TLP **7** (3): 249–300, 2007. © 2007 Cambridge University Press
doi:10.1017/S1471068406002754 Printed in the United Kingdom

A knowledge-based approach for selecting information sources[*]

THOMAS EITER, MICHAEL FINK and HANS TOMPITS
Institut für Informationssysteme, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria
(*e-mail:* {eiter,michael,tompits}@kr.tuwien.ac.at)

submitted 16 April 2004; revised 13 August 2005; accepted 21 April 2006

Abstract

Through the Internet and the World-Wide Web, a vast number of information sources has become available, which offer information on various subjects by different providers, often in heterogeneous formats. This calls for tools and methods for building an advanced information-processing infrastructure. One issue in this area is the selection of suitable information sources in query answering. In this paper, we present a knowledge-based approach to this problem, in the setting where one among a set of information sources (prototypically, data repositories) should be selected for evaluating a user query. We use extended logic programs (ELPs) to represent rich descriptions of the information sources, an underlying domain theory, and user queries in a formal query language (here, XML-QL, but other languages can be handled as well). Moreover, we use ELPs for declarative query analysis and generation of a query description. Central to our approach are declarative *source-selection programs*, for which we define syntax and semantics. Due to the structured nature of the considered data items, the semantics of such programs must carefully respect implicit context information in source-selection rules, and furthermore combine it with possible user preferences. A prototype implementation of our approach has been realized exploiting the DLV KR system and its plp front-end for prioritized ELPs. We describe a representative example involving specific movie databases, and report about experimental results.

KEYWORDS: knowledge representation, nonmonotonic reasoning, logic programming, answer-set programming, information-source selection, data repositories, preference handling

1 Introduction

Through the Internet and the World-Wide Web (WWW), a wealth of information has become available to a large group of users. A huge number of documents, files, and data repositories on a range of subjects are offered by different providers, which may be non-profit individuals, organizations, or companies. Such data repositories

[*] Part of the material in this paper has appeared, in preliminary form, in the Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR ’02), pp. 49–60, April 22–25, Toulouse, France, 2002.

are currently in heterogeneous formats, but the trend is that XML becomes a future de-facto standard for releasing data on the Web, since this eases data exchange. Nonetheless, the quality of their contents may differ significantly with respect to aspects such as their accuracy, coverage of certain topics and completeness for them, or refresh cycle, to mention just a few.

Accessing and processing data on the Web calls for developing tools and methods for an advanced information-processing infrastructure. *Mediators* (Wiederhold 1993) and special *information agents* ("middle agents" (Decker *et al.* 1997)), which provide various services including finding, selecting, and querying relevant information sources, play an important role here. The potential of knowledge-based approaches—and in particular of logic programming—for developing reasoning components for intelligent information agents is recognized in the AI community and outlined, e.g., by Dimopoulos and Kakas (2001), Eiter *et al.* (2002b), and Sadri and Toni (2000).

In this paper, we pursue this issue further and present a declarative approach for information-source selection in the following setting. Given a query by a user in some formal query language and a suite of information sources over which this query might be evaluated, which of these sources is the best to answer the query, i.e., such that the utility of the answer, measured by the quality of the result and other criteria (e.g., costs), is as large as possible for the user? Note that this problem is in fact not bound to information sources on the Web but is of interest in any context where different candidate information sources (e.g., scientific databases, newspaper archives, stock exchange predictions, etc.) are available and one of them should be selected. Selection of a single source may be desired because of (high) cost associated with accessing each source, for instance. Furthermore, problems arising by integrating data from different sources (like inconsistencies between sources) can be circumvented this way.

For a concrete example, consider the following scenario to illustrate our ideas.

Example 1
Assume that some agent has access to XML information sources, s_1, s_2, and s_3, about movies. Furthermore, suppose that the following XML-QL[1] query is handed to the agent, which informally asks a source for the titles of all movies directed by Alfred Hitchcock:

```
FUNCTION HitchcockMovies($MovieDB:"Movie.dtd") {
  CONSTRUCT <MovieList> {
    WHERE <MovieDB> <Movie>
             <Title> $t </Title>
             <Director> <Personalia>
                 <FirstName> "Alfred" </FirstName>
                 <LastName> "Hitchcock" </LastName>
             </Personalia> </Director>
          </Movie> </MovieDB>
    IN source($MovieDB)
    CONSTRUCT <Movie> $t </Movie>
 } </MovieList> }
```

[1] For details about XML-QL, cf. Section 2.2.

Here, `$t` is a variable into which the value of attribute `Title` is selected, for usage in the resulting construction. Suppose the agent knows that s_1 is a very good source for information about directors, while s_2 has usually good coverage about person data; all that is known about s_3, however, is that it is not very reliable. In this situation, we would expect that the agent selects s_1 for querying.

Obviously, a sensible solution to this problem is nontrivial and involves various aspects such as taking basic properties of the information sources, knowledge about their contents, and knowledge about the particular application domain into account. These aspects have to be suitably combined, and reasoning may be needed to elicit implicit knowledge. We stress that the general problem considered here is distinct from a simple keyword-based search as realized by Web engines like Google,[2] and consequently we do not propose a method for competing with these tools here.[3] In fact, we are concerned with *qualitative* selection from different alternatives, based on rich meta-knowledge and a formal semantics, thereby respecting preference and context information which involves heuristic defaults.

Our approach, which incorporates aspects mentioned above, makes several contributions, which are briefly summarized as follows.

(1) We base our method on the *answer-set programming* paradigm, in which problems are encoded in terms of nonmonotonic logic programs and solutions are extracted from the models of these programs (cf. Baral (2003) for a comprehensive treatise on answer-set programming). More precisely, we use *extended logic programs* (ELPs) under the answer-set semantics (Gelfond and Lifschitz 1991), augmented with *priorities* (cf., e.g., Brewka and Eiter (1999), Delgrande *et al.* (2003), or Inoue and Sakama (2000) for work about priorities in answer-set programming) and *weak constraints* (Buccafurri *et al.* 2000; Leone *et al.* 2006), to represent rich descriptions of the information sources, an underlying domain theory, and queries in a formal language. We perform query analysis by ELPs and compute *query descriptions*. Here, we consider XML-QL (Deutsch *et al.* 1999), but our approach is not committed to semi-structured data and XML per se, and other formal query languages can be handled as well (e.g., Schindlauer (2002) adopts our query-analysis method for the ubiquitous SQL language for relational databases).

(2) At the heart, a declarative *source-selection program* represents both *qualitative* and *quantitative* criteria for source selection, in terms of rules and soft constraints. The rules may access information supplied by other programs, including object and value occurrences in the query. For example, a rule r_1 may state that a query about a person Alfred Hitchcock should be posed to source s_1. Furthermore, ordinal rule priorities can be employed in order to specify source-selection preference. For example, a priority may state that a certain rule mentioning a last name in the query is preferred over another rule mentioning the concept *person* only. Rules and priorities are of qualitative nature and are taken into account for singling out a coherent decision for the source selection in model-theoretic terms. Quantitative

[2] Google's homepage is found at `http://www.google.com`.

[3] Note that Google does not index XML files or databases underlying Web query interfaces, and hence cannot be readily applied for the purposes considered here.

criteria (like, e.g., cost) are used to discriminate between different such options by means of an objective function which is optimized. To this end, conditions in terms of conjunctions of literals can be stated whose violation is penalized (e.g., the selection of a certain source might be penalized but not strictly forbidden), and total penalization is minimized. Such a two step approach seems to be natural and provides the user with a range of possibilities to express his or her knowledge and selection desires in convenient form.

(3) We consider the interesting and, to the best of our knowledge, novel issue of *contexts* in nonmonotonic logic programs, which is similar to preference based on specificity (Delgrande and Schaub 1994; Geerts and Vermeir 1993; Geerts and Vermeir 1995). Structured data items require a careful definition of the selection semantics, since an attribute might be referenced following a path of indirections, starting from a root object and passing through other objects. In Example 1, for instance, the attribute *FirstName* is referenced with the path *Movie/Director/Personalia/FirstName*, which starts at an object of type *Movie* and passes through objects of type *Director* and *Personalia*. Each of these objects opens a context in which *FirstName* is referenced along the remaining path. Intuitively, a context is less specific the closer we are at the end of the path. Thus, for example, the reference from *Personalia* is less specific than from *Movie*, and the latter should have higher priority. Note that such priority is not based on *inheritance* (which is tailored for "flat" objects). Therefore, inheritance-based approaches such as those by Laenens and Vermeir (1990) or Buccafurri *et al.* (1996) do not apply here. Furthermore, implicit priorities derived from context information as above must be combined with explicit user preferences from the selection policy, and arising conflicts must be resolved.

(4) We have implemented a prototype, based on the KR system DLV (Leone *et al.* 2006) and its front-end plp (Delgrande *et al.* 2001) for prioritized ELPs, which we used to build a model application involving movie information sources. It comprises several XML databases, wrapped from movie databases on the Web, and handles queries in XML-QL. Experiments that we have conducted showed that the system behaved as expected on a number of natural queries, some of which require reasoning from the background knowledge to identify the proper selection.

