9

Integration and retrieval systems

Summary. The goal of this chapter is to give evidence for the practical applicability of the models and methods presented. After having proposed a logical framework and an architecture for representing information semantics as well as the possibility to generate metadata based on this framework and methods to reason about information contents, we now present existing systems that implement some of the methods discussed. We focus on these methods and explain the specific implementation using a common example.

In this section we will discuss some existing systems for retrieving and integrating information on the Web. Rather than giving an overview of the variety of systems available, we select three systems that address the issues discussed in the last chapters. We start our discussion with the OntoBroker system, which implements the basic functionality of a single ontology information integration and retrieval system. Further OntoBroker provides support for rule-based context transformation. As a second system, we look at OBSERVER, a multiple-ontology system. We will focus on the use of more than one ontology in the system and describe how ontologies are integrated in OBSERVER. Further, OBSERVER uses a query re-writing technique to translate between different ontologies that is based on the same ideas as the approach discussed in Chap. 6. Finally, we turn our attention to the BUSTER system, which uses the hybrid approach. Here we focus on the use of the shared terminology in query formulation and processing. Further, the BUSTER system implements functionality for querying spatially related information similar to the ideas described in Sect. 8.3. We describe these techniques and their use in information retrieval.

In order to give a better impression of the systems, their differences and similarities, we use a simple example from the travel domain and describe how the systems solve this specific integration problem. The task of the example

is to retrieve hotels with a room rate that is under a certain threshold from different information sources with accommodations. Table 9.1 shows the part of the available information we will focus on.

Table 9.1. Data from the example problem

Name	Location	Category	Price
Radisson	Copenhagen	Congress Hotel	580
Mercure	Hamburg	Four star	190
Ritz	London	First Class	130
...	...	...	...

This small set of information already contains a number of very relevant integration problems that arise in many practical applications. First of all, we have to decide, whether all of information items are actually representing hotels. This is a problem in particular if the categories mentioned in the table are defined in different ontologies. As the hotels are in different countries, the room rates are given in different currencies that have to be normalized and finally the questions of spatial relevance with respect to the users needs arises. In the following we will see that the different systems differ in the way they focus on a specific problem.

In the following, we first discuss the use of the Ontobroker system that uses a global ontology and flexible transformation functions for comparing hotel types and prices. In the following session, the use of multiple ontologies in the OBSERVER system is presented. We show how OBSERVER uses semantic relationships between classes from different ontologies to compare data in the different sources and to select an optimal translation. Finally, we discuss the BUSTER system and explain the use of a shared vocabulary for describing features of accommodations as well as the determination of spatial relevance as a part of the information sharing process.

9.1 OntoBroker

The OntoBroker system [Decker et al., 1999] has been developed for supporting the access to distributed sources of digital information such as document repositories or Web sites. OntoBroker mediates between the different formats and structures that might be present in these sources by encoding the available information in a pivot format and relating it to a domain ontology that is shared across all sources. Consequently, the domain ontology is the central part of the OntoBroker architecture. As a successful use of OntoBroker relies on the existence of the shared ontology, OntoBroker comes with an editor

that supports the creation of domain-specific ontologies [Sure et al., 2002]. In order to link information to the ontology, available information items have to be modelled as instances of the ontology. In the case of well-structured information sources such as databases and spreadsheets, this step is done using specialized wrappers that extract information from the sources and assign it to classes and relations in the ontology. For less structured information like text documents and web pages, OntoBroker relies on an annotation tool that supports the user in adding special markup to the available information, thereby explicitly linking it to the ontology [Staab et al., 2001].

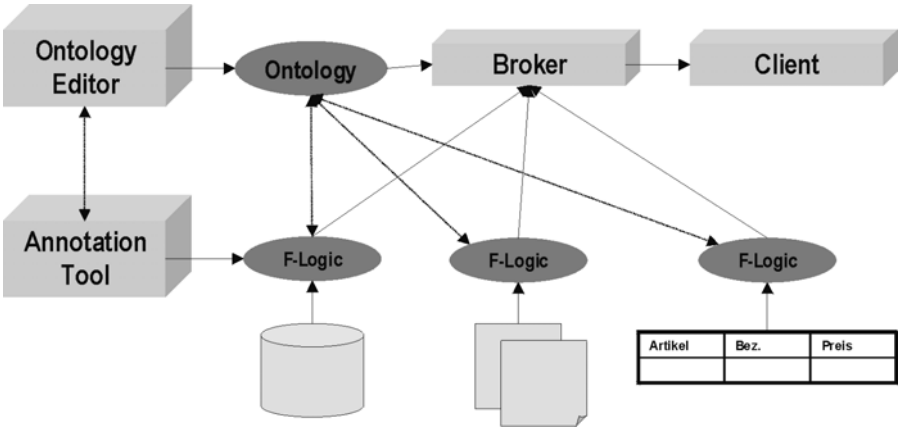


Fig. 9.1. The general architecture of the OntoBroker system

The ontology together with the instance information extracted from the different information sources behave like a deductive database. The actual broker system provides the corresponding reasoning facilities in terms of providing answers to complex queries concerning information items, their properties and relations. The broker makes this query-answering functionality available to client applications which may be rather generic query interfaces for arbitrary information or specialized applications relying on a specific domain ontology and a fixed set of information sources. Fig. 9.1 gives an overview of the OntoBroker architecture and its different components. In the following, we will have a closer look at the way OntoBroker represents information and ontologies and the use of rules for implementing functional context transformation.

9.1.1 F-Logic and its relation to OWL

The representation formalism for ontologies and information used in OntoBroker is F-Logic. Unlike the Web Ontology Language that us based on Descrip-

tion Logics, F-logic has its foundation in logic programming languages. More specifically, F-logic extends horn-logic language with constructs of frame languages supporting the straightforward representation of class-based knowledge representation. The OntoBroker inference engine translates these constructs back into horn logic and uses standard logic programming techniques for answering queries. The use of horn logic has some implications for the expressive power of F-logic as compared to OWL that we will briefly summarize in the following.

Correspondences with OWL

Focusing on a frame-based representation of knowledge, F-logic has a number of commonalities with OWL, in particular with OWL Lite. We summarize these common features in the following.

- *Classes.* F-logic can be used to express class membership and subclassing. Assigning an instance I to a class C is denoted as `I:C`, which corresponds to the OWL expression `Instance(I type(C))`. Further, a class C can be declared to be a subclass of another class D using the axiom `C::D`. This corresponds to the OWL expression `SubClassOf(C D)`.

