal{A}} \neq \emptyset$ for a single concrete service:

$$\mathcal{A}, \mathcal{G}, \mathcal{O} \models \exists i_1, \dots, i_{n_{\mathcal{A}}}. (\exists x. (g(x) \wedge as(x, i_1 \dots i_{n_{\mathcal{A}}}))) \quad (8)$$

For multiple concrete services the proof obligation for this matching criterion is logically equivalent to the one used for a single concrete service.

If none of the above holds, then the service cannot provide any of the required results, which is similar to the set-theoretic relation $R_{\mathcal{G}} \cap R_{\mathcal{A}} = \emptyset$.

These relations (together with the respective intentions $I_{\mathcal{A}} = \forall$, $I_{\mathcal{A}} = \exists$, $I_{\mathcal{G}} = \forall$, and $I_{\mathcal{G}} = \exists$) can be used for precisely determining if the service fulfills the goal given.

Notice that the input involved in the relations above has to be made available by the requester. This does not impose that the requester has to list for every goal all the information he has available, but he can for example offer a service that provides his available information on demand. In addition, some input information can automatically be extracted from the goal description e.g. if the requester wants to fly from Innsbruck to Madrid we already know that he can provide Innsbruck as the departure location and Madrid as the arrival location. However, how this information is made available to the

contracting process is beyond the scope of the paper, and it is assumed that it will be available in some way during the contracting phase.

In the logical relations above we do not put any restriction on the logical expressivity allowed, as our intention is to formalize the intuition behind service contracting. Similar relations have been formalized using Transaction Logic and implemented using \mathcal{F} LORA-2 in [11] and [12].

Contracting based on the above formalizations will in general involve communication with the provider and will be expensive. For example, we can determine whether a given flight can be booked by communicating with the provider, as such availability is dynamic over time and, furthermore, it is not realistic to expect an airline to include its complete flights database as part of the contracting capability description. However, such contracting will be only attempted for abstract services that have been efficiently filtered at discovery time and that are already known as relevant for the goal at hand.

5 Related Work

Software Component Retrieval. The problem of semi-automatically retrieving software components is very similar to the automatic location of services. Specification matching has been proposed in several works e.g. [8, 9, 19, 23] to evaluate how software components relate to a given query i.e. user's need. Specification matching relies on the axiomatization of software components and user queries. A formal (logical) relation is then defined and whether a given query and component satisfy this relation is checked. Such a relation must capture the notion of reusability i.e. if the relation holds for formally specified components and queries, it means that the component can be reused to solve the problem captured by the query.

The work on software component retrieval has not defined a conceptual model for the location of relevant components, but only different notions of match for a given query and a given component have been studied [12]. While such notions of match focus on locating a software component that can be used in the place where the software component represented by the query could, in service discovery we focus on what results can be delivered by the service. Therefore, the notions of match studied for software component retrieval have to be adapted to the Web services domain. Service contracting is not directly considered, as it is outside the application area of software component retrieval. A more detailed account of the work on software component retrieval and its relation to service discovery is given in [12].

Automatic Web Service Discovery. A number of proposals for using Description Logics [15] and OWL-S [4], or similar descriptions for the automatic discovery of services are available [17, 14, 2, 6]. However, none of them provides a conceptual model and they regard discovery as a one step process. In addition, these approaches are not suitable for contracting as they do not employ rules for describing the relation between the results of the service and the input given.

METEOR-S discovery [21] is very similar to the approaches mentioned above, but it uses request templates similar to our pre-defined goals. It also annotates service reg-

istries, specializing them on a given domain and exploiting such annotations during discovery. However, it does not define a conceptual model and it is not suitable for contracting.

LARKS [20] deals with the description of agent capabilities and requests⁴ and the matchmaking of those. The discovery model used in LARKS defines different filters of different complexity and accuracy, allowing the user to select the trade-off between the efficiency and accuracy he needs. However, this model does not address the problem of the different levels of abstraction that are expected in service descriptions, and does not discuss how the requests will be defined by users. Furthermore, it does not consider the contracting of services.

For service discovery, none of the afore-mentioned proposals has discussed the intuitive notions of match a requester or a provider have in mind when requesting or advertising a service i.e. the intentions. The work in [7] discusses variance in service discovery, a complementary aspect to the intentions we have discussed in this paper.

Our previous work on service discovery and contracting [11] already offered a distinction between these two steps. We have built on top of it and examined the conceptual model described in [18] to elaborate a comprehensive conceptual model including both aspects and to discuss the notions of match involved.