The reason to use a knowledge-based approach—and in particular an answer-set programming approach—for source selection rather than a standard decision-theoretic approach based on utility functions is motivated by the following advantages:

- Source-selection programs, which are special kinds of extended logic programs, are declarative and have a well-defined formal semantics, both under qualitative as well as under quantitative criteria.
- The formalism is capable of handling incomplete information and performing nonmonotonic inferences, which, arguably, is an inherent feature of the problem domain under consideration.
- Changes in the specification of the source-selection process are easily incorporated by modifying or adding suitable rules or constraints, without the need

for re-designing the given program, as may be the case, e.g., in procedural languages.

- Finally, the declarative nature of the answer-set semantics formalism permits also a coupling with sophisticated ontology tools, as well as with reasoning engines for them, providing advanced features for the domain knowledge. In particular, the approach of Eiter *et al.* (2004, 2005b), providing a declarative coupling of logic programs under the answer-set semantics with description-logic knowledge bases, can be integrated into our framework.

We note that while we focus here on selecting a *single source*, our approach can be easily extended to select *multiple information sources*, as well as to perform ranked selections (cf. Section 6.4).

The rest of this paper is organized as follows. The next section contains the necessary prerequisites from answer-set programming and XML-QL, and Section 3 gives a brief outline of our approach. In Section 4, we consider the generation of an internal query representation, while Section 5 addresses the modeling of sources. Section 6, then, is devoted to source-selection programs and includes a discussion of some of their properties. The implementation and the movie application, as well as experimental results, are the topics of Section 7. Section 8 addresses related work, and Section 9 concludes the main part of the paper with a brief summary and open research issues. Certain technical details and additional properties of our approach are relegated to an appendix.

2 Preliminaries

2.1 Answer-set programming

We recall the basic concepts of answer-set programming. Let $\mathscr{L}$ be a function-free first-order language. Throughout this paper, we denote variables by alphanumeric strings starting with an upper-case letter, anonymous variables by '_', and constants by alphanumeric strings starting with a lower-case letter or by a string in double quotes.

An *extended logic program* (ELP) (Gelfond and Lifschitz 1991) is a finite set of rules over $\mathscr{L}$ of form

$$L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n, \tag{1}$$

where each L_i, $0 \leqslant i \leqslant n$, is a *literal*, i.e., an atom A or a negated atom $\neg A$, and "*not*" denotes *negation as failure*, or *default negation*. Intuitively, a rule of form (1) states that we can conclude L_0 if (i) $L_1, \ldots, L_m$ are known and (ii) $L_{m+1}, \ldots, L_n$ are *not* known. For a rule r as above, we call the literal L_0 the *head* of r (denoted $H(r)$) and the set $\{L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n\}$ the *body* of r (denoted $B(r)$). Furthermore, we define $B^+(r) = \{L_1, \ldots, L_m\}$ and $B^-(r) = \{L_{m+1}, \ldots, L_n\}$. If $B(r) = \emptyset$, then r is called a *fact*. We write $r(V_1, \ldots, V_n)$ to indicate that rule r has variables $V_1, \ldots, V_n$. To ease notation, for any program Π and any set S of literals, $\Pi \cup S$ stands for the program $\Pi \cup \{L \leftarrow \mid L \in S\}$. Finally, for a literal L, we write $\neg L$ to denote its complementary literal, i.e., $\neg L = A$ if $L = \neg A$, and $\neg L = \neg A$ if $L = A$, for any atom A.

The semantics of an ELP Π is given in terms of the semantics of its ground instantiation, $\mathscr{G}(\Pi)$, over the Herbrand universe $U_{\mathscr{L}}$ of $\mathscr{L}$, which is the language generated by Π. The program $\mathscr{G}(\Pi)$ contains all instances of rules from Π, i.e., where the variables are (uniformly) replaced with arbitrary terms from $U_{\mathscr{L}}$. Recall that a literal, rule, program, etc., is *ground* iff it contains no variables. In what follows, we assume that all such objects are ground.

An *interpretation*, X, is a consistent set of (ground) literals, i.e., X does not contain a complementary pair $A, \neg A$ of literals. A literal, L, is *true* in X if $L \in X$, and *false* otherwise. The body, $B(r)$, of a rule r is true in X iff (i) each $L \in B^+(r)$ is true in X and (ii) each $L \in B^-(r)$ is false in X. Rule r is true in X iff $H(r)$ is true in X whenever $B(r)$ is true in X. Finally, a program, Π, is true in X, or X is a *model* of Π, iff all rules in Π are true in X. We write $X \models \alpha$ to indicate that an object α, which may be either a literal, the body of a rule, a rule, or a program, is true in X.

Let X be a set of literals and Π a program. The *Gelfond-Lifschitz reduct*, or simply *reduct*, Π^X, of Π *relative to* X is given by

$$\Pi^X = \{H(r) \leftarrow B^+(r) \mid r \in \Pi \text{ and } B^-(r) \cap X = \emptyset\}.$$

We call X an *answer set* of Π iff X is a minimal model of Π^X with respect to set inclusion. Observe that any answer set of Π is *a fortiori* a model of Π. The set of all *generating rules* of an answer set X with respect to Π is given by

$$GR(X, \Pi) = \{r \in \Pi \mid X \models B(r)\}.$$

Example 2
Let $\Pi = \{s \leftarrow \mathit{not}\ t;\ n \leftarrow;\ t \leftarrow n, \mathit{not}\ s;\ w \leftarrow t\}$. For the interpretation $X_1 = \{n, t, w\}$, we have $\Pi^{X_1} = \{n \leftarrow\ ;\ t \leftarrow n;\ w \leftarrow t\}$. Clearly, X_1 is a minimal model of Π^{X_1}, and thus X_1 is an answer set of Π. Note that $X_2 = \{s, n\}$ is another answer set of Π.

A (possibly non-ground) program Π is *locally stratified* (Przymusinski 1988) iff there exists a mapping λ assigning each literal occurring in $\mathscr{G}(\Pi)$ a natural number such that, for each rule $r \in \mathscr{G}(\Pi)$, it holds that (i) $\lambda(H(r)) \geqslant \max_{L \in B^+(r)} \lambda(L)$ and (ii) $\lambda(H(r)) > \max_{L \in B^-(r)} \lambda(L)$. Note that Π is (*globally*) *stratified* (Apt *et al.* 1988) if, additionally, for all positive (resp., negative) literals L and L' with the same predicate, $\lambda(L) = \lambda(L')$ holds. It is well-known that if a program is locally stratified, then it has at most one answer set.

A refinement of the answer-set semantics is the admission of preferences among the rules of a given ELP, yielding the class of *prioritized logic programs*. Several approaches in this respect have been introduced in the literature, like, e.g., those by Brewka and Eiter (1999) or Inoue and Sakama (2000); here, we use a preference approach based on a method due to Delgrande *et al.* (2003), defined as follows.

Let Π be an ELP and $<$ a strict partial order between the elements of Π (i.e., $<$ is an irreflexive and transitive relation). Informally, for rules $r_1, r_2 \in \Pi$, the relation $r_1 < r_2$ expresses that r_2 has preference over r_1. Define the relation $<_{\mathscr{G}}$ over the ground instantiation $\mathscr{G}(\Pi)$ of Π by setting $\hat{r}_1 <_{\mathscr{G}} \hat{r}_2$ iff $r_1 < r_2$, for $\hat{r}_1, \hat{r}_2 \in \mathscr{G}(\Pi)$.

Then, the pair $(\Pi, <)$ is called a *prioritized extended logic program*, or simply a *prioritized logic program*, if the relation $<_{\mathscr{G}}$ is a strict partial order.

The semantics of prioritized programs is as follows. Let $(\Pi, <)$ be a prioritized logic program where Π is ground, and let X be an answer set of Π. We call X a *preferred answer set* of $(\Pi, <)$ iff there exists an enumeration $\langle r_i \rangle_{i \in I}$ of $GR(X, \Pi)$ such that, for every $i, j \in I$, we have that:

(P_1) $B^+(r_i) \subseteq \{H(r_k) \mid k < i\}$;
(P_2) if $r_i < r_j$, then $j < i$; and
(P_3) if $r_i < r'$ and $r' \in \Pi \setminus GR(X, \Pi)$, then $B^+(r') \nsubseteq X$ or $B^-(r') \cap \{H(r_k) \mid k < i\} \neq \emptyset$.

Conditions (P_1)–(P_3) realize a strongly "prescriptive" interpretation of preference, in the sense that, whenever $r_1 < r_2$ holds, it is ensured that r_2 is known to be applied or blocked ahead of r_1 (with respect to the order of rule application). More specifically, (P_2) guarantees that all generating rules are applied according to the given order, whilst (P_3) assures that any preferred yet inapplicable rule is either blocked due to the non-derivability of its prerequisites or because it is defeated by higher-ranked or unrelated rules. As shown by Delgrande *et al.* (2003), the selection of preferred answer sets can be encoded by means of a suitable translation from prioritized logic programs into standard ELPs.