- *Range Restrictions.* F-Logic can express a special type of property restriction. In particular, we can restrict the types of values that are allowed to be in the range of an attribute A that is assigned to a particular class C to some other class D. The corresponding F-Logic expression `C[A =>> D]` has the same effect as the OWL statement `SubClassOf(C restriction(A allValuesFrom(D)))`.

- *Facts.* Finally, we can express information about actual instances of an ontology in a frame-like fashion. For stating that an object O is of type C and shows a value V in the attribute A we write `O:C[A->V]`. In OWL we would express the same information using the following statement: `Instance(O type(C) value(A V))`.

We see that the direct overlap between F-Logic and OWL is a fragment of OWL that allows us to state simple schema information. In particular, the fragment always exactly corresponds to the OWL part of RDF Schema. We will see, however, that F-logic offers other means for defining the meaning of information mainly in terms of its rule language. In contrast to the original proposal , the variant on F-Logic implemented in Ontobroker is based on a semantics that borrows from logic programming rather than the standard semantics of first order logic. In fact, the DLP fragment discussed briefly in Chap. 3 also corresponds to the largest segment on which the different semantics of OWL and F-Logic intersect, adding further interest to this fragment.

Differences from OWL

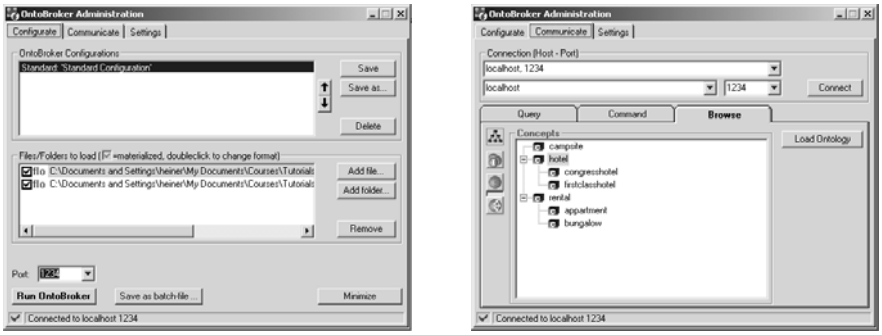
Being based on logic programming rather than description logics, F-Logic offers some features that go beyond the expressive power of OWL. These additional features can be used to capture some of the build in modelling primitives of OWL using special axioms.

- *General relations.* An often criticized limitation of OWL is the restriction to binary relations between objects. F-logic does not have this restriction and is able of representing predicates of arbitrary arity for capturing complex relations between multiple objects.
- *Parameterized attributes.* A special case of the use of general relations in F-Logic is the ability to parameterize the attributes of a class. We denote that V is the value of an attribute A of object O with respect to a certain parameter P (e.g. the length in inches) as `O[A@(P) ->> V]`. This feature is useful for describing different scales and measures.
- *Rules and queries.* The main difference between OWL and F-Logic is the axiom and rule language. F-Logic offers the possibility to state general implication axioms that act as rules and queries. The general form of a rule is `FORALL V, H <- B`, where V is a list of goal variables, H is the head and B the body of the rule. The body of a rule consists of an arbitrary F-Logic formula containing the operators `AND`, `OR`, `NOT`, `<-`, `->` or `<->` and quantifiers `FORALL` and `EXISTS`. Queries are rules with an empty head.

F-Logic rules can be used to model OWL features like the disjointness of classes, transitivity and inverses of relations and others. Beyond that, rules provide a powerful mechanism for encoding other features such as relational algebra and domain-specific knowledge about relations. In the following we will see how this can be used to mediate between different information sources in our example problem.

9.1.2 Ontologies, sources and queries

The OntoBroker strategy of using F-logic as a uniform language for information items and ontological background knowledge leads to a very flexible way of managing knowledge and information sources in the system. In particular, arbitrary F-logic files can be loaded into the OntoBroker system regardless of whether they contain information, ontological information or both. The corresponding interface of the standard client is shown in Fig. 9.2a. Here the user can also choose to compile out the rule base in order to increase run-time performance. The specifications in the different files loaded to the system are treated as one big knowledge base that can be used to answer queries about information and background knowledge. In particular, all schema information is considered as representing one ontology common to all information sources. Fig. 9.2b shows the ontology interface of the OntoBroker client that allows



(a) Source management (b) Ontology browser

Fig. 9.2. The OntoBroker client

the user to browse the ontological knowledge of the system.

Applying OntoBroker to our example problem first of all requires us to wrap the different information sources into a common F-logic representation and load the corresponding knowledge to the system. Each row in Table 9.1 is translated into an object with certain values for the relevant attributes corresponding to the columns of the table. The F-logic representations of the three example entries in the table are the following:

```
radisson:congresshotel[location->denmark; price->580].
mercure:four-star[location->germany; price->210].
ritz:first-class[location->england; price->130].
```

The first heterogeneity problem mentioned in the problem statement is the use of different categories of hotels. This problem can be addressed by a common ontology that relates the different types mentioned in the data to the more general concept hotel that can be used to query objects belonging to the different special types of hotels. The corresponding part of the F-logic definition is the following:

```
congresshotel::hotel
four-star::hotel
first-class::hotel
```

Using this ontology, we can query the system for all hotels that have a price of less than 200 using the following query:

```
FORALL Y,X <- Y:hotel[price->X] AND lessorequal(X,200).
```

The system returns the `ritz` as the answers to the query, because it can be shown to belong to the class of hotels and to have a price of less that 200. Fig. 9.3 summarizes the situation.

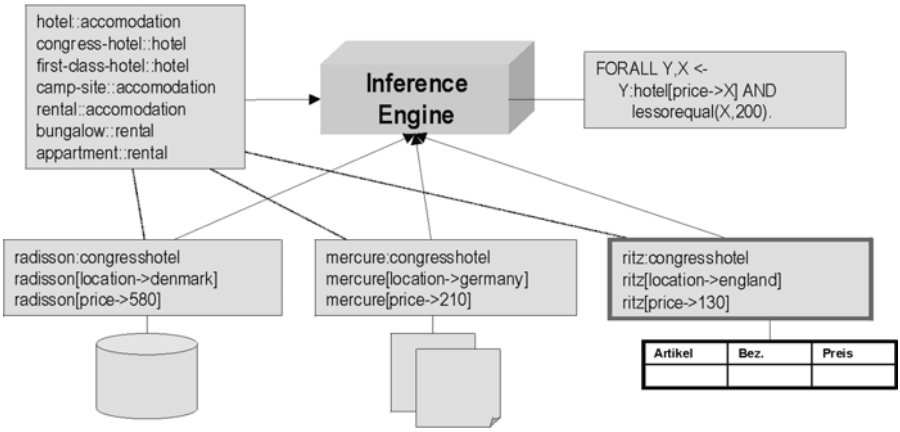


Fig. 9.3. Example of ontology-based retrieval

Using the rule language allows us to retrieve objects and their values based on complex criteria and background knowledge. In that OntoBroker behaves like a knowledge-based system capable of deriving new facts from given ones.

