Variance in e-Business Service Discovery

Stephan Grimm[1], Boris Motik[1], and Chris Preist[2]

[1] FZI Research Center for Information Technologies at the University of Karlsruhe
Karlsruhe, Germany
{grimm,motik}@fzi.de
[2] HP Laboratories
Bristol, UK
chris.preist@hp.com

Abstract. Automating the process of B2B partner discovery and contract negotiation is expected to significantly optimise company processes. Numerous existing proposals for discovery follow the approach where service descriptions are expressed by concept expressions in description logics (DL), and description matching is performed by well-known DL inferences. However, these approaches do not always produce results one might intuitively expect, due to a gap between the formal semantics of service descriptions and human intuition. In this paper, we address this problem by analysing the connection between the modeler's intuition and formal logic used to operationalise discovery. Furthermore, we show how to correctly map the intuition into description logic constructs. Finally, we investigate different inferences used to realise service discovery.

1 Introduction

In the vision of Semantic Web Services [5, 9], online services are annotated with semantic descriptions, thus allowing to automate various business processes such as service discovery and composition. The semantic description specifies the functionality of a service, both in terms of the business value that the service provides and in terms of the business interactions involved. The vocabulary for expressing semantic descriptions is defined by upper-level ontologies such as OWL-S[3] and WSMO[4]. By reusing a common vocabulary, service modelers can produce semantic descriptions of their services that can be shared and understood on the Web. Currently, service modelling requires a significant amount of expertise. However, for wide acceptance of Semantic Web Services, modelling should become easier and more intuitive.

Semantic service descriptions can be used to automate the process of locating e-business services that meet the needs of the requestor; this process is also known as *semantic service discovery*. A framework that aims at automating discovery requires the following elements:

[3] http://www.daml.org/services/owl-s/1.0/
[4] http://wsmo.org

- a language for expressing service descriptions with a formal semantics which matches the modeler's intuition.
- matching algorithms capable of reasoning with service descriptions used to realise the discovery task.

In [4, 2, 13, 8, 14] description logics (DL[5]) have been used as formal languages for expressing service descriptions. DLs are closely connected to OWL[6], the ontology language for the Semantic Web. Hence, using DLs ensures compatibility with existing ontology standards. Furthermore, the formal semantics of description logics allows precise definition of the semantics of service descriptions. Finally, discovery algorithms can be formally defined in terms of well-known DL inferences. Unfortunately, initial experiments show that DL modelling primitives do not directly correspond to the modeler's intuition [4], so straightforward modelling of service descriptions is likely to produce counterintuitive results. In this paper, we present our preliminary methodological guidelines for modelling service descriptions. In particular, we focus on the notion of variance. Namely, a service description usually represents numerous variants of a concrete service that can be performed. We believe that precise control of variance in service descriptions is crucial to ensure the quality of the discovery process. The main goal of our work is to provide methodological guidelines to ensure that the formal semantics corresponds to the modeler's intuition.

This paper is structured as follows. In Section 2, we informally introduce the concepts used for modelling service semantics. In Section 3, we show how to operationalise these concepts in a logical framework. In Section 4, we analyse different inferences used to realise service discovery and their relationship to the modeler's intuition. In Section 5, we discuss the use of these inferences for discovery and present a new approach to ranking discovery results.

2 Modelling Service Semantics

The term 'service' can be used in different ways [15, 11]. In particular, it sometimes refers to an abstract business interaction between two parties, and sometimes to a computational entity with a web service interface. As our notion of discovery is based on the business-level semantics of a service, we use the term 'service' in the first sense.

Following [11], we distinguish between a concrete *service instance* and an abstract service class. A service instance corresponds to a contract between a provider and a requestor, defining all details of a business interaction.

An advertisement or a request for a service can naturally be understood as a set of service instances acceptable to the provider or the requestor. Hence, a *service description* should intuitively be understood as defining a space of possible service instances. A service description is therefore an abstract class acting as a template for service instances.

[5] We assume the reader is familiar with the basics of description logics. See [1] for an overview.

[6] http://www.w3.org/2004/OWL/

2.1 Representing Service Instances

We put the examples we use in the context of a logistics scenario, where providers of shipping services offer transportation of goods between certain locations and where requestors wish to transport their goods.