6 Conclusions and Future Work

In this paper we presented a model for the automatic location of services that considers the static and dynamic aspects of service descriptions and identifies what notions of match and techniques are useful for the matching of both. Our model presents three important features: ease of use for the requester, efficient pre-filtering of relevant services, and accurate contracting of services that fulfill a given requester goal. We further elaborated previous work and results on Web service discovery by analyzing what steps and what kind of descriptions are necessary for an efficient and usable automatic service location. Furthermore, we analyzed the intuitive and formal notions of match that are of interest for locating services that fulfill a given goal. Although having a formal underpinning, the proposed model does not impose any restrictions on how to implement it for specific applications, but proposes some useful formalisms for providing such implementation. Recently we started with the prototypical implementation of the proposed framework. As soon as the implementation of the prototype will be completed, we will start with the evaluation of our model based on concrete use cases. Further refinement of the model and the respective implementation based on the empirical results obtained from our experiments is part of our future work.

Acknowledgements. We would like to thank all members of the WSMO and WSML Working Groups for fruitful discussions. In particular, Michael Kifer contributed signif-

⁴ Although LARKS is a language for describing agent capabilities, it can equally be applied to Web services.

icantly in discussions of proposed service description and discovery model. This work has been supported by the SFI (Science Funds Ireland) under the DERI-Lion project, the European Commission under the projects DIP, Knowledge Web, SEKT, and SWWS and by FIT-IT under the project RW².

References

1. T. Bellwood, L. Clément, D. Ehnebuske, A. Hatley, Maryann Hondo, Y.L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, and C. von Riegen. UDDI Version 3.0, July 2002.
2. B. Benatallah, M-S. Hacid, C. Rey, and F. Toumani. Request rewriting-based Web Service Discovery. In *The Semantic Web - ISWC 2003*, pages 242–257, October 2003.
3. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. <http://www.w3.org/TR/wsdl>, March 2001.
4. The OWL Services Coalition. OWL-S 1.1 Beta Release. July 2004.
5. D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2), 2002.
6. J. Gonzalez-Castillo, D. Trastour, and C. Bartolini. Description logics for matchmaking of services. In *KI-2001 Workshop on Applications of Description Logics*, September 2001.
7. S. Grimm, B. Motik, and C. Preist. Variance in e-Business Service Discovery. *Semantic Web Services Workshop at ISWC 2004*, November 2004.
8. J.J Jeng and B.H.C. Cheng. Using Automated Reasoning Techniques to Determine Software Reuse. *Intl. Journal of Soft. and Know. Engineering*, 2(4), Dec. 1992.
9. J.J Jeng and B.H.C. Cheng. Specification Matching for Software Reuse: A Foundation. In *SSR'95. ACM SIGSOFT*. ACM Press, 1995.
10. U. Keller, R. Lara, and A. Polleres (eds.). WSMO Web Service Discovery. Technical report, DERI, November 2004.
11. M. Kifer, R. Lara, A. Polleres, C. Zhao, U. Keller, H. Lausen, and D. Fensel. A Logical Framework for Web Service Discovery. In *Semantic Web Services Workshop at ISWC*, 2004.
12. R. Lara, W. Binder, I. Constantinescu, D. Fensel, U. Keller, J. Pan, M. Pistore, A. Polleres, I. Toma, P. Traverso, and M. Zaremba. Semantics for Web Service Discovery and Composition. Technical report, Knowledge Web, December 2004.
13. H. Lausen, D. Roman, and U. Keller (editors). Web Service Modeling Ontology (WSMO). Working draft, DERI, March 2004. <http://www.wsmo.org/2004/d2/v0.2/>.
14. Lei Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. In *WWW'03*, Budapest, Hungary, May 2003.
15. D. Nardi, F. Baader, D. Calvanese, D. L. McGuinness, and P. F. Patel-Schneider (eds.). *The Description Logic Handbook*. Cambridge, January 2003.
16. D. Olmedilla, R. Lara, A. Polleres, and H. Lausen. Trust Negotiation for Semantic Web Services. In *SWSWPC Workshop at ICWS 2004*, July 2004.
17. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Service Capabilities. In *ISWC*, pages 333–347. Springer Verlag, 2002.
18. Chris Preist. A Conceptual Architecture for Semantic Web Services. In *Proceedings of the International Semantic Web Conference 2004 (ISWC 2004)*, November 2004.
19. E.J. Rollings and J.M. Wing. Specifications as Search Keys for Software Libraries. In *Proceedings of the Eighth International Conference on Logic Programming*, June 1991.

20. K. Sycara, S. Widoff, M. Klusch, and J. Lu. LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. *Autonomous Agents and Multi-Agent Systems*, pages 173–203, 2002.
21. K. Verma, K. Sivashanmugam, A. Sheth, and A. Patil. METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. *Journal of Information Technology and Management*, 2004.
22. W3C. SOAP Version 1.2 Part 0: Primer, June 2003.
23. A.M. Zaremski and J.M. Wing. Specification Matching of Software Components. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 6:333–369, 1997.