9.1.3 Context transformation

The use of an ontology of different types of hotels helps us to cope with the different hotel categories mentioned in the information sources. It cannot, however, solve the problem of differences in units and scales used in the description of attribute values. In our example, the prices for a room are given in different local currencies, which are Euros, UK Pounds and Danish Crowns. In order to make these prices comparable to each other and to the criteria given in the query, we have to normalize them to a common currency, say US Dollars. This problem corresponds to the notion of context transformation used before. In order to be able to transform from one context (in this case currency) to another, we first have to make the context of a piece of information explicit. Parameterized attributes are an elegant and flexible way of doing this. We therefore extend the description of information items by a currency parameter for the price attribute:

```
radisson:congresshotel[price@(dkcrowns)->580].
mercure:four-star[price@(euro)->210].
ritz:first-class[price@(ukpounds)->130].
```

The actual transformation between different contexts can be specified using complex F-logic rules that specify the value of an attribute in one context in terms of its value in other contexts. The translation can either be pointwise, from one specific context to another, or general. In the case of different

currencies, we can formulate a general rule for currency conversion that refers to an exchange rate.

```
FORALL X,Y,Z,A,B,C X[price@(A)->Y] <-
    X[price@(B)->Z] AND
    (Y is (Z*C)/100) AND
    A[exchangerate@(B)->C].
```

The rule above specifies a general transformation rule between arbitrary currencies by referring to currency objects that have the exchange rate to different other currencies as a parameterized attribute. When performing the transformation, the inference engine binds the object representing the goal currency to the variable in the rule, reads its exchange rate with respect to the currency mentioned in the description of the hotel and calculates the price in the goal currency, which is returned as the result. For the case of US Dollars, we use the following definitions of the currency object `usdollar`.

```
usdollar[exchangerate@(euro)->90].
usdollar[exchangerate@(ukpounds)->173].
usdollar[exchangerate@(dkcrowns)->27].
```

The currency transformation is now triggered by explicitly mentioning a goal currency in the query. In our case, we now look for hotels that have a price of less than 200 US Dollars:

```
FORALL Y,X <- Y:hotel[price@(usdollar)->X] AND lessorequal(X,200).
```

As summarized in Fig. 9.4, the result is no longer the `ritz`, but the two other hotels, because the price of 130 UK pounds corresponds a a much higher price in US Dollar while the prices denoted in Euro and Danish Crowns are actually lower than 200 if measured in Dollars.

The application of this kind of context transformation is of course not limited to measures and scales. We can also formulate rules that establish between different classes of hotels (for example first class and four star hotels). These rules, however, will always depend on the specific domain. We will turn our attention to systems that offer generic solutions for translating between different classifications in the following section.

9.2 OBSERVER

The OntoBroker approach described above can be seen as a good example of the core functionality an ontology-based information-integration system provides. In practice, however, some of the design decisions made for OntoBroker turn out to be unrealistic. The first is the restriction to a single ontology that covers all sources of data. As mentioned before, the restriction to a single ontology leads to significant maintenance problems when new

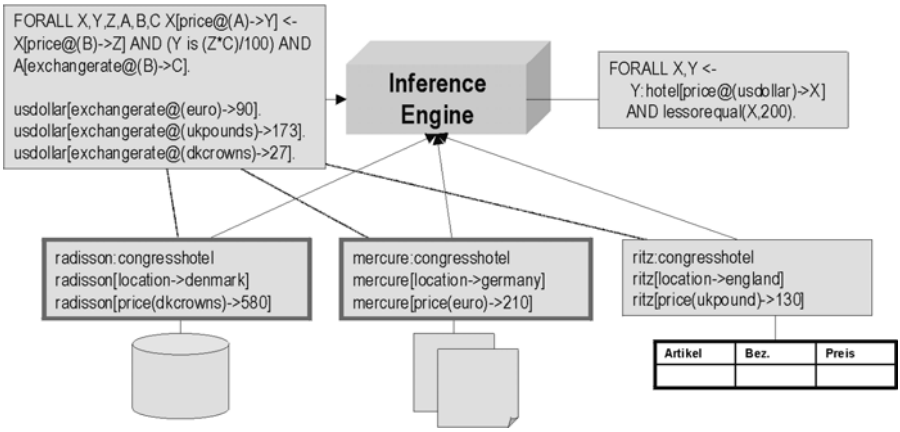


Fig. 9.4. Example context transformation

information sources are added. The other problematic aspect is the need to create and maintain a logical representation of information items as instances of the ontology. In the presence of large information sources the logical representation becomes the bottleneck of the system. In this section we will discuss the OBSERVER system, which provides solutions for the two problems mentioned above: the system allows the existence of multiple ontologies, including the use of different ontologies to represent different views on the same domain, and provides and uses the semantic information to generate plans of how to query the different sources rather than including individual information items into the reasoning process.

In the following, we describe how OBSERVER addresses the example integration problem focusing on these two aspects.

9.2.1 Query Processing in OBSERVER

The OBSERVER system implements a special query-processing strategy for dealing with multiple information sources that are based on different ontologies. This strategy consists of three basic steps shown in Fig. 9.5. The strategy is incremental in the sense that the system first tries to answer a query only using data that is linked to the user's ontology and establishes connection to other information sources one by one in case the user is not satisfied with the result so far. In the following, we briefly discuss the different steps shown in Fig. 9.5.