In this logistics context, a service instance contains the exact information about the item to be shipped, the cities of origin and destination and the time and date when the item should be picked up. Such a concrete service instance can be represented using a simple relational data model. Table 1 represents a concrete service instance *c1* in a relational form. The service is shipping (1), and takes place from Plymouth (5) to Bremen (6). Both Plymouth and Bremen are Locations (3,4). Only one item, *p1*, will be shipped (7), and this item is a crate (2). The properties of *p1*, such as weight, width, height and depth, are given by (8 – 11). Finally, the price of shipping is 250 EUR (12), the package will be picked up on 01.04.2004 (13) and will be delivered on 04.04.2004 (14).

(1)	$Shipping(c1)$	(6)	$to(c1, Bremen)$	(11)	$depth(p1, 3)$
(2)	$Crate(p1)$	(7)	$item(c1, p1)$	(12)	$price(c1, 250)$
(3)	$Location(Plymouth)$	(8)	$weight(p1, 50)$	(13)	$pickUp(c1, '01.04.2004')$
(4)	$Location(Bremen)$	(9)	$width(p1, 1)$	(14)	$delivery(c1, '04.04.2004')$
(5)	$from(c1, Plymouth)$	(10)	$height(p1, 2)$		

Table 1. A Service Instance

The same service instance is represented as a directed labelled graph in Figure 1. This notation is compatible with the usual semi-structured data models such as OEM [10] or RDF [3].

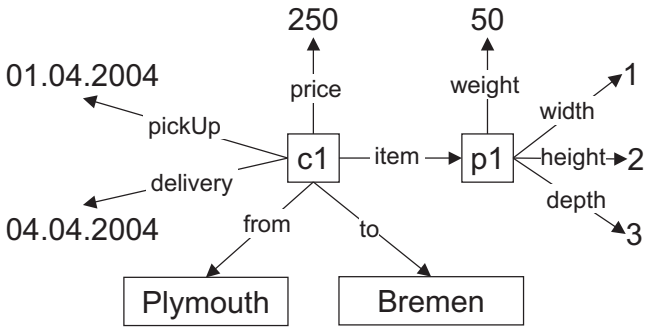

Fig. 1. A Service Instance Represented as a Directed Labelled Graph

2.2 Variance in Service Descriptions

Service descriptions typically specify numerous different service instances, thus introducing *variance*. For example the service provider might support shipping from all UK cities to all cities in Germany. The appropriate service description

should then contain service instances covering all possible pairs of source and destination locations, including e.g. a service instance where shipping is conducted from Plymouth to Bremen, as well as one where shipping is conducted from Dover to Hamburg (represented in Figure 2). Since the service description does not constrain all properties (such as *weight*) to concrete values, this induces variance.

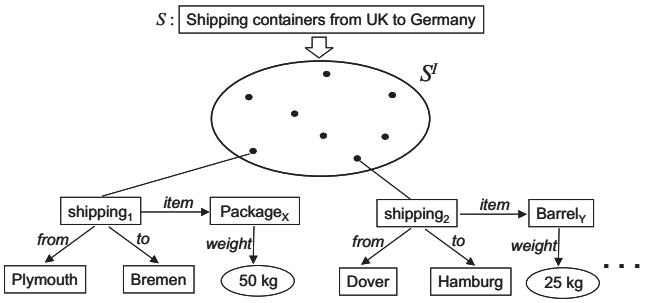

Fig. 2. Expressing Variance in Service Descriptions

The above example shows variance which represents *intended diversity* in service instances. However, another form of variance is due to *incomplete knowledge*. Inspired by the model-theoretic semantics of first-order logic, we use the notion of *possible worlds* to intuitively explain this kind of variance. Namely, under open-world semantics, a modeler must explicitly state which service instances are not covered by the service description. For each aspect of the service instance which has not been fully specified there are several possible worlds, each one reflecting a particular way of resolving incompleteness.

It is beneficial to separate the variance due to intended diversity from the variance due to incomplete knowledge at the methodological level. This in turn influences the way service descriptions are modelled, as well as the way discovery is performed. Variance due to incomplete knowledge is resolved by allowing many different possible worlds, each resolving unspecified issues in a different way. Variance due to intended diversity is resolved by allowing alternative service instances within one possible world. Hence, given incomplete information is resolved in a specific way (resulting in a particular possible world), the intended diversity is reflected by different alternative service instances corresponding to the service description.