Preferred answer sets of a prioritized program $(\Pi, <)$ where Π is non-ground are given by the preferred answer sets of the prioritized program $(\mathscr{G}(\Pi), <_{\mathscr{G}})$, where $<_{\mathscr{G}}$ is as above. Note that the concept of prioritization realizes a *filtering* of the answer sets of a given program Π, as every preferred answer set of $(\Pi, <)$ is an answer set of Π, but not vice versa.

Besides imposing *qualitative selection criteria*, like assigning preferences between different rules, another refinement of the answer-set semantics are *weak constraints* (Buccafurri *et al.* 2000; Leone *et al.* 2006), representing a *quantitative filtering of answer sets*. Formally, a weak constraint is an expression of form

$$\Leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n\ [w : l], \tag{2}$$

where each L_i, $1 \leqslant i \leqslant n$, is a literal (not necessarily ground) and $w, l \geqslant 1$ are natural numbers.[4] The number w is the *weight* and l is the *priority level* of the weak constraint (2). Given an interpretation X, the weight of a ground weak constraint c of the above form with respect to a level $l' \geqslant 1$, $weight_{c,l'}(X)$, is w, if $X \models L_i$, $1 \leqslant i \leqslant m$, $X \not\models L_{m+j}$, $1 \leqslant j \leqslant n$, and $l' = l$, and 0 otherwise; the weight of a non-ground weak constraint c with respect to a level l, $weight_{c,l}(X)$, is given by $\sum_{c' \in \mathscr{G}(c)} weight_{c',l}(X)$, where $\mathscr{G}(c)$ denotes the set of all ground instances of c. Weak constraints select then those answer sets X of the weak-constraint-free part of a program Π for which the associated vector

$$weights(X) = (weight_{l_{\max}}(X), weight_{l_{\max}-1}(X), \ldots, weight_0(X))$$

[4] The part "$[w : l]$" is convenient syntactic sugar for the original definition by Buccafurri *et al.* (2000), which merely provided a partitioning of the weak constraints in priority levels.

is lexicographic smallest, where $l_{\max}$ is the highest priority level occurring and $weight_l(X) = \sum_{c \in wc(\Pi)} weight_{c,l}(X)$, for each l, with $wc(\Pi)$ denoting the set of all weak constraints occurring in Π. Informally, first those answer sets are pruned for which the weight of violated constraints is not minimal at the highest priority level; from the remaining answer sets, those are pruned where the sum of weights of violated constraints in the next lower level is not minimal, and so on. For example, if we add in Example 2 the weak constraints $c_1 : \Leftarrow n, not\ w\ [3:1]$ and $c_2 : \Leftarrow t, w\ [1:2]$, then we have $weights(X_1) = (1,0)$ and $weights(X_2) = (0,3)$; hence, the answer set X_1 is discarded.

The numeric lexicographic preference can be reduced by usual techniques to an objective function $H^{\Pi}(X)$, which assigns each answer set X an integer such that those answer sets X for which $H^{\Pi}(X)$ is minimal are precisely those for which $weights(X)$ is lexicographically smallest. In the above example, X_2 is selected as the "optimal" answer set. While the availability of both weights and levels is syntactic sugar, they are very useful for expressing preferences in a more natural and convenient form. In the example above, putting c_2 at level 2 dominates c_1 which is at level 1. Weights within the same layer can be used for fine-tuning. For formal details and more discussion, we refer the reader to Leone *et al.* (2006).

2.2 XML-QL

We next introduce basic concepts of XML-QL (Deutsch *et al.* 1999), a query language for data stored in the *Extensible Markup Language* (XML). We assume that the reader is familiar with XML, which has emerged as a standard for providing (semi-structured) data on the Web. While syntactically similar to the *Hypertext Markup Language* (HTML), features have been added in XML for data-representation purposes such as user-defined tags and nested elements. Unlike relational or object-oriented data, XML is *semi-structured*, i.e., it can have irregular (and extensible) structure and attributes (or schemas) are stored with the data. The structure of an XML document can be optionally modeled and validated against a *Document Type Descriptor* (DTD). In this paper, we take a database-oriented view of XML documents, considering them as databases and a corresponding DTD as its database schema. For a comprehensive introduction to semistructured data and database aspects about them, we refer to Abiteboul *et al.* (2000).

XML-QL is a declarative, relationally complete query language for XML data, which can not only query XML data, but also construct new XML documents from query answers, i.e., it can also be used to restructure XML data. Its syntax deviates from the well-known "select-from-where syntax" of the *Structured Query Language* (SQL), but can be decomposed into three syntactical units as well:

1. a *where part* (keyword `WHERE`), specifying a selection condition by element reference and comparison predicates;
2. a *source part* (keyword `IN`), declaring a data source for the query (an external file, or an internal variable); and
3. a *construct part* (keyword `CONSTRUCT`), defining a structure for the resulting document.

In the latter part, subqueries can be built by nesting.

XML-QL uses element patterns to match data in an XML document. Elements are referenced by their names and are traversed according to the XML source structure. Thus, *reference paths* can be identified with every matching. Variables are in general not bound to elements but to element contents (but syntactic sugar exists for element binding). Furthermore, elements can be joined by values using the same variable in two matchings, i.e., theta-joins can be expressed.

Let us briefly illustrate the most basic concepts in the following example; for further details, we refer to Deutsch *et al.* (1999) and Abiteboul *et al.* (2000).

Example 3
Throughout the paper, we consider XML-QL queries stored as XML-QL functions, which serves two purposes. First, it allows us to efficiently query several XML documents by dynamic bindings of data sources, and second, we can additionally specify that a data source has to obey a certain DTD. The following query is represented as an XML-QL function. Upon its invocation, the variable `$MovieDB` is instantiated with the name of an XML document to be queried, which has to be structured according to the DTD detailed in Appendix A.

```
FUNCTION ExampleQuery($MovieDB:"Movie.dtd") {
  WHERE <MovieDB> <Movie> $m1 </Movie> </MovieDB> IN source($MovieDB),
        <Actor> $a </Actor> IN $m1,
        <MovieDB> <Movie> $m2 </Movie> </MovieDB> IN source($MovieDB),
        <Actor> $a </Actor> IN $m2,
        $m1 != $m2
  CONSTRUCT <x2Actor> $a </x2Actor>
}
```

In the where part of the above query, variables `$m1` and `$m2` are bound to different matchings under the reference path *MovieDB/Movie*. Furthermore, the two matchings are joined by the (common) value of variable `$a`, referenced under element *Actor*. The construct part of the query creates a new XML document by listing the values of `$a` marked-up with tags `<x2Actor>`. Intuitively, the query returns all actors found in a given XML document about movies which act in at least two movies.

3 Overview of the approach

Before presenting the technical details of our approach, it is helpful to give a short overview. While the motivating example in Section 1 is simple, it shows that the source-selection process involves different kinds of knowledge, including

- knowledge about which "interesting" information should be extracted from a given formal query expression Q,
- knowledge about the information sources and their properties,
- background knowledge about the application domain and the ontology used for its formalization, and
- specific rules which guide the source selection, based on preferences or generic principles.

In our approach, this is formalized in terms of the notion of a *selection base*

$$\mathscr{S} = (\Pi_{qa}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u),$$

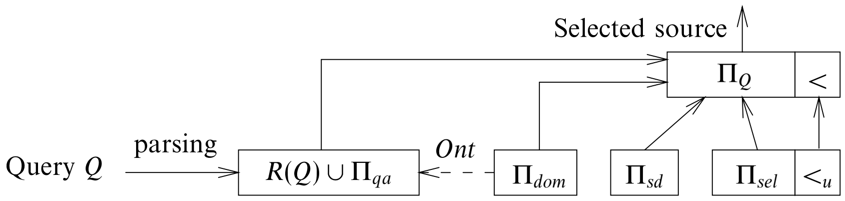

Fig. 1. Using a selection base $\mathscr{S} = (\Pi_{qa}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u)$ for source selection for a query Q.

where $\Pi_{qa}, \Pi_{sd}, \Pi_{dom}$ are ELPs, called *query-analysis program*, *source description*, and *domain theory*, respectively, and $(\Pi_{sel}, <_u)$ is a prioritized logic program with a special syntax, called *source-selection program*. Given a selection base $\mathscr{S}$ as above, the possible solutions of a query Q relative to $\mathscr{S}$ are determined by the *selection answer sets* of the source-selection program $(\Pi_{sel}, <_u)$, which are defined as preferred answer sets of a prioritized logic program $\mathscr{E}(\mathscr{S}, Q) = (\Pi_Q, <)$, associated with $\mathscr{S}$ and Q, as shown in Figure 1.