- *Query Formulation.* in the first step the user selects one of the existing ontologies in the system as source for the query vocabulary. In the following,

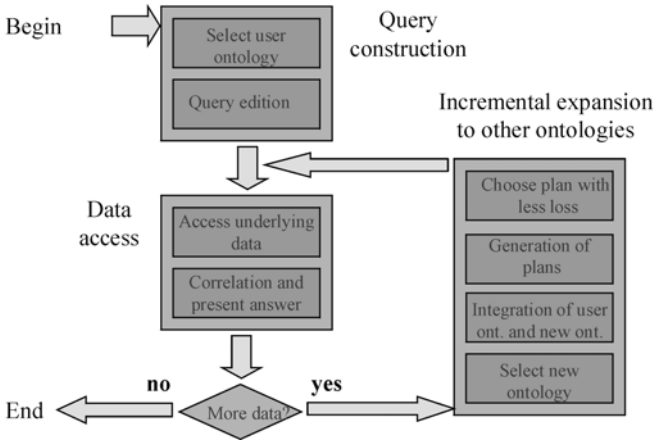


Fig. 9.5. Incremental query extension in OBSERVER

we call this ontology the user ontology. After having decided on a particular ontology, a query to the system can be formulated using the terms of the ontology that can be combined using operators of the CLASSIC description logic [Borgida et al., 1989].

- *Data access.* in OBSERVER an ontology is associated with a number of information sources. In this step, the system retrieves answers to the user query from the data sources associated with the ontology chosen in the first step. The user query is processed by expanding the query into an extended relational algebra expression. This expression is evaluated on the information source using special wrappers and the results are passed to the user as a partial answer to the query.
- *Query expansion.* if the user is not satisfied with the answer given by the system, the user query is incrementally expanded to other information sources. As these sources use different ontologies, the user query has to be re-written into the terminology used by the additional source (target ontology). For this purpose, OBSERVER uses semantic relations between the different ontologies in the system. These semantic relations include synonym, hypernym and hyponym relations as well as overlap, disjointness and coverage. These relations that are stored in a central repository can be interpreted as equivalence and subsumption in the description logic used to represent knowledge in the system. When re-formulating the query, OBSERVER distinguishes two cases:
  - In some cases, all terms in the query can be replaced by synonym terms in the target ontology. In this case, the re-formulated query is equivalent to the original one and there is no loss of information resulting from the translation (referred to as full translation).

– Often, not all the terms in a user query have synonym terms in the target ontology. In this case, OBSERVER performs a partial translation of these terms using a similar approach to the one described in Sect. 6.2. In particular, the terms are replaced by unions of hyponym or intersections of hyperym terms and the corresponding query is used to retrieve data accepting a certain loss of information. This case is referred to as partial translation.

In the case of a partial translation, there are different possibilities of combining replacements of terms in the query. For each of these possible translations, OBSERVER estimates the loss of information and selects the approach that can be assumed to have the smallest loss.

As shown in Fig. 9.5 steps two and three are repeated iteratively incorporating more and more information sources until the user is satisfied with the result. After the first iteration, the second step also includes a re-formulation of the retrieved information into the user ontology.

9.2.2 Vocabulary integration

We consider an extension of our example integration problem, where the information shown in Table 9.1 is taken from two different sources of information. Each of these information sources uses a different ontology that provides the terms used to describe the category of the accommodation. We assume that the information sources use the ontologies shown in Fig. 9.6. The hierarchy on the left-hand side is the user ontology that is used to formulate the query.

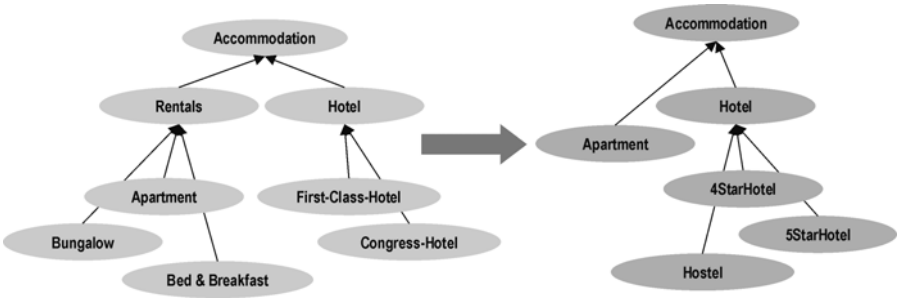


Fig. 9.6. The ontology integration problem

Using this ontology the user states the query for hotels with a price of less than 200. In the following, we focus our attention on the type information

contained in the query. The restriction on the price and the necessary currency conversion are assumed to be handled by information-source wrappers.

Looking at the information in Table 9.1, we see that consulting the information source associated with the user ontology will only produce the first item in the table as a result. The other two information items are classified according to the ontology on the right-hand side of Fig. 9.6. In order to decide whether this information is an answer to the query for hotels, the user ontology has to be integrated with this ontology using information about semantic relations between terms in the two models. We use the set of semantic relations shown in Table 9.2 to combine the two ontologies.

Table 9.2. Data from the example problem

IAO.APARTMENT	is a synonym of	SAO.apartment
IAO.HOTEL	is a hyponym of	SAO.hotel
IAO.HOTEL	is a hypernym of	SAO.4StarHotel
IAO.HOTEL	is a hypernym of	SAO.5StarsHotel
<IAO.HOTEL,80% >	overlaps	<SAO.Hotel,50%>
IAO.PRICE	is a synonym of	SAO.price

The semantic relations in Table 9.2 indicate that the terms apartment, hotel and price used in both models are synonyms. Further, we find the information that 4StarHotel and 5StarHotel are hyponyms of hotel and that the term hotel in the user ontology refers to a more specific concept than the term accommodation in the ontology of the additional information. In addition to the semantic relations, the system uses semantic descriptions of the different concepts in the ontologies. Consider the following definition of the term first class hotel in the user ontology:

```
(define-concept FirstClassHotel
    (AND Hotel
         (ALL Stars (> 3)))
```

It defines a first-class hotel to be a hotel with at least 4 stars. Using the semantic relations and the semantics of the description language, we can determine the relation of the term first class hotel to the terms 4 star hotel and 5 star hotel from the target ontology that have the following definition:

```
(define-concept 4StarHotel
    (AND Hotel
        (ALL Stars (= 4))))


(define-concept 5StarHotel
```

```
(AND Hotel
    (ALL Stars (= 5))))
```

Using a description-logic reasoner we can compute semantic relations between all the terms in the user and the target ontology. The resulting integrated model (Fig. 9.7) provides the basis for re-formulating user queries that do not have a full translation.