Consider a service description of a service provider which advertises shipping from all UK cities to all German cities, as illustrated in Figure 3. In every possible world there will be service instances for shipping from, say, Plymouth to Bremen. In addition, because the provider has not explicitly stated that he does not ship to the USA, there will be a possible world which additionally contains service instances for shipment from Plymouth to Boston. The two kinds of variance are

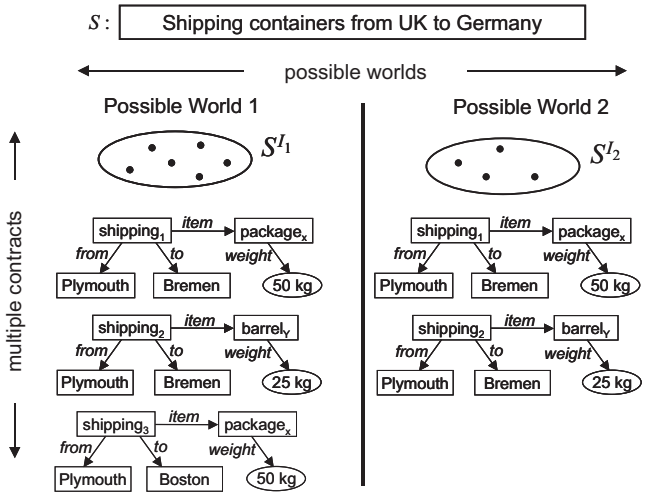

Fig. 3. Different Kinds of Variance

reflected by the two dimensions: the horizontal dimension represents resolution of incomplete knowledge, producing different possible worlds, and the vertical dimension represents intended diversity, producing alternative service instances within each possible world.

2.3 The Discovery Phase

Discovery is the task of locating service providers who meet a requestors needs. Discovery is performed by matching a service description of a requestor to the service descriptions of potential providers, in order to detect which of them are relevant. Following the intuition in [14, 11], two service descriptions match if there is a service instance acceptable for both descriptions. In this case, the service instance provides a basis for a business interaction between the provider and the requestor.

If the discovery phase returns a successful match between requestor and provider, it means that the two parties can potentially do business with each other. Before the business interaction is actually carried out, direct communication between the two parties may be necessary to further refine service parameters. This takes place during a pre-contractual phase, which may include negotiation [14, 11]. Hence, a match during discovery does not necessarily imply a successful business interaction; it only shows the potential for such an interaction. In this paper we focus on discovery only, leaving negotiation and pre-contractual interaction outside of our scope.

3 Using Logic to Operationalise Discovery

In this section we operationalise the intuition from Section 2 in the framework of description logics, and thus obtain a platform for service discovery.

3.1 From Intuition to Logic

To bridge the gap between the intuition and formal logic, we map the notions from Section 2 to description logic constructs in the following way:

- A service description maps to a set of DL-axioms $D = \{\phi_1, \phi_2, \ldots, \phi_n\}$. Some of the axioms ϕ_i in D impose restrictions on an atomic concept S, which represents the service to be performed.
- Domain-specific background knowledge maps to a knowledge base $\mathcal{KB}$ that contains all relevant domain-level facts.
- A possible world resolving incomplete knowledge issues in a particular way maps to a single DL model I of $\mathcal{KB} \cup D$.
- A service instance maps to a relational structure in I.
- The service instances that are acceptable w.r.t. a service description D map to the individuals in the extension S^I of the concept representing the service.
- Variance due to intended diversity is reflected by S^I containing different individuals.
- Variance due to incomplete knowledge is reflected by $\mathcal{KB} \cup D$ having several models $I_1, I_2, \ldots$.
- Matching a provider's service description D_p against a requestor's service description D_r w.r.t. a domain knowledge base $\mathcal{KB}$ is captured by a boolean function $match(\mathcal{KB}, D_r, D_p)$, which specifies how to apply DL inferences to perform matching.

In contrast to [4], our approach makes an explicit distinction between the service description D and the concept description S, as we allow the axioms ϕ_i in D to restrict concepts other than S. Furthermore, note that the concept S is equivalent to the unary predicate $S(x)$ in first-order logic.

3.2 Basic Elements for Modelling Service Semantics

The axioms in a service description D constrain the set of acceptable service instances in S^I. These restrictions usually constrain various properties of a service instance. We now analyse the various ways in which a property can be constrained, and show how to express this in DL.

- **Variety**
  A property can either be restricted to a *fixed* value or it can *range* over instances of a certain class. This can be expressed with qualifying DL concept constructors, such as $\forall\, r.\{i\}$ and $\forall\, r.C$, respectively. For any acceptable service instance, the value of such a property must either be a certain individual or a member of a certain class.