The components of a selection base serve the following purposes:

Query-analysis program Π_{qa}: For any query Q as in Example 1, a high-level description is extracted from a low-level (syntactic) representation, $R(Q)$, given as a set of elementary facts, by applying Π_{qa} to $R(Q)$ and ontological knowledge, *Ont*, about concepts (types) and synonyms from the domain theory Π_{dom}, in terms of facts for predicates $class(O)$ and $synonym(C_1, C_2)$. Informally, the rules of Π_{qa} single out the essential parts of Q, such as occurrence of attributes and values in the query, comparison and joins, or subreference paths of attributes from objects on a reference path. For instance, in Example 1, the attribute *FirstName* from an object of type *Director* is referenced via path *Personalia/FirstName* on the reference path *Movie/Director/Personalia/FirstName* from the root.

Source description Π_{sd}: This program contains information about the available sources, using special predicates for query topics, cost aspects, and technical aspects.

Domain theory Π_{dom}: The agent's knowledge about the specific application domain (like, e.g., the movie area) is represented in the domain theory Π_{dom}. It includes ontological knowledge and further background knowledge, permitting (modest) common-sense reasoning. The ontology is assumed to have concepts (classes), attributes, and instance and subconcept information, which are provided via $class(O)$, $class\_att(C, A)$, $instance(O, C)$, and $is\_a(C_1, C_2)$ predicates, respectively. Furthermore, it is assumed to have information about concept synonyms, provided via predicate $synonym(C_1, C_2)$. The ontology may be partly established using meta-information about the data in the information sources (e.g., an XML DTD), and with ontology rules. Since ontological reasoning is orthogonal to our approach, we do not consider it here and refer to Eiter *et al.* (2003) for a further elaboration.

Source-selection program $(\Pi_{sel}, <_u)$: The information source selection is specified by rules and constraints, which refer to predicates defined in the above programs. It comprises both *qualitative aspects* and *quantitative aspects* in terms of optimization criteria (concerning, e.g., cost or response time), which are expressed using weak constraints (Buccafurri *et al.* 2000). Furthermore, the user can define preferences between rules, in terms of a strict partial order, $<_u$. These preferences are combined with implicit priorities that emerge from the *context* in which source selection rules should be applied, and possible preference conflicts are resolved.

Given a query Q, the overall evaluation relative to $\mathscr{S}$, then, proceeds in three steps:

Step 1 (query description): The input query Q is parsed and mapped into the internal query representation, $R(Q)$, which is extended using Π_{qa} and *Ont* to the full query description.

Step 2 (qualitative selection): From $R(Q)$, Π_{qa}, Π_{sd}, and Π_{dom}, the qualitative part of Π_{sel} is used to single out different query options by respecting qualitative aspects only, where explicit preferences, $<_u$, and implicit priorities must be taken into account. To this end, a priority relation $<$ is computed on rules, which is then used in a prioritized logic program $(\Pi_Q, <)$. Candidate solutions are computed as preferred answer sets of $(\Pi_Q, <)$.

Step 3 (optimization): Among the candidates of Step 2, the one is chosen which is best under the quantitative aspects of Π_{sel}, and the selected source is output.

4 Query description

An integral feature of our approach is a meaningful description of a given formal query expression Q in an internal format. For our purposes, we need a suitable representation of the constituents of Q in terms of predicates and objects. Simply mapping Q (which is represented as a string) to logical facts which encode its syntax tree (i.e., the external format) does not serve our purposes. Rather, we need a meta-level description which provides "interesting" information about Q, such as occurrence of an attribute or a value in Q, related to the scope of appearance.

For example, in the query of Example 1, the value "Hitchcock" occurs in a selection on the attribute *LastName* reached by the reference path *Personalia/LastName* from *Director*. In the internal query representation, this selection will be represented by the fact $selects(o_3, equal, \text{"}Hitchcock\text{"})$, where o_3 is an internal name for the full reference path "*Movie/Director/Personalia/LastName*" (i.e., the reference path from the root), and by a fact $cref(o_3, \text{"}Director\text{"}, \text{"}Personalia/LastName\text{"}, q_1)$, where q_1 is an internal identifier for the query. Also, a fact $occurs(o_3, \text{"}Hitchcock\text{"})$ will be present that less specifically states that "*Hitchcock*" is associated with this reference path.

The general format of these three predicates, which play a vital part in our architecture, is as follows:

- $cref(O, C, P, Q)$ states that within the full reference path O in the syntax tree for query Q, the path from C to the leaf is P;[5]

[5] Note that Eiter *et al.* (2003) and Fink (2002) name this predicate *access*, and reference paths are called *access paths*.

- *occurs*(O, V) states that the value V is associated with the full reference path O in the overall query; and
- *selects*(O, R, V) is similar to *occurs*, but details the association with a comparison operator R.

In accord to the syntactical units of XML-QL, in our query-analysis method we adopt the general view in which a query expression consists of a *where part*, a *source part*, and a *construct part*. For the description of Q, we employ facts on designated predicates, which are independent of a fixed query language. These facts are divided into two groups, which we refer to as *parser facts* and *derived facts*, respectively.

4.1 Parser facts

The first group of facts, denoted $R(Q)$, is generated by a query parser, and is regarded as a "low-level" part of the query representation. The query parser scans the query string Q for extracting "interesting" information, and assembles information about structural information (such as about subqueries, and in which of them a reference to a certain attribute is made). The main purpose of $R(Q)$ is to filter and reduce the information which is present in the syntax tree of Q, and to assemble it into suitable facts. For that, the parser must introduce identifiers (names) for queries, subqueries, and other query constituents—in particular, references to *items* (i.e., attributes or concepts), which in a query are selected or compared to values or other items. Every item reference is given by a maximal reference path in Q, which we call an *item reference path* (*IRP*). The parser names each occurrence of an IRP with a unique constant (note that the same IRP may have multiple occurrences in Q).

Example 4
In Example 1, the query is named q_1. It has a subquery in the construct `<MovieList>` part, identified by q_2. There are three IRPs, namely

"*Movie/Title*",
"*Movie/Director/Personalia/FirstName*", and
"*Movie/Director/Personalia/LastName*".

Their identifiers are o_1, o_2, and o_3, respectively.

4.2 Derived facts

The second group of facts are those which are derived from $R(Q)$ by means of a further analysis. Compared to $R(Q)$, these facts can be regarded as a "high-level" description of the query. In particular, for the attribute or concept at the end of an IRP, the *contexts* of reference are determined, which are the suffixes of the IRP starting at some concept (as known from the underlying ontology). Intuitively, instances of this concept have the referenced item as a (nested) attribute. Detaching the leading concept from the suffix results in the notion of a *context-reference pair*, defined as follows:

Definition 1
A pair (C, P), where C is a concept from the ontology and P is a path, is a *context-reference pair* (CRP) of a query Q if Q contains an IRP with suffix "C/P".

Example 5
Continuing Example 1, assume that the concepts *MovieDB*, *Movie*, *Director*, and *Person* are in the ontology, and it is known that "*Personalia*" is a synonym of "*Person*" in the ontology. Then, from the IRP o_1 = "*Movie/Title*", the CRPs

$$(\text{“}MovieDB\text{”}, \text{“}Movie/Title\text{”}) \text{ and } (\text{“}Movie\text{”}, \text{“}Title\text{”})$$

are determined, and from the IRP o_2 = "*Movie/Director/Personalia/LastName*", the CRPs

$$\begin{aligned}
&(\text{“}MovieDB\text{”}, \text{“}Movie/Director/Personalia/FirstName\text{”}),\\
&(\text{“}Movie\text{”}, \text{“}Director/Personalia/FirstName\text{”}),\\
&(\text{“}Director\text{”}, \text{“}Personalia/FirstName\text{”}), \text{ and}\\
&(\text{“}Personalia\text{”}, \text{“}FirstName\text{”})
\end{aligned}$$

are obtained.

The high-level description facts are computed declaratively by evaluating a query-analysis logic program, Π_{qa}, to which the facts $R(Q)$ and further facts *Ont*, which provide ontological knowledge about concepts and synonyms from the domain theory, are added as "input". Furthermore, the program enriches the low-level predicate *subpath* by synonym information and closing *subpath* transitively. In summary, the query description is given by the (unique) answer set of the logic program $Ont \cup \Pi_{qa} \cup R(Q)$.

A detailed list of all query-description predicates, as well as the complete query-analysis program, can be found in Appendix B.

5 Source description

Besides query information and domain knowledge, the source-selection process requires a suitable description of the information sources to select from. This is provided by means of meta-knowledge represented in the source-description part of the knowledge base, given in the form of a (simple) logic program, Π_{sd}, which is assumed to have a unique answer set. Different predicates can be used for this purpose, depending on the specific application. In the following, we introduce, in an exemplary fashion, a basic suite of predefined *source-description predicates*, which cover several aspects of an information source:

(i) Thematic aspects:
- $accurate(S, T, V)$: source S, topic T, value V;
- $covers(S, T, V)$: source S, topic T, value V;
- $specialized(S, T)$: source S, topic T;
- $relevant(S, T)$: source S, topic T.