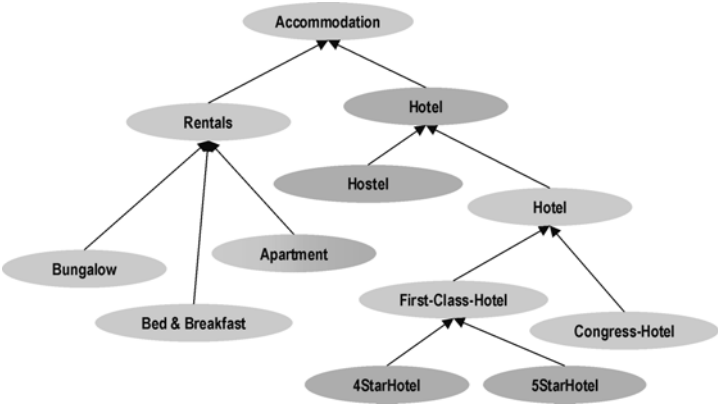


Fig. 9.7. Combined view on the ontologies

9.2.3 Query plan generation and selection

The integrated ontology now provides a basis for re-formulating queries across the different ontologies present in the system. While terms that have a direct synonym in the target ontology are just replaced by this synonym, terms without a direct correspondence have to be approximated by a combination of similar terms. This can be done in different ways, leading to a situation where one query can be translated in different ways. We will illustrate this using our example problem.

In order to answer our example query we will state a query in terms of a concept expression describing hotels with a price of less than 200. Using the logic provided in OBSERVER, this concept expression is the following:

```
(AND Hotel (ALL price (<200)))
```

Here, the terms *Hotel* and *price* are both taken from the user ontology. As we can see from the semantic relations, the term price has a corresponding term in the target ontology and can therefore be directly replaced. Although there is also a term “hotel” in the target ontology, it is not a synonym. We therefore

have to consider different approximation of this term. The approximation approach taken in OBSERVER is similar to the one described in Sect. 6.2. In particular, each term that does not have a synonym is either replaced by the conjunction of its parents or the union of its children in the integrated concept hierarchy. In our example this results in two possible translations of the query:

1. `(AND Hotel (ALL price (<200))`
2. `(AND (OR 4StarHotel 5StarHotel) (ALL price (<200)))`

The first alternative corresponds to using the upper, the second to using the lower approximation.

As OBSERVER allows us to use both kinds of approximations and even to mix them in cases where more than one term has to be approximated, we need a criterion to choose the best combination of these approximations. In OBSERVER this is done by estimating the loss of information for each possible translation based on statistics about the information sources. The notion of loss of information is based on upper and lower bounds on the expected precision and recall of a query, where precision and recall are defined in the usual way [Mena et al., 2000b]. As the following example shows, the estimation of loss of information is also useful in cases where we only replace one term as in the example above, as it helps us to choose between the upper and the lower approximations. Using the measures defined in [Mena et al., 2000b] we get the following figures:

Replacement	Hotel	(OR 4StarHotel 5StarHotel)
Precision	(23%, 30%)	(100%, 100%)
Recall	(100%, 100%)	(22%, 50%)
Loss	(53%, 62%)	(33%, 63%)

As the figures show, in contrast to our expectation, replacing the term hotel from the user ontology by the term hotel from the target ontology is not the best choice in our case. While it has about the same maximal possible loss of information, replacing the term by its lower approximation leads to a better lower bound in the loss (one-third as opposed to one-half of the information). In our example, OBSERVER will therefore decide to use the lower approximation for the term hotel and return all four and five star hotels from the information sources classified by the target ontology.

9.3 The BUSTER system

While the OBSERVER approach to information integration is quite similar to the techniques described in this part of the book (in particular Chap.

6), the need to create and maintain sematic relations between multiple ontologies in the system is a drawback. In order to cope with this problem OBSERVER is able to deduce new semantic relations from combinations of existing ones using canonical terms. This can be seen as a step in the direction of the hybrid integration approach mentioned in Chap. 2. The BUSTER system [Visser and Schuster, 2002], developed at the University of Bremen, is an example of a system that more explicitly uses the hybrid approach, implementing the methods described in this book. In the following we will briefly describe the BUSTER system focusing on those features distinguishing it from systems like OBSERVER, in particular the extensive use of a shared base vocabulary and methods for retrieval based on spatial criteria.

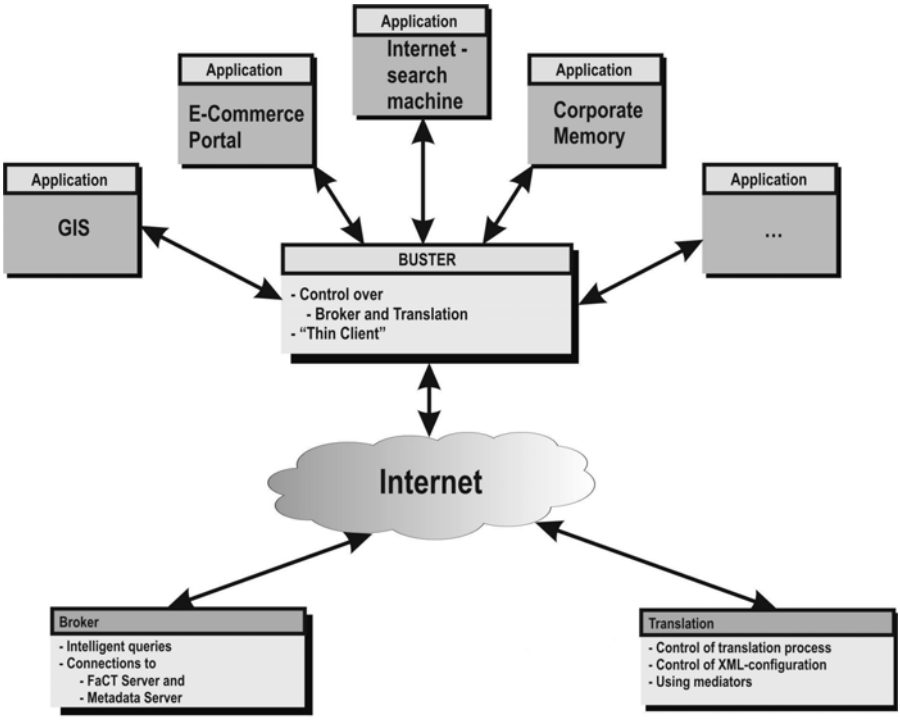


Fig. 9.8. BUSTER – Intelligent middleware for information sharing

BUSTER is meant to provide an intelligent middleware for information sharing. We envision that the BUSTER system is used by many different applications like search engines, e-commerce platforms or corporate memories in order to access heterogeneous and distributed information resources. For this purpose, the BUSTER system provides two subsystems, one for

information filtering and one for information integration. These subsystems are mainly independent of each other and can be accessed by clients over the World Wide Web (compare Fig. 9.8).