– **Availability**
  A property can either be *obligatory*, requiring all acceptable service instances to have a value for it, or *optional*, allowing service instances without a property value. By using existential quantification of the form $\exists\, r.\top$ service instances are required to have a value for r.
– **Multiplicity**
  A property can either be *multi-valued*, allowing service instances with several different property values, or *single-valued*, requiring service instances to have at most one value for the property. By the number restriction $\leq 1\, r$, a property can be marked as single-valued.
– **Coverage**
  A property can be explicitly known to cover a range. If it is *range-covering*, the service description enforces that in every possible world, for any value in the range, there is an acceptable service instance with this property value. This introduces variance due to intended diversity. This can be expressed by including an additional axiom of the form $C \sqsubseteq \exists\, r^{-}.S$ in D, where the concept C is the range of the property r to be covered[7]. Conversely, a non-range-covering property induces variance due to incomplete knowledge, as in different possible worlds different subsets of the range will be covered.

We illustrate different aspects of property restrictions by the following example. It consists of a requestor's service description D_r, and two service descriptions D_{p_A} and D_{p_B} specified by potential providers A and B, as well as a domain knowledge base KB.

$$
\begin{aligned}
D_r = \{\ & S_r \equiv Shipping \sqcap \exists\, from.\{Plymouth, Dover, Dublin\} \sqcap \\
& \quad \exists\, to.\{Bremen\} \sqcap \forall\, payment.EPayment \sqcap \\
& \quad \exists\, item.(Package \sqcap \forall\, weight. =_{18}) , \\
& Shipping \sqsubseteq = 1\, item , \\
& \{Plymouth, Dover, Dublin\} \sqsubseteq \exists\, from^{-}.S_r , \\
& EPayment \sqsubseteq \exists\, payment^{-}.S_r \qquad \}
\end{aligned}
$$

$$
\begin{aligned}
D_{p_A} = \{\ & S_{p_A} \equiv Transportation \sqcap \exists\, from.UKCity \sqcap \\
& \quad \exists\, to.GermanCity \sqcap \exists\, payment.CreditCard \sqcap \\
& \quad \forall\, item.(\forall\, weight. \leq_{20}) , \\
& UKCity \sqsubseteq \exists\, from^{-}.S_{p_A} , \\
& GermanCity \sqsubseteq \exists\, to^{-}.S_{p_A} , \\
& CreditCard \sqsubseteq \exists\, payment^{-}.S_{p_A} \qquad \}
\end{aligned}
$$

$$
\begin{aligned}
D_{p_B} = \{\ & S_{p_B} \equiv Shipping \sqcap \exists\, from.USCity \sqcap \\
& \quad \exists\, to.USCity \sqcap \exists\, payment.CreditCard \sqcap \\
& \quad \forall\, item.Container , \\
& USCity \sqsubseteq \exists\, from^{-}.S_{p_B} , \\
& USCity \sqsubseteq \exists\, to^{-}.S_{p_B} , \\
& CreditCard \sqsubseteq \exists\, payment^{-}.S_{p_B} \qquad \}
\end{aligned}
$$

[7] This is obtained by transforming the axiom $\forall x : C(x) \rightarrow \exists y : [r(y, x) \wedge S(y)]$ into description logic by standard manipulation of first-order formulae.

$$KB = \{ \begin{array}{ll} Package \sqsubseteq Container\,, & Plymouth : UKCity\,, \\ Shipping \sqsubseteq Transportation\,, & Dover : UKCity\,, \\ & Bremen : GermanCity \} \end{array}$$

Through the description D_r, a requestor asks for a service capable of shipping a package weighing 18 kg from Plymouth, Dover or Dublin to Bremen, accepting any kind of electronic payment. Provider A offers transportation for freights weighing at most 20 kg from cities in the UK to cities in Germany and requires payment by credit card. Provider B offers a similar kind of service between cities in the USA, with less restrictions on the items to be shipped. We now discuss different types of property restrictions employed in this example.

Variety: The requestor restricts the *to* property to a fixed value, i.e. it requires the destination to be *Bremen*. In the descriptions of the providers the same property ranges over classes. For the origin location, the requestor restricts the range of the property *from* to a set of explicitly specified values.

Availability: The providers make the *payment* property obligatory by existential quantification. The requestor does not necessarily require payment specification in the service instance; however, if payment is specified, it must be a kind of electronic payment.

Multiplicity: In most cases, it makes sense to declare properties as single-valued by functional constraints in the domain ontology. Sometimes there are natural multi-valued properties, such as *item*. In the example, the requestor imposes an additional constraint making *item* single-valued, specifying that he wants to ship only one item.

Coverage: The origin and destination locations of the providers cover a range. For example, by an additional axiom in D_{p_A}, the provider A requires any UK city to occur as a value of the property *from* in some service instance. In this way the provider explicitly states that they support shipping from all UK cities[8].