The first two predicates express the accuracy and coverage of a source for a topic, using values from $\{low, med, high\}$. The others are for stating that a source is specialized or relevant for a particular topic, respectively.

(ii) Cost aspects:
- $avg\_download\_time(S, V)$: source S, value V;

- $avg\_down\_time(S, V)$: source S, value V;
- $charge(S, V)$: source S, value V.

Costs for accessing an information source can be expressed by these predicates, again using values *low*, *med*, *high*, and, for *charge*, also *no*. While *charge* is used for direct costs, *avg_download_time* and *avg_down_time* are indirect costs (taking network traffic into account).

(iii) Technical aspects:

- $source\_type(S, T_1, T_2)$: source S, organizational type T_1, query type T_2;
- $source\_language(S, L)$: source S, language L;
- $data\_format(S, F)$: source S, format F;
- $update\_frequency(S, V)$: source S, value V;
- $last\_update(S, D)$: source S, date D;
- $reliable(S, V)$: source S, value V;
- $source(S)$: source S;
- $up(S)$: source S.

Different kinds of sources are distinguished by their type of organization (commercial or public) and by the type of data access provided (queryable, downloadable, or both). Besides source language and data format (XML, relational, HTML, text, or other), the frequency of data update (low, medium, or high), the date of the last update, or the reliability of a source (low, medium, or high) may be criteria for source selection. Finally, *source* and *up* are used to identify sources and to express that a source is currently accessible, respectively.

As already pointed out, the above predicates are just a rudiment of a vocabulary for source description, and we are far from claiming that they capture all aspects or that they capture each one in sufficient detail or granularity (like, e.g., the three-valued scale used). However, the user or administrator has the possibility to introduce further predicates and define them in the source-description program Π_{sd}. Note that Π_{sd} can take advantage of default rules to handle incomplete information, e.g., that a source is accessible by default, or that the language of text items is English.

We assume here furthermore that detailed source descriptions are edited by an administrator of an overall information system hosting the considered selection process. This does not preclude that a preliminary or partial description is created automatically, addressing aspects such as source language, type, data format, etc., nor that the information system is open for new sources entering it, advertising their description to a source registration. However, a number of aspects for selection, such as coverage, specialization, or relevance, might be difficult to assess automatically and require experience gained from interaction with a source like in real-life scenarios (think of different travel agencies offering flights, for instance). Here, the administrator might bring in such knowledge initially, and the description might be updated in accord to new information obtained, e.g., by performance monitoring and user feedback. For updating an employed description, approaches such as those discussed by Alferes *et al.* (2002) or Eiter *et al.* (2002a) may be applied. In general, however, this is a complex and interesting issue, but is beyond the scope of this paper.

In concluding, we remark that the proviso that Π_{sd} possesses a unique answer set can be ensured, e.g., by requiring (local) stratification of Π_{sd}, or by the condition that its well-founded model is total. In principle, the case of multiple answer sets of Π_{sd} could be admitted as well, which would give rise to different scenarios that could be handled in different ways; e.g., adhering to a credulous or skeptical reasoning principle, according to which the different scenarios are considered *en par* or such that only selections in all scenarios are retained, or to a preference-based approach which discriminates between the different scenarios. However, we do not elaborate further on this issue.

6 Source selection

We now introduce the central part of our architecture, viz. *source-selection programs.* Basically, a source-selection program is a prioritized logic program $(\Pi_{sel}, <_u)$ having four parts: (i) a core unit Π^c_{sel}, containing the actual source-selection rules, (ii) a set Π^{aux}_{sel} of auxiliary rules, (iii) an order relation $<_u$ defined over members of Π^c_{sel}, and (iv) an optimization part Π^o_{sel}, containing weak constraints.

6.1 Syntax

We first make the vocabulary of source-selection programs formally precise.

Definition 2
A *source-selection vocabulary*, $\mathscr{A}_{sel}$, consists of the following pairwise disjoint categories:

(i) function-free vocabularies $\mathscr{A}_{qd}$, $\mathscr{A}_{sd}$, and $\mathscr{A}_{dom}$, referred to as the *query-description vocabulary*, the *source-description vocabulary*, and the *domain-theory vocabulary of* $\mathscr{A}_{sel}$, respectively, where $\mathscr{A}_{qd}$ and $\mathscr{A}_{sd}$ contain the predicates introduced in Section 4 and 5;
(ii) the predicate $query\_source(S, Q)$, expressing that source S is selected for evaluating query Q;
(iii) the predicates $default\_class(O, C, Q)$ and $default\_path(O, P, Q)$; and
(iv) a set $\mathscr{A}_{aux}$ of auxiliary predicates.

Informally, the predicates $default\_class(O, C, Q)$ and $default\_path(O, P, Q)$ are projections of $cref(O, C, P, Q)$ and serve to specify a default status for selection rules depending on context-reference pairs matched in the query Q. For example, a predicate $default\_class(O, \text{“}Person\text{”}, Q)$ in the body of rule r expresses that r is eligible in case the concept *Person* occurs in the reference path O and there is no other rule r' that refers to some CRP (C', P') matched in Q. These defaults are semantically realized using a suitable rule ordering.

The set of all literals over atoms in $\mathscr{A}_\ell$, for $\ell \in \{qd, sd, dom, aux, sel\}$, is denoted by Lit_ℓ.

Definition 3
Let $\mathscr{A}_{sel}$ be a source-selection vocabulary. A *source-selection program over $\mathscr{A}_{sel}$* is a tuple $(\Pi_{sel}, <_u)$, where

(i) Π_{sel} is a collection of rules over $\mathscr{A}_{sel}$ consisting of the following parts:
  (a) the *core unit*, Π^c_{sel}, containing rules of form

$$query\_source(S, Q) \leftarrow L_1, \dots, L_m, not\ L_{m+1}, \dots, not\ L_n,$$

  (b) a set Π^{aux}_{sel} of *auxiliary rules* of form

$$L_0 \leftarrow L_1, \dots, L_m, not\ L_{m+1}, \dots, not\ L_n,$$

  and
  (c) an *optimization part*, Π^o_{sel}, containing weak constraints of form

$$\Leftarrow\ L_1, \dots, L_m, not\ L_{m+1}, \dots, not\ L_n\ [w : l],$$

  where L_0 is either a literal from Lit_{aux} or is of form $\neg query\_source(\cdot, \cdot)$, $L_i \in Lit_{sel}$ for $1 \leqslant i \leqslant n$, and $w, l \geqslant 1$ are integers, and
(ii) $<_u$ is a strict partial order between rules in Π^c_{sel}.

The elements of $<_u$ are called *user-defined preferences.* If $r_1 <_u r_2$, then r_2 is said to have *preference over* r_1.

The rules in the core unit Π^c_{sel} serve for selecting a source, based on information from the domain description, the source description, the query description, and possibly from auxiliary rules. The latter may be used, e.g., for evaluating complex conditions. In terms of $<_u$, preference of source selection can be expressed. As well, the weak constraints in Π^o_{sel} are used to filter answer sets under quantitative conditions.

By assembling all constituents for source selection into a single compound, we arrive at the notion of a *selection base*, as informally described in Section 3.

Definition 4
Let $\mathscr{A}_{sel}$ be a source-selection vocabulary. A *selection base over $\mathscr{A}_{sel}$* is a quintuple $\mathscr{S} = (\Pi_{qa}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u)$, consisting of the query-analysis program Π_{qa} over $\mathscr{A}_{qd}$, programs Π_{sd} and Π_{dom} over $\mathscr{A}_{sd}$ and $\mathscr{A}_{dom}$, respectively, and a source-selection program $(\Pi_{sel}, <_u)$ over $\mathscr{A}_{sel}$.

Given that the components Π_{qa}, Π_{sd}, and Π_{dom} are understood, the source-selection program $(\Pi_{sel}, <_u)$ in a selection base $\mathscr{S}$ is the most interesting part, and $\mathscr{S}$ might be referred to just by this program. Furthermore, we assume in what follows that the source-selection vocabulary $\mathscr{A}_{sel}$ contains only those constants actually appearing in the elements of a selection base over $\mathscr{A}_{sel}$. Thus, we usually leave $\mathscr{A}_{sel}$ implicit.