Fig. 9.9 shows the interaction of the two subsystems in a typical integration scenario. In a first step, relevant information sources are selected based on the user's information need. This is done by a broker component were information sources register and provide access information. The decision whether a source is relevant is based on source metadata provided by a metadata server. A user request is matched against the metadata of an information source that, same as the user query, is based on a shared vocabulary. The actual decision step uses an external OWL reasoner for deductive matching.

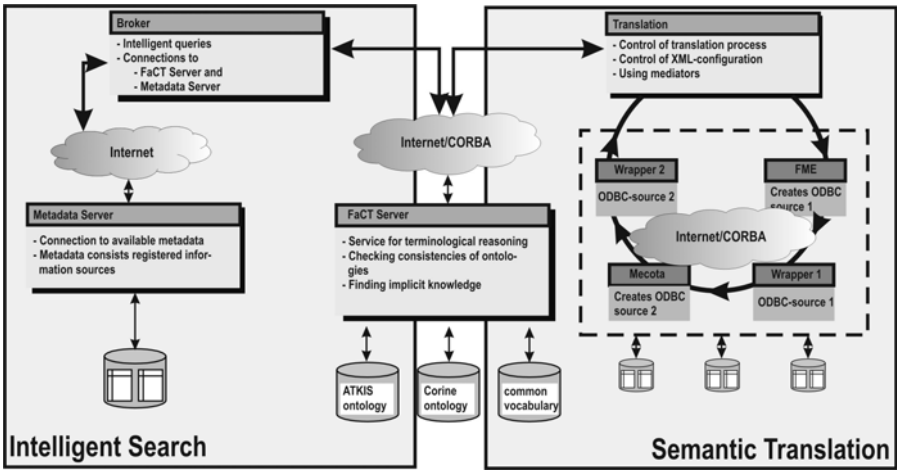


Fig. 9.9. Information Filtering and Integration in BUSTER [Neumann et al., 2001]

After an information source has been chosen, its content is translated into the user's format by the integration component. This structural and syntactic integration is performed by a classical mediator–wrapper architecture. The core of this part is the MECOTA system, a rule based mediator that uses abductive reasoning to translate between different information contexts [Wache, 1999, Wache, 2003].

9.3.1 The use of shared vocabularies

In principle, the query processing in BUSTER works quite similarly to the OBSERVER system: user queries are translated into the vocabulary provided by the ontologies assigned to different sources. The main difference lies in the fact that the user does not commit to a user ontology representing his

personal view of the domain but rather to a basic vocabulary that is used to define concepts in all the source ontologies (compare Sect. 2.4.2).

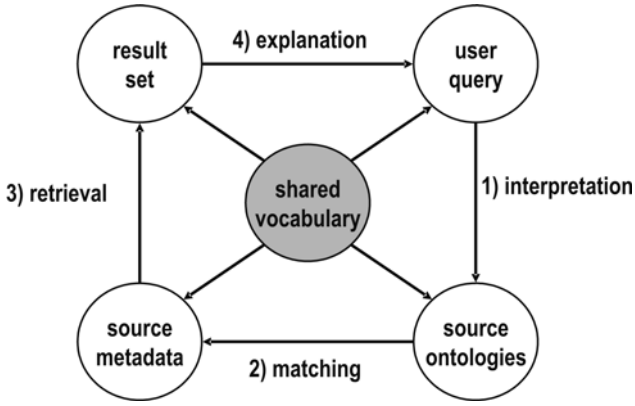


Fig. 9.10. The role of shared vocabularies in BUSTER

By formulating the user query in terms of this basic vocabulary we ensure that the query can interpreted with respect to all source ontologies in the system. In particular, we can determine these concepts in a source ontology that are most similar to the concept we asked for. Here, being similar means the direct parents and children of the query concept after we have classified it into the source ontology. After translating the query into the terminology provided by a source ontology – this is done as described in Sect. 6.2, we can match the query against metadata provided for the information source. The metadata for an information source is comparable to the descriptions used in the OntoBroker system. In particular, information items contained in the source and their properties are described using terms from the shared vocabulary. This again guarantees that we can determine those items that are an answer to the translated query by logical deduction. Finally, the descriptions of matching information items are returned to the user and a short explanation is given why the item has been matched – currently this explanation points to the concept in the source ontology that has been used for matching. Fig. 9.10 provides an overview of the process.

9.3.2 Retrieving accommodation information

For our example problem we use a shared vocabulary containing basic relations and terms from the accommodation domain that can be used to specify different types of accommodations as well as to describe actual accommodations. In the initial interaction with the system the user will be

asked to select a domain and the corresponding vocabulary (see Fig. 9.11).

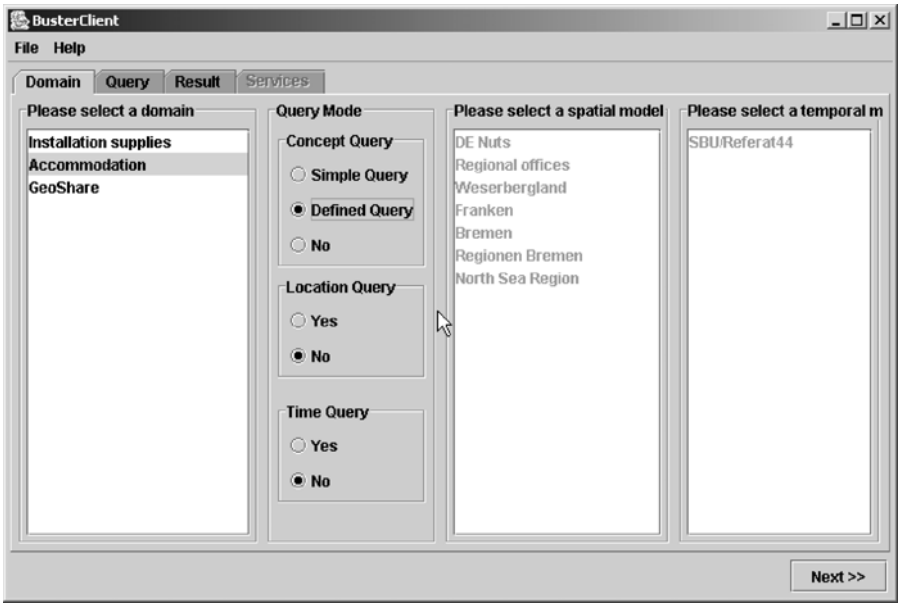


Fig. 9.11. Selection of shared models

After having selected a query the user is asked to formulate a query using the shared vocabulary. The query formulation is supported by a query-construction interface which is dynamically generated from the chosen vocabulary. As shown in Fig. 9.12 the vocabulary for the accommodation domain specifies five properties for the concept accommodation that can be further specified:

- *Meals.* Information about available meals for example used to distinguish full-pension, half-pension and bed and breakfast.
- *Facilities.* Information about available facilities including fixed installation such as TV-sets as well as services offered.
- *Stars.* Number of stars assigned to the accommodation. This can also be used describe other kinds of distinctions received by the accommodation.
- *Building.* Information about the type of building, e.g. apartments vs. one single complex.