4 Matching Service Descriptions

We now discuss the application of DL inferences to perform service discovery. To better understand the semantics of the inferences we start by explaining how variance, introduced in Section 2, influences the matching process.

4.1 Treating Variance due to Incomplete Knowledge

There are two possible ways to deal with variance due to incomplete knowledge in the matching process. The first is to ask if there is a way of resolving unspecified issues such that the two service descriptions accept a common service instance. Let α denote the actual formula used to check whether two service descriptions

[8] In Section 5.1 we show that such an axiom for specifying range-coverage is insufficient when there are several range-covering properties in one service description.

match. Then the above question may be answered by checking the satisfiability of $KB \cup D_p \cup D_r \cup \{\alpha\}$, i.e. whether $KB \cup D_p \cup D_r \cup \{\alpha\}$ has a model. In other words, we check whether D_r and D_p specify a common service instance in at least one possible world.

The second possibility is to ask whether the requestor and provider have common service instances, regardless of how unspecified issues are resolved. This is equivalent to checking whether $KB \cup D_p \cup D_r \models \alpha$, i.e whether α is satisfied in every model of $KB \cup D_p \cup D_r$. In other words, we check whether D_r and D_p specify a common service instance in each possible world.

4.2 Treating Variance due to Intended Diversity

There are three ways of checking whether intended diversity allows a common service instance, each resulting in a different matching formula α. The first possibility is to ask if there is a service instance acceptable to both the requestor and the provider. This amounts to checking whether the intersection ${S_r}^I \cap {S_p}^I$ is non-empty, and can be performed by $\alpha = S_r \sqcap S_p \not\sqsubseteq \bot$.

Two other possibilities are to check whether the service description of the requestor is either more specific or more general than the one of the provider. This amounts to verifying whether ${S_r}^I \subseteq {S_p}^I$ or ${S_p}^I \subseteq {S_r}^I$, and can be performed by $\alpha = S_r \sqsubseteq S_p$ or $\alpha = S_p \sqsubseteq S_r$ respectively.

4.3 Inferences for Discovery Matching

Substituting the different definitions of α into the two approaches applying α results in several possible inferences. We discuss some of these alternatives on a simplified version of the example from Section 3.

$$D_r = \{ \quad S_r \equiv Shipping \sqcap \exists\, from.\{Plymouth, Dublin\}\ , \\ Plymouth : \exists\, from^{-}.S_r\ , \\ Dublin : \exists\, from^{-}.S_r \qquad \}$$

$$D_{p_A} = \{ \quad S_{p_A} \equiv Shipping \sqcap \exists\, from.UKCity\ , \\ UKCity \sqsubseteq \exists\, from^{-}.S_{p_A} \qquad \}$$

$$D_{p_B} = \{ \quad S_{p_B} \equiv Shipping \sqcap \exists\, from.USCity\ , \\ USCity \sqsubseteq \exists\, from^{-}.S_{p_B} \qquad \}$$

$$KB = \{ \quad Plymouth : UKCity\ , \quad Shipping \sqsubseteq = 1\ from \ \}$$

For each different inference we provide formalisations in first order logic and in description logic. The latter can be performed by any description logic reasoner capable of checking knowledge base satisfiability.

Satisfiability of Concept Conjunction We begin with the inference that follows directly from the intuition of a non-empty intersection of the sets of service instances associated to the two service descriptions, as proposed in [2, 13, 14]. In the following, the symbol ι represents an individual not occurring in KB, D_r or D_p.

Inference:	*Satisfiability of Concept Conjunction*
Intuition:	Is there a way to resolve unspecified issues such that D_r and D_p specify some common service instance?
Formula:	$KB \cup D_r \cup D_p \cup \{\exists x : S_r(x) \wedge S_p(x)\}$ is consistent $\Leftrightarrow$ $KB \cup D_r \cup D_p \cup \{\iota : (S_r \sqcap S_p)\}$ is satisfiable

This is the weakest check with respect to both kinds of variance. Along the dimension of intended diversity, it is sufficient to find one common service instance. Along the dimension of incomplete knowledge, it is sufficient to find one possible world in which such a service instance exists regardless of all other possible worlds.