Example 6
Consider a simple source-selection program, $(\Pi_{sel}, <_u)$, for our movie domain, consisting of the following constituents:

- Source-selection rules:

$r_1: \quad query\_source(s_2, Q) \leftarrow default\_class(O, \text{“}Person\text{”}, Q);$

$r_2: \quad query\_source(s_1, Q) \leftarrow selects(O, equal, \text{“}Hitchcock\text{”}), cref(O, \text{“}Director\text{”}, \text{“}Personalia/LastName\text{”}, Q);$

$r_3: \quad query\_source(S, Q) \leftarrow default\_path(O, \text{“}LastName\text{”}, Q), default\_class(O, T, Q), accurate(S, T, high).$

- Auxiliary rules:

$r_4: \quad high\_acc(T, Q) \leftarrow cref(O, T, P, Q), accurate(S, T, high);$

$r_5: \quad high\_cov(T, Q) \leftarrow cref(O, T, P, Q), covers(S, T, high).$

- Optimization constraints:

$c_1: \quad \Leftarrow query\_source(S, Q), high\_acc(T, Q), not\ accurate(S, T, high)\ [10:1];$

$c_2: \quad \Leftarrow query\_source(S, Q), high\_cov(T, Q), not\ covers(S, T, high)\ [5:1].$

- User preferences:

$r_1(Q, \_) <_u r_3(Q, \_, \_, \_).$

Intuitively, r_1 advises to choose source s_2 if the query involves persons and no more specific rule is eligible. Rule r_2 states to choose source s_1 if the query contains an explicit select on the movie director Hitchcock. Rule r_3 demands to choose a source if, on some query reference path, “*LastName*” is accessed under some concept T (with arbitrary intermediate reference path), and the source is highly accurate for T. Rules r_4 and r_5 define auxiliary predicates which hold on concepts T appearing in the query such that some source with high accuracy and coverage for T exists. The weak constraints c_1 and c_2 state penalties for choosing a source that does not have high accuracy (assigning weight 10) or coverage (assigning weight 5) for a concept in the query while such a source exists. Finally, $r_1(Q, \_) <_u r_3(Q, \_, \_, \_)$ expresses preference of instances of r_3 over r_1 on the same query.

6.2 Semantics

The semantics of a source-selection program $(\Pi_{sel}, <_u)$ in a selection base $\mathscr{S} = (\Pi_{qa}, \Pi_{dom}, \Pi_{sd}, \Pi_{sel}, <_u)$ on a query Q is given by means of a *selection answer set* of $(\Pi_{sel}, <_u)$, which is defined as a preferred answer set of a prioritized ELP $\mathscr{E}(\mathscr{S}, Q)$ associated with $\mathscr{S}$ and Q. The program $\mathscr{E}(\mathscr{S}, Q)$ is of the form $(\Pi_Q, <)$, where program Π_Q contains ground instances of rules and constraints in Π_{sel}, and further rules ensuring that a single source is selected per query and rules defining the default-context predicates. The order relation $<$ is formed from the user preferences $<_u$ and the implicit priorities derived from context references in the core unit and from auxiliary rules. Thereby, preference information must be suitably combined, as well as arising conflicts resolved, which we do by means of a cautious conflict-elimination policy.

We commence the formal details with the following notation: For any rule $r = H(r) \leftarrow B(r)$, its *defaultization*, r^{Δ}, is given by $H(r) \leftarrow B(r), not\ \neg H(r)$.[6] We assume that user-defined preferences between rules carry over to their defaultizations.

Definition 5
Let $\mathscr{S} = (\Pi_{qa}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u)$ be a selection base and Q a query. Then, the program Π_Q contains all ground instances of the rules and constraints in $\Pi_{sel}^{aux} \cup \Pi_{sel}^{o}$, as well as all ground instances of the following rules:

(i) the defaultization r^{Δ} of r, for each $r \in \Pi_{sel}^{c}$;
(ii) the *structural rule*

$$\neg query\_source(S, Q) \leftarrow query\_source(S', Q), S \neq S'; \tag{3}$$

and
(iii) the *default-context rules*

$$default\_class(O, C, Q) \leftarrow cref(O, C, \_, Q),$$
$$default\_path(O, P, Q) \leftarrow cref(O, \_, P, Q).$$

Intuitively, the defaultization makes the selection rules in Π_{sel} defeasible with respect to the predicate *query_source*, the structural rule enforces that only one source is selected, and the default-context rules define the two default predicates. Since our language has no function symbols, Π_Q is finite, and its size depends on the constants appearing in Π_{qa}, $R(Q)$, Π_{sd}, and Π_{dom}.

Definition 6
For $\mathscr{S} = (\Pi_{qa}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u)$ and query Q, we call any answer set of $\Pi_{qa} \cup R(Q) \cup \Pi_{sd} \cup \Pi_{dom}$ a *selection input of* $\mathscr{S}$ *for* Q. The set of all selection inputs of $\mathscr{S}$ for Q is denoted by $Sel(\mathscr{S}, Q)$. For $Y \in Sel(\mathscr{S}, Q)$, we define

$$Y_{def} = Y \cup \{default\_class(o, c, q), default\_path(o, p, q) \mid cref(o, c, p, q) \in Y\}.$$

Note that, in general, a selection base may admit multiple selection inputs for a query Q. However, in many cases, there may exist only a single selection input—in particular, if the source description Π_{sd} and the domain knowledge Π_{dom} have unique answer sets. In our framework, this is ensured if, e.g., these components are represented by (locally) stratified programs.

Definition 7
Given a selection base $\mathscr{S} = (\Pi_{qa}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u)$ and a query Q, a rule $r \in \Pi_Q$ is *relevant for* Q iff there is some $Y \in Sel(\mathscr{S}, Q)$ such that $B^{\dagger}(r)$ is true in Y_{def}, where $B^{\dagger}(r)$ results from $B(r)$ by deleting each element which does not contain a predicate symbol from $\mathscr{A}_{qd} \cup \mathscr{A}_{sd} \cup \mathscr{A}_{dom} \cup \{default\_class, default\_path\}$.

In the sequel, we denote for any binary relation R its transitive closure by R^*.

[6] Defaultization is also known in the literature as the *extended version* of a rule (Kowalski and Sadri 1990; Van Nieuwenborgh and Vermeir 2002).

We continue with the construction of the preference relation $<$, used for interpreting a source-selection program $(\Pi_{sel}, <_u)$ relative to a selection base $\mathscr{S}$ and a query Q in terms of an associated prioritized logic program $(\Pi_Q, <)$.

Informally, the specification of $<$ depends on the following auxiliary relations:

- the preference relation $\preceq_c$, taking care of implicit context priorities;
- the intermediate relation $\trianglelefteq$, representing a direct combination of user-defined preferences with context preferences; and
- the preference relation $<'$, removing possible conflicts within the joined relation $\trianglelefteq$ and ensuring transitivity of the resultant order $<$.

More specifically, the relation $\preceq_c$ is the first step towards $<$, transforming structural context information into explicit preferences, in virtue of the following specificity conditions:

- default contexts for concepts are assumed to be more specific than default contexts for attributes;
- context references are more specific than default contexts; and
- with respect to the same IRP, rules with a larger CRP (C, P) are considered more specific than rules with a shorter CRP (C', P') (i.e., where P' is a subpath of P).

The second step in the construction of $<$ is the relation $\trianglelefteq$, which is just the union of the user preferences $<_u$ and the context priorities $\preceq_c$. In general, this will not be a strict partial order. To enforce irreflexivity, we remove all tuples $n_{r_1} \trianglelefteq n_{r_2}$ lying on a cycle, resulting in $<'$. Finally, taking the transitive closure of $<'$ yields $<$. The formal definition of relation $<$ is as follows.

Definition 8
Let $\mathscr{S}$ be a selection base, Q a query, and Π_Q as in Definition 5. For $r_1, r_2 \in \Pi_Q$, define

(i) $r_1 \preceq_c r_2$ iff r_1 and r_2 are relevant for Q, $r_1 \neq r_2$, and one of (O_1)–(O_3) holds:

  (O_1) $\mathit{default\_path}(o_1, p_1, q) \in B(r_1)$, and either $\mathit{cref}(o_2, t_2, p_2, q) \in B(r_2)$ or $\mathit{default\_class}(o_2, t_2, q) \in B(r_2)$,

  (O_2) $\mathit{default\_class}(o_1, t_1, q) \in B(r_1)$ and $\mathit{cref}(o_2, t_2, p_2, q) \in B(r_2)$,

  (O_3) $\mathit{cref}(o, t_1, p_1, q) \in B(r_1)$, $\mathit{cref}(o, t_2, p_2, q) \in B(r_2)$, and t_1/p_1 is a subpath of t_2/p_2,

(ii) $r_1 \trianglelefteq r_2$ iff $r_1 <_u r_2$ and r_1 and r_2 are relevant for Q, or $r_1 \preceq_c r_2$, and

(iii) $r_1 <' r_2$ iff $r_1 \trianglelefteq r_2$ but not $r_2 \trianglelefteq^* r_1$.

Then, the relation $<$ is given as the transitive closure of $<'$.