Units: the size of the accommodation in terms of number of units.

For each of these properties, the vocabulary also defines a set or even a hierarchy of possible values. In Fig. 9.12 we see parts of the fillers for the facility property.

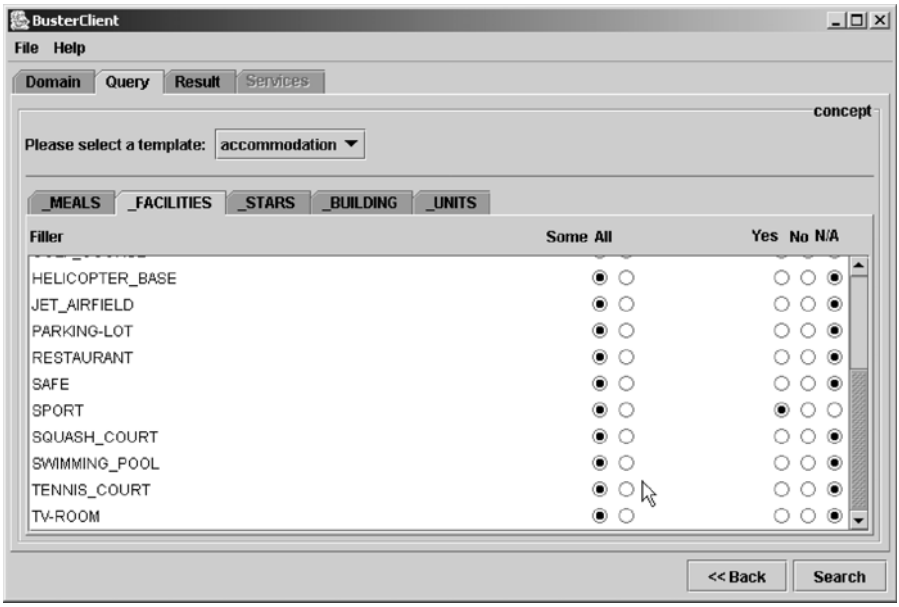


Fig. 9.12. Query construction based on shared terminology

Based on the query formulated by the user the system now searches the different information sources and returns relevant results. Fig. 9.13 shows a situation where the user is looking for a sports hotel. The results are shown as a list on the left-hand side of the screen. The right-hand side contains an explanation for the currently selected result. The explanation consists of the actual query being asked by the user, the matching concept from the source ontology – in this case "golf hotel" – and the metadata describing the result.

9.3.3 Spatial and temporal information

The systems we have discussed so far – and this observation also holds for information integration systems in general – are mainly focusing on the integration of conceptual information. We have seen how the systems deal with different classifications of information items and differences in underlying measures and scales. If we look at our example problem, however, it is quite obvious that there is not only a conceptual side to the integration problem. When looking for a certain accommodation, we also have to take care of spatial and temporal aspects of the information:

- Is the accommodation close enough to the place we really want to go to, e.g. the location of a conference?
- Is the accommodation available at the respective time we need to be at the place, e.g. the duration of the conference?

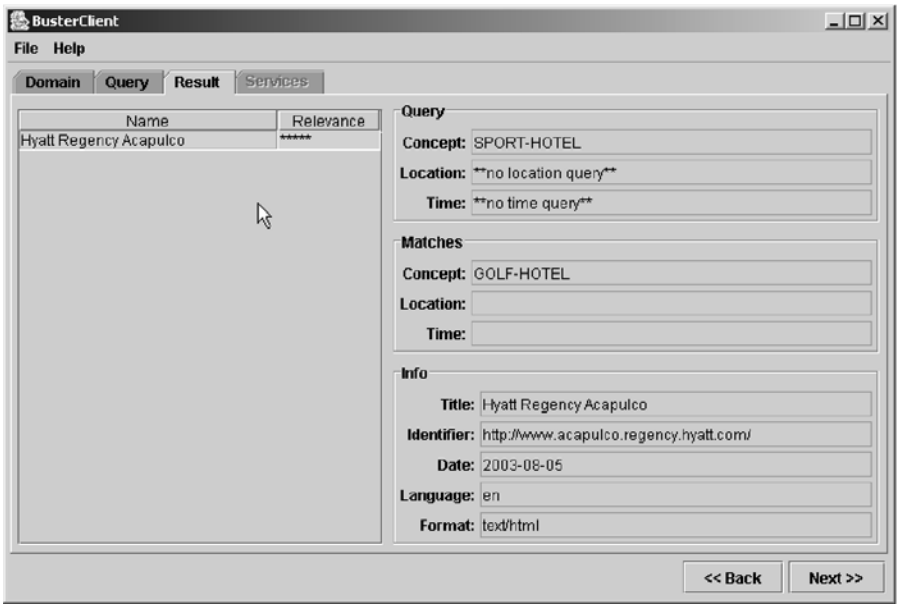


Fig. 9.13. Presentation and Explanation of the Result Set

Currently, these aspects are not well supported by many systems, because they require different kinds of reasoning mechanisms. While many systems rely on reasoning about class hierarchies, in particular about the subclass relation, reasoning about spatial relevance of a piece of information needs inferences over part-of hierarchies and neighborhood graphs. The BUSTER system tries to close this gap by distinguishing between conceptual, spatial and temporal aspects of an information request. Each of these components is evaluated separately and only information meeting all of the criteria is returned. In the following, we briefly describe the processing of spatial queries in the BUSTER system.