Applied to the example above, $match(KB, D_r, D_{p_A})$ yields a positive result because the concept $S_r \sqcap S_{p_A}$ is satisfiable w.r.t. $KB \cup D_r \cup D_{p_A}$. This matches our intuition, since the restrictions of the two service descriptions are not contradictory. On the other hand, $match(KB, D_r, D_{p_B})$ also yields a positive result. At first glance, this looks counterintuitive and seems to be a false positive match since none of the cities involved are US cities. However, this does not contradict the intuitive meaning we gave to the inference, which is checking for an existence of a common service instance in one possible world. The concepts $UKCity$ and $USCity$ are not explicitly defined as disjoint. Hence, in some models of $KB \cup D_r \cup D_{p_B}$ they can have a common instance. To remedy this problem, modelers should impose additional constraints to reduce variance due to incomplete knowledge. Unfortunately, axioms such as $UKCity \sqcap USCity \sqsubseteq \bot$ are often not found in domain ontologies, thus preventing the effective use of this inference.

Entailment of Concept Subsumption Another inference that has been applied to matchmaking for discovery in e.g. [8] or [7] is checking for subsumption, either of the requestor's description by the provider's or vice versa.

Inference:	*Entailment of Concept Subsumption (Specialised Request)*
Intuition:	Do the service instances of D_p encompass the service instances of D_r, regardless of how unspecified issues are resolved?
Formula:	$KB \cup D_r \cup D_p \models \forall x : S_r(x) \rightarrow S_p(x)$ $\Leftrightarrow$ $KB \cup D_r \cup D_p \cup \{\iota : (S_r \sqcap \neg S_p)\}$ is unsatisfiable

Inference:	*Entailment of Concept Subsumption (Generalised Request)*
Intuition:	Do the service instances of D_r encompass the service instances of D_p, regardless of how unspecified issues are resolved?
Formula:	$KB \cup D_r \cup D_p \models \forall x : S_p(x) \rightarrow S_r(x)$ $\Leftrightarrow$ $KB \cup D_r \cup D_p \cup \{\iota : (S_p \sqcap \neg S_r)\}$ is unsatisfiable

In contrast to *Satisfiability of Concept Conjunction* this check is very strong, since it requires one of the service descriptions to be more specific than the other for all service instances in all possible worlds. In [8] and [4], specialisation has been denoted as PlugIn-match and generalisation as Subsumes-match. Neither of the directions in which subsumption can be applied works in our example, i.e. neither $match(KB, D_r, D_{p_A})$ nor $match(KB, D_{p_A}, D_r)$ yields a positive result. This is because neither of the two service descriptions is more specific or more general than the other. The requestor accepts shipping from Plymouth or Dublin and the provider accepts shipping from UK cities. Since Dublin is not in the UK, neither of the two associated sets of service instances fully contains the other in every possible world.

Entailment of Concept Non-Disjointness To overcome the deficiencies of the above matching formulas we propose the following one, which exploits incomplete knowledge in service descriptions without being too weak.

Inference:	*Entailment of Concept Non-Disjointness*
Intuition:	Do D_r and D_p specify some common service instance, regardless of how unspecified issues are resolved?
Formula:	$KB \cup D_r \cup D_p \models \exists x : S_r(x) \wedge S_p(x)$ $\Leftrightarrow$ $KB \cup D_r \cup D_p \cup \{S_r \sqcap S_p \sqsubseteq \bot\}$ is unsatisfiable

This check is stronger than *Satisfiability of Concept Conjunction* because it checks for an intersection in every possible world. However, it is not as strong as *Entailment of Concept Subsumption* because it does not require one of the sets of acceptable service instances to be fully contained in the other set. In our example, $match(KB, D_r, D_{p_A})$ yields a positive result whereas $match(KB, D_r, D_{p_B})$ does not, thus matching our intuition. By specifying that the property *from* is range-covering, provider A ensures that there is a service instance for shipping from each UK city in every possible world. Thus a service instance for shipping from Plymouth yields a match since Plymouth is a UK city in each possible world. Furthermore there is at least one possible world in which Plymouth is not a US city, since it has not been explicitly specified as such. Therefore the service description of provider B does not match with the service description of the requestor.

5 Applying Matching to Discovery

Having provided an intuitive and formal framework for describing service semantics, we now discuss its practical usage.

5.1 Practicality of Various Inferences

Satisfiability of Concept Conjunction is a very weak check which will often yield false positive matches, mainly due to incompleteness in domain ontologies and service descriptions. However, it can still be used as a first check to filter out incompatible service descriptions which cannot have a common service instance under any circumstances.