Example 7
Reconsider $(\Pi_{sel}, <_u)$ from Example 6. Suppose the domain ontology contains the concepts "*MovieDB*", "*Actor*", "*Movie*", "*Director*", and "*Person*", and that "*Personalia*" and "*Person*" are synonymous. Assume further that the query of Example 1 (represented by q_1) has a unique selection input Y, containing the source-description facts

$$\mathit{accurate}(s_1, \text{“}\mathit{Director}\text{”}, \mathit{high}),\ \mathit{covers}(s_2, \text{“}\mathit{Person}\text{”}, \mathit{high}), \text{ and } \mathit{reliable}(s_3, \mathit{low}),$$

together with the following facts resulting from the query description and the default-context rules:

$$\begin{aligned}
&cref(o_2, \text{“}Person\text{”}, \text{“}FirstName\text{”}, q_1),\\
&cref(o_2, \text{“}Director\text{”}, \text{“}Personalia/FirstName\text{”}, q_1),\\
&cref(o_3, \text{“}Person\text{”}, \text{“}LastName\text{”}, q_1),\\
&cref(o_3, \text{“}Director\text{”}, \text{“}Personalia/LastName\text{”}, q_1),\\
&selects(o_3, equal, \text{“}Hitchcock\text{”}),\\
&default\_class(o_2, \text{“}Person\text{”}, q_1),\\
&default\_class(o_3, \text{“}Person\text{”}, q_1),\\
&default\_class(o_3, \text{“}Director\text{”}, q_1),\\
&default\_path(o_3, \text{“}LastName\text{”}, q_1).
\end{aligned}$$

These elements are exactly those contributing to relevant instances of Π_Q. The relevant instances of r_1, r_2, and r_3 are given by the ground rules $r_1(q_1, o_2)$, $r_1(q_1, o_3)$, $r_2(q_1, o_3)$, and $r_3(q_1, s_1, o_3, \text{“}D\text{”})$.[7] Intuitively, we expect $r_2(q_1, o_3)$ to have highest priority among these rule instances, since the bodies of the instances of r_1 and r_3 contain default predicates while r_2 references a specific context. Actually, the order relation $<$ includes, for the relevant instances of r_1, r_2, and r_3, the pairs $r_1(q_1, o_2) < r_2(q_1, o_3)$, $r_1(q_1, o_3) < r_2(q_1, o_3)$, and $r_3(q_1, s_1, o_3, \text{“}D\text{”}) < r_2(q_1, o_3)$.

Note that both r_4 and r_5 have two relevant instances. However, they do not influence the above rule ordering. Informally, they are either unrelated to or “ranked between” $r_2(q_1, o_3)$ and the relevant instances of r_1 and r_3 (since the *cref* predicates of r_4 and r_5 refer to the same context as the context referenced in the body of r_2, or to a subpath of such a context). Hence, the relevant instance of r_2 has highest priority.

As for r_1 and r_3, the auxiliary relation $\trianglelefteq$ contains two further structural priorities, namely $r_3(q_1, s_1, o_3, \text{“}D\text{”}) \preceq_c r_1(q_1, o_2)$ and $r_3(q_1, s_1, o_3, \text{“}D\text{”}) \preceq_c r_1(q_1, o_3)$. They are in conflict with the user preferences $r_1(q_1, o_2) <_u r_3(q_1, s_1, o_3, \text{“}D\text{”})$ and $r_1(q_1, o_3) <_u r_3(q_1, s_1, o_3, \text{“}D\text{”})$, respectively. This is resolved in the resultant relation $<$ by removing these preferences.

Note that, in Definition 8, the final order $<$ enforces a cautious conflict resolution strategy, in the sense that it remains “agnostic” with respect to priority information causing conflicts. Alternative definitions of $<'$, such as removal of a minimal cutset eliminating all cycles in $\trianglelefteq$, may be considered as well; however, this may lead to a nondeterministic choice since, in general, multiple such cutsets exist. Different choices lead to different orders $<$, which may lead to different results of the source-selection program. Thus, unless a well-defined *specific* minimal cutset is singled out, by virtue of preference conflicts, the result of the source-selection process might not be deterministic. Furthermore, an extended logic program component computing a final order based on minimal cutsets is more involved than a component computing the relations in Definition 8.

[7] For brevity, we write here and in the remainder of this example “*D*” for “*Director*”.

Combining Definitions 5 and 8, we obtain the translation $\mathscr{E}(\cdot,\cdot)$ as follows:

Definition 9
Let $\mathscr{S}$ be a selection base and Q a query. Then, the *evaluation* $\mathscr{E}(\mathscr{S},Q)$ *of* $\mathscr{S}$ *with respect to* Q is given by the prioritized logic program $(\Pi_Q,<)$, where Π_Q and $<$ are as in Definitions 5 and 8, respectively.

Selection answer sets of source-selection programs are then obtained as follows:

Definition 10
Let $\mathscr{S} = (\Pi_{qa},\Pi_{sd},\ \Pi_{dom},\Pi_{sel},<_u)$ be a selection base, Q a query, and $\mathscr{E}(\mathscr{S},Q) = (\Pi_Q,<)$ the evaluation of $\mathscr{S}$ with respect to Q. Then, $X \subseteq Lit_{sel}$ is a *selection answer set of* $(\Pi_{sel},<_u)$ *for* Q *with respect to* $\mathscr{S}$ iff X is a preferred answer set of the prioritized logic program $(\Pi_Q \cup Y,<)$, for some $Y \in Sel(\mathscr{S},Q)$.

A source s is *selected for* Q iff $query\_source(s,q)$ belongs to some selection answer set of $(\Pi_{sel},<_u)$ for Q (with respect to $\mathscr{S}$), where the constant q represents Q.

Example 8
In our running example, $(\Pi_{sel},<_u)$ has a unique selection answer set X with respect to $\mathscr{S}$ for query q_1 from Example 6. It contains $query\_source(s_1,q_1)$, which is derived from the core rule $r_2(q_1,o_3)$, having the highest priority among the applicable rules leading to a single preferred answer set for the weak-constraint free part of Π_{sel}. If we replace, e.g., r_1 by the rule

$$query\_source(s_2,Q) \leftarrow cref(O,\text{“}Person\text{”},P,Q)$$

and adapt the corresponding user preference to $r_1(Q,\_,\_) <_u r_3(Q,\_,\_,\_)$, then the weak-constraint free part of Π_{sel} has two preferred answer sets: one, X_1, is identical to X (where applying $r_2(q_1,o_3)$ is preferred to applying $r_1(q_1,o_3)$, given that $r_1(q_1,o_3) < r_2(q_1,o_3)$); in the other answer set, X_2, the rule $r_1(q_1,o_2)$ is applied and $query\_source(s_2,q_1)$ is derived. Informally, the replacement removes the preference of $r_2(q_1,o_3)$ over $r_1(q_1,o_2)$, since the corresponding *cref* predicates refer to different contexts (“*.../FirstName*” and “*.../LastName*”, respectively). Thus, $r_1(q_1,o_2)$ has maximal preference like $r_2(q_1,o_3)$.

Given that X_1 has weight 5, caused by violation of $c_2(s_1,q_1,\text{“}Person\text{”})$, but X_2 has weight 10, caused by violation of $c_1(s_2,q_1,\text{“}Director\text{”})$, X_1 is the selection answer set of $(\Pi_{sel},<_u)$ for Q.

6.3 Properties

In this section, we discuss some basic properties of our framework.

The first property links our evaluation method of source-selection programs to the usual semantics of prioritized logic programs. For this purpose, we introduce the following concept: Given logic programs Π_1 and Π_2, we say that Π_1 is *independent* of Π_2 iff each predicate symbol occurring in some rule head of Π_2 does not occur in Π_1. Intuitively, if Π_1 is independent of Π_2, then Π_1 may serve as an “input” for Π_2. This idea is made precise by the following proposition, which is an immediate consequence of results due to Eiter *et al.* (1997) and Lifschitz and Turner (1994):

Proposition 1
Let Π_1 and Π_2 be two extended logic programs, possibly containing weak constraints, and let X be a set of ground literals. If Π_1 is independent of Π_2, then X is an answer set of $\Pi_1 \cup \Pi_2$ iff there is some answer set Y of Π_1 such that X is an answer set of $\Pi_2 \cup Y$.

Now, taking the specific structure of our source-selection architecture into account, we obtain the following characterization.

Theorem 1
Suppose $\mathscr{S} = (\Pi_{qa}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u)$ is a selection base and Q a query. Let $\mathscr{E}(\mathscr{S}, Q) = (\Pi_Q, <)$ and $\Pi_{\mathscr{S}}(Q) = \Pi_{qa} \cup R(Q) \cup \Pi_{dom} \cup \Pi_{sd} \cup \Pi_Q$. Then, X is a selection answer set of $(\Pi_{sel}, <_u)$ for Q with respect to $\mathscr{S}$ iff X is a preferred answer set of $(\Pi_{\mathscr{S}}(Q), <)$.