The idea behind processing spatial queries in BUSTER is the use of names of spatial locations. These names can include the names of cities and countries, but also less well-defined locations such as regions or landscapes. A problem that arises with the use of place names is the fact that different information sources will often use different place names. This might be due to the fact that the same place has different names (e.g. Chemnitz vs. Karl-Marx Stadt), to differences in the granularity of the information (countries vs. federal states) or the use of special names that do not have a counterpart in other terminologies (sales regions of a certain company). Fig. 9.14 shows the query interface of the system that allows the user to specify spatial criteria.

Fig. 9.14. Query interface for conceptual, spatial and temporal criteria

Once such a name appears in a user query, the system has to determine which part of the information that satisfies the conceptual part of the query is also relevant with respect to the place name in the query. In order to deal with this problem, the BUSTER system uses so called place-name structures [Voegele et al., 2003]. Place-name structures consist of a combination of a partonomy of spatial regions each connected with a name. In our example, this partonomy would for example contain the path: Europe, Scandinavia, Denmark. Depending on the spatial region chosen by the user, different answers will be returned. If the user selects the name Europe, all three hotels will be returned as they are all located in cities that are part of Europe. If the user narrows down the requested region to Scandinavia, only the Radisson in Copenhagen will be returned, because the other cities are not considered to belong to Scandinavia. The upper-left part of Fig. 9.15 shows a similar partonomy related to federal states in Germany.

Besides clearly defined regions such as countries, a place-name structure may also contain names of less well-defined regions such as mountain ranges or seas. The upper right part of Fig. 9.15 shows an example of a mountain range (square box in the hierarchy) inserted into the partonomy by relating its spatial extension to federal states. In the case of our example, the user might ask for a hotel on the Baltic Sea. Clearly none of the hotels are part of the Baltic Sea, making clear that a partonomy alone is not enough to process spatial queries. For this purpose, BUSTER combines the partonomy with a

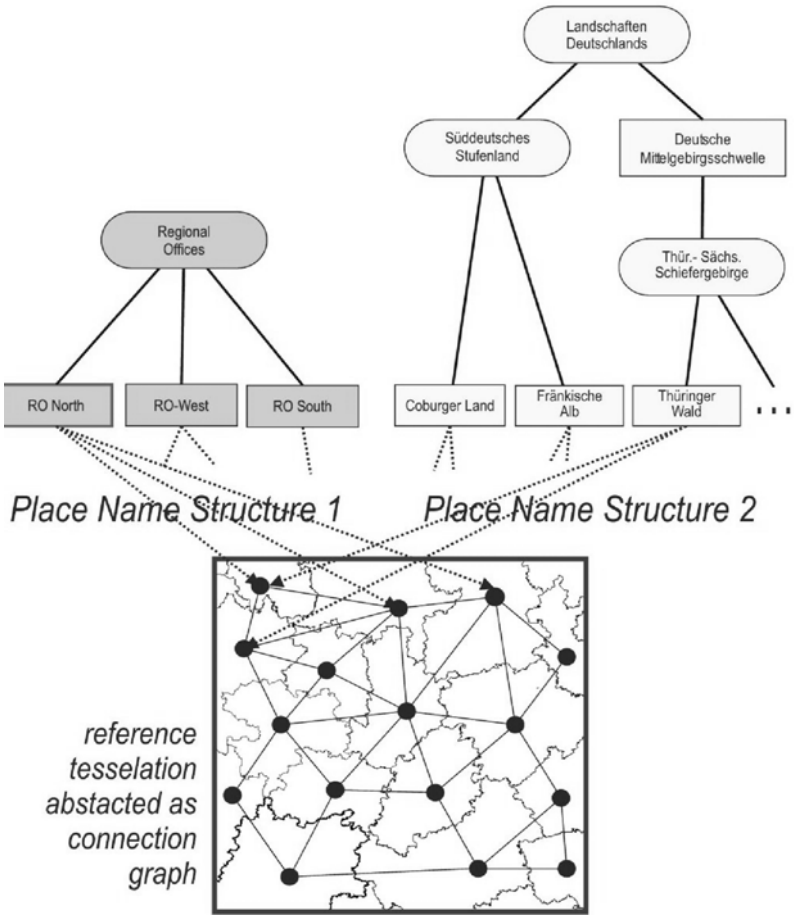


Fig. 9.15. Representation of spatial knowledge (from [Voegele et al., 2003])

connection graph for an underlying tessellation of spatial regions with well defined boundaries (compare the lower part of Fig. 9.15). In our example this connection graph provides the knowledge that Denmark as well as Germany are connected to the Baltic Sea and are therefore more relevant than the UK. Therefore, the Radisson in Copenhagen and the Mercure in Hamburg are returned as results.

Another function of the tessellations underlying the place-name structures is the integration of spatial information during query answering. It allows different information sources to use different partonomies of place names.

As long as they are based on the same underlying tessellation, the spatial relevance can still be determined based on this shared model.

9.4 Conclusions

The systems discussed in this chapter address various aspects of identifying and integrating heterogeneous and distributed information sources of related topics. The integration problem is addressed on different levels including syntax, structure and semantic integration. The systems have successfully been applied in different domains such as database integration, experience management in large companies and geographic information processing. These applications show that the models and methods described in this book are not only of theoretical interest, but that they contribute to a practical solution for information-sharing problems. Especially, we conclude that the general framework described in this book can be put to work using existing Web technologies: shared ontologies can be encoded in RDF Schema, OWL can be used to build source ontologies. Information sources in terms of collections of HTML documents can be linked to these ontologies using specialized wrappers and annotation tools. Finally, mapping and filtering methods can be implemented on top of existing subsumption reasoners that can be accessed over the Web. This tight coupling with existing technologies makes us optimistic about the potential contribution of the framework to a more intelligent Semantic Web. We also have to notice, however, that currently successful applications are only reported in rather restricted application domains rather than an open Semantic Web environment. Issues such as scalability and automatic generation of mappings still need investigation before systems are ready to move out to the Web.

Further reading

The OntoBroker system is presented in [Decker et al., 1999] in more detail. Frame-logic, the logical formalism used in OntoBroker is introduced in [Kifer et al., 1995]. The most complete description of the OBSERVER system is in [Mena and Illarramendi, 2001]. The methods for query planning based on approximate re-writing is discussed in [Mena et al., 2000b]. A description of the BUSTER system can be found in [Visser and Schuster, 2002]. The methods for determining spatial relevance in the BUSTER system are described in [Schlieder et al., 2001] and [Voegele et al., 2003].

Part IV

Distributed ontologies