Entailment of Concept Non-Disjointness requires modelers to be careful and to include all important information in each possible world. This can be achieved, for example, by range-covering property restrictions. Unfortunately, expressing range-coverage using DL axioms, as described in Section 3, lacks expressiveness if several properties should simultaneously cover a range. Applied to the example in Section 3.2, $match(KB, D_r, D_{p_A})$ does not yield a positive result. Namely, properties *from* and *to* occurring in D_{p_A} are range-covering, but this is stated separately for each property: D_{p_A} does not require all combinations of possible property values to be covered. To express the coverage of a range $C_1 \times C_2$ for two properties at once, one needs the formula $\forall x_1, x_2 : C_1(x_1) \wedge C_2(x_2) \rightarrow \exists y : [r_1(y, x_1) \wedge r_2(y, x_2) \wedge S(y)]$. Unfortunately this formula cannot be translated to DL as it is no longer in the two-variable fragment of first-order predicate logic. We are currently investigating whether the combination of DL with rules, as suggested in [6], can provide a solution to this problem.

From the discussion in Section 4, we argue that subsumption is too strong for the general case. In the following subsection we propose a subsumption-based ranking mechanism that supersedes *Entailment of Concept Subsumption* as an inference for discovery.

5.2 Ranking Service Descriptions

Discovery may result in a service requestor locating several possible service providers, each able to provide a different set of service instances, with different service parameters. The requestor (and providers) may have *preferences* over the choice of parameter; for example, the requestor may prefer to ship from Plymouth, though other ports are possible too. However, preference information is often sensitive and is therefore rarely revealed explicitly [12]. For this reason, a discovery framework should not require the modeler to reveal such information. It can be used implicitly in negotiations between two entities which have been put in contact with each other through the discovery process. We show that there is a way to rank potential service providers without revealing preference information by defining a partial order for service descriptions based on subsumption.

To see this in practice, let us return to the example from Section 3.2 and slightly modify it. A requestor formulates a service description D_r that accepts shipping from one of the cities $\{Plymouth, Portsmouth, Dover\}$. In any negotiation situation the requestor prefers *Plymouth* while *Portsmouth* is second choice. Now suppose provider X advertises the availability of $\{Plymouth, Dover\}$ as origins, provider Y advertises the availability of $\{Edinburgh, Dover\}$, and provider Z advertises the availability of $\{Portsmouth, Newcastle\}$. Even without any knowledge about the preferences of the requestor, discovery can provide some order on the matches as to which is potentially most useful. Of the alternatives offered by Y, *Edinburgh* is of no interest, leaving *Dover*. However, provider X offers *Plymouth* and *Dover*. Hence, provider X offers a superset of relevant options, so X is 'better' than Y. X also provides more alternatives than Z. However, the discovery mechanism has no access to the requestor's preferences, and *Portsmouth* could as well be the preferred option as could *Plymouth* or *Dover*. Hence, it can return a list of service providers $[X, Y, Z]$, together with a partial order $\{X \geq Y\}$ to the requestor. The requestor can then assess the maximal elements in the partial order (in this case, X and Z) and contact one or more of them.

This example shows that one provider is 'better' than another if the variance it provides, which is relevant to the requestor, is a superset of the other. We formalise this notion by saying that, for two service descriptions D_{p_1} and D_{p_2} specified by providers and a service description D_r specified by a requestor, $D_{p_1} \geq D_{p_2}$ w.r.t. D_r if the following inference yields a positive result.

Inference:	*Entailment of Partial Concept Subsumption*
Intuition:	Do the service instances common to D_r and D_{p_1} encompass the service instances common to D_r and D_{p_2}, regardless of how unspecified issues are resolved?
Formula:	$KB \cup D_r \cup D_{p_1} \cup D_{p_2} \models \forall x : S_r(x) \wedge S_{p_2}(x) \rightarrow S_r(x) \wedge S_{p_1}(x)$ $\Leftrightarrow$ $KB \cup D_r \cup D_{p_1} \cup D_{p_2} \cup \{\iota : (S_r \sqcap S_{p_2} \sqcap \neg S_{p_1})\}$ is unsatisfiable

This relationship defines a partial order on the set of all available service descriptions advertised by providers[9]. As the above example shows, there may be more than one maximal element and more than one minimal element in the set. In our example, supplier Z is both maximal and minimal.

An interesting property of this partial order is the following: if a description D_p is such that S_p subsumes S_r, then D_p is a maximal element.[10] Researchers have proposed both concept subsumption and concept intersection as alternative definitions of matching during discovery, and have argued that subsumption is preferable. Our partial order is a generalisation of this. A subsumption match will always be a maximal element in the partial order, and so will be preferable to all

[9] Technically, it defines a partial order on the equivalence classes of service descriptions defined by the equivalence relation $D_{p_1} \sim D_{p_2}$ iff $D_{p_1} \geq D_{p_2}$ and $D_{p_2} \geq D_{p_1}$.