Proof
Let Π_0 denote the program $\Pi_{qa} \cup R(Q) \cup \Pi_{sd} \cup \Pi_{dom}$. Recall that X is a selection answer set of $(\Pi_{sel}, <_u)$ for Q with respect to $\mathscr{S}$ iff X is a preferred answer set of $(\Pi_Q \cup Y, <)$, for some answer set Y of Π_0. Since the predicate symbols occurring in the heads of rules in a source-selection program do not occur in rules from the query description, the source description, or the domain theory, we obviously have that Π_0 is independent of Π_Q. Moreover, it holds that X is a preferred answer set of $(\Pi_Q \cup Y, <)$ only if X is an answer set of $\Pi_Q \cup Y$. Hence, applying Proposition 1, we have that X is an answer set of $\Pi_Q \cup Y$, for some answer set Y of Π_0, iff X is an answer set of $\Pi_Q \cup \Pi_0$. From this, the assertion of the theorem is an immediate consequence. $\square$

We remark that from a logic programming point of view, Theorem 1 might seem to be a more natural definition of selection answer sets. However, our approach is motivated by providing a high-level means for specifying source-selection problems, which is accomplished by decomposition. Note, in particular, that a user will only need to specify the relation $<_u$ as opposed to $<$. Hence, the property of Theorem 1 shall rather be understood as a possibility to "compile" a selection base and selection inputs with respect to a query into a single logic program.

Strengthening Theorem 1, the construction of $\mathscr{E}(\mathscr{S}, Q)$ can itself be realized in terms of a *single* logic program of the form $\Pi_{\mathscr{S}}(Q) \cup \Pi_{obj}(Q)$ over an extended vocabulary, by describing preference relations *directly at the object level*, such that each answer set encodes the priority relation $<$ and is a preferred answer set of $(\Pi_{\mathscr{S}}(Q) \cup \Pi_{obj}(Q), <)$ if and only if its restriction to Lit_{sel} is a selection answer set of $(\Pi_{sel}, <_u)$ for Q with respect to $\mathscr{S}$. More details about this property are given in Appendix D.

Concerning the computational complexity of source selection, we note that, given a query Q and the grounding of the program $\Pi_{\mathscr{S}}(Q)$ for a selection base $\mathscr{S}$ as in Theorem 1, deciding whether $(\Pi_{sel}, <_u)$ has some selection answer set for Q is NP-complete (since the grounding of $\Pi_{obj}(Q)$ can be constructed in polynomial time from the grounding of $\Pi_{\mathscr{S}}(Q)$), and computing any such selection answer set is complete for $\mathrm{FP}^{\mathrm{NP}}$, which is the class of all problems solvable in polynomial time with an NP oracle. However, for a fixed selection base and small query size (which is a common assumption for databases), the problems are solvable in *polynomial time* (cf. again Appendix D for more details about the complexity of source-selection programs).

One of the desiderata of our approach is that each answer set selects at most one source, for any query Q. The following result states that this property is indeed fulfilled.

Theorem 2
Let X be a selection answer set of $(\Pi_{sel}, <_u)$ for query Q with respect to $\mathscr{S}$. Then, for any constant q, it holds that

$$|\{s \mid query\_source(s, q) \in X\}| \leqslant 1.$$

Proof
The presence of the structural rule (3) in the evaluation program Π_Q enforces that, whenever X contains two ground atoms $query\_source(s, q)$ and $query\_source(s', q)$, X must be inconsistent, and thus X violates the consistency criterion of answer sets. □

Lastly, the following result concerns the order of application of source-selection rules, stating that source selection is blocked in terms of priorities as desired.

Theorem 3
Let X be a selection answer set of $(\Pi_{sel}, <_u)$ for query Q with respect to $\mathscr{S}$, and let $r^{\Delta} \in \Pi_Q$ be the defaultization of some rule r belonging to the grounding of Π^{c}_{sel} for Q with respect to $\mathscr{S}$. Suppose that $B(r)$ is true in X but $H(r) \notin X$. Then, there is some $r' \in \Pi_Q$ such that

(i) either r' belongs to the grounding of Π^{aux}_{sel} for Q with respect to $\mathscr{S}$ and $H(r') = \neg H(r)$, or r' is the defaultization of a rule from the grounding of Π^{c}_{sel} for Q with respect to $\mathscr{S}$,
(ii) $B(r')$ and $H(r')$ are true in X, and
(iii) either r^{Δ} and r' are incompatible with respect to $<$, or else $r^{\Delta} < r'$ holds.

Proof
Given that X is a selection answer set of $(\Pi_{sel}, <_u)$ for query Q with respect to $\mathscr{S}$, we have that X is a preferred answer set of the prioritized logic program $(\Pi_Q \cup Y, <)$, where Y is some selection input of $\mathscr{S}$ for query Q, and, *a fortiori*, that X is an answer set of $\Pi_Q \cup Y$. From the latter and the hypothesis that $H(r) \notin X$, it follows that $r^{\Delta} = H(r) \leftarrow B(r), not\ \neg H(r)$ is not a member of $GR(X, \Pi_Q \cup Y)$. Hence, in view of the assumption that $B(r)$ is true in X, we get that $\neg H(r) \in X$ must hold.

Since X is a preferred answer set of $(\Pi_Q \cup Y, <)$, there is some enumeration $\langle r_i \rangle_{i \in I}$ of $GR(X, \Pi_Q \cup Y)$ such that Conditions (P_1)–(P_3) hold (cf. Section 2). We take $r' = r_{\ell}$, where ℓ is as follows. Given that $\neg H(r) \in X$, there is a smallest index $i_0 \in I$ such that $r_{i_0} \in GR(X, \Pi_Q \cup Y)$ and $H(r_{i_0}) = \neg H(r)$. If r_{i_0} belongs to the grounding of Π^{aux}_{sel} for Q with respect to $\mathscr{S}$, then $\ell = i_0$. Otherwise, by the syntactic form of a source-selection program, r_{i_0} must be an instance of the structural rule (3). By (P_1)–(P_3), the defaultization $\bar{r}$ of a rule from the grounding of Π^c_{sel} for Q with respect to $\mathscr{S}$ must exist such that $\bar{r} = r_{j_0} \in GR(X, \Pi_Q \cup Y)$ and $j_0 < i_0$. In this case, $\ell = j_0$.

We show that r' satisfies Conditions (i)–(iii). Clearly, Condition (i) is satisfied. Furthermore, Condition (ii) is an immediate consequence of the fact that $r' \in GR(X, \Pi_Q \cup Y)$. It remains to show that Condition (iii) holds.

Towards a contradiction, assume that $r' < r^\Delta$. Since $r' \in GR(X, \Pi_Q \cup Y)$ and $r^\Delta \notin GR(X, \Pi_Q \cup Y)$, from Condition (P_3) we get that $B^-(r^\Delta) \cap \{H(r_k) \mid k < \ell\} \neq \emptyset$, as $B^+(r) \subseteq X$ and $B^+(r^\Delta) = B^+(r)$. Now, obviously $\{H(r_k) \mid k < \ell\} \subseteq X$. Moreover, since $B^-(r) \cap X = \emptyset$ and $B^-(r^\Delta) = B^-(r) \cup \{\neg H(r)\}$, we obtain that $\neg H(r) \in \{H(r_k) \mid k < \ell\}$. Hence, there must be some $k_0 < \ell$ such that $r_{k_0} \in GR(X, \Pi_Q \cup Y)$ and $H(r_{k_0}) = \neg H(r)$. But this contradicts the condition that i_0 $(\geqslant \ell)$ is the smallest index i such that $r_i \in GR(X, \Pi_Q \cup Y)$ and $H(r_i) = \neg H(r)$. Hence, we either have that r' and r^Δ are incompatible with respect to $<$, or $r^\Delta < r'$ must hold. □

6.4 Extended source selection

The semantics of source-selection programs we defined so far aims at selecting at most one source. We can easily modify this definition, however, to accommodate also the selection of multiple sources at a time. To this end, we only have to modify the structural rule (3) in Definition 5 appropriately.

For example, using language elements provided by the DLV system (Leone *et al.* 2006; Faber *et al.* 2004), the simultaneous selection of up to a given number k of sources can be accomplished by replacing (3) with the following rules:

$$\begin{aligned} \mathit{false} &\leftarrow \mathit{not\ false},\ \mathit{query}(Q),\ \mathit{max\_sources}(K),\\ &\qquad \#\mathit{count}\{S' : \mathit{query\_source}(S', Q)\} > K,\\ \neg\mathit{query\_source}(S, Q) &\leftarrow \mathit{source}(S),\ \mathit{query}(Q),\ \mathit{max\_sources}(K),\\ &\qquad 1 <= \#\mathit{count}\{S' : \mathit{query\_source}(S', Q)\} <= K,\\ &\qquad \mathit{not\ \ query\_source}(S, Q), \end{aligned}$$

where $\mathit{max\_sources}(K)$ holds for $K = k$. Here, $\#\mathit{count}\{S' : \mathit{query\_source}(S', Q)\}$ is an *aggregate expression* which singles out the number of all sources S' for which an instance of $