[10] Furthermore, the equivalence class containing it is the top element of the partial order.

non-subsumption matches. However, even if no subsumption matches occur in a given set, we can still define the partial order and determine preferred matches.

6 Conclusion

In this paper, we have described service discovery based on semantic descriptions of services and DL reasoning. We have given formal service descriptions a clear intuitive semantics based on the notion of a service instance, whereas a service description constrains the set of allowed service instances. In this context we have identified two different kinds of variance that arise in service modelling. We have shown how variance due to incomplete knowledge can be modelled by different possible worlds for resolving unspecified issues, and how variance due to intended diversity can be modelled by several service instances in one possible world.

We have mapped these intuitive notions into description logics, thus allowing the discovery framework to be realised with Semantic Web standards, such as OWL-DL. We have introduced a set of attributes by which restrictions on a service can be characterised on an intuitive level. We have further shown how such restrictions can be expressed in description logics.

Based on different ways of handling variance during the matching process, we have discussed several inferences that can be applied to perform discovery. In particular, we have proposed *Entailment of Concept Non-Disjointness* as a new inference to overcome some deficiencies of inferences proposed in the literature. We have identified some problems with inferences based on subsumption reasoning and satisfiability of concept conjunction, that had not been outlined before. For each of the inferences, we have based the semantics of matching on our intuitive notions of service instances and variance in order to provide a thorough understanding of the matching process. Furthermore, we have proposed a new way of ranking service providers, which generalises existing proposals in the literature.

In future, we shall expand and round up the set of primitives necessary for modelling service descriptions. Our final goal is to provide modelers with a set of intuitive modelling primitives with a mapping to a well-defined formalisation. We hope that such an approach will allow domain experts to model services, without requiring an in-depth knowledge of logic. Finally, we wish to investigate different interpretations of variance in service instances, and explore ways to capture these within a single modelling framework.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, January 2003.
2. J. Gonzales-Castillo, D. Trastour, and C. Bartolini. Description Logics for Matchmaking of Services. In *Proc. of the KI-2001 Workshop on Applications of Description Logics*, volume 44, 2001.

3. O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification, http://www.w3.org/TR/REC-rdf-syntax/.
4. L. Li and I. Horrocks. A Software Framework For Matchmaking Based on Semantic Web Technology. In *Proc. of the Twelfth World Wide Web Conference*, 2003.
5. S. McIlraith and D. Martin. Bringing Semantics to Web Services. *IEEE Intelligent Systems*, 18(1):90–93, 2003.
6. B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. *To appear in Proc. of the 3rd Intern. Semantic Web Conf. (ISWC)*, 2004.
7. T.D. Noia, E.D. Sciascio, F.M. Donini, and M. Mogiello. A System for Principled Matchmaking in an Electronic Marketplace. *Journal of Electronic Commerce*, 2004.
8. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic mathcing of web service capabilities. In *Proc. of the 1st Intern. Semantic Web Conf. (ISWC)*, pages 333–347, 2002.
9. M. Paolucci and K. Sycara. Autonomous Semantic Web Services. *IEEE Internet Computing*, 7(5):34–41, 2003.
10. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object Exchange across Heterogeneous Information Sources. In *11th Conf. on Data Engineering*, pages 251–260, Taipei, Taiwan, 1995. IEEE Computer Society.
11. Chris Preist. A Conceptual Architecture for Semantic Web Services. *To appear in Proc. of the 3rd Intern. Semantic Web Conf. (ISWC)*, 2004.
12. K. Sycara, M. Paolucci, J. Soudry, and N. Srinivasan. Dynamic discovery and coordination of agent-based semantic web services, 2004.
13. D. Trastour, C. Bartolini, and J. Gonzales-Castillo. A Semantic Web approach to service description for Matchmaking of Services. In *Proc. of the First Semantic Web Working Symposium*, 2001.
14. D. Trastour, C. Bartolini, and C. Preist. Semantic web support for the business-to-business e-commerce lifecycle. In *Proc. of the Eleventh Intern. Conf. on World Wide Web*, pages 89–98, 2002.
15. H. Akkermans Z. Baida, J. Gordijn. A Shared Service Terminology for Online Service Provisioning. *To appear in Proc. of the Sixth Intern. Conf. on Electronic Commerce (ICEC04), Delft*, 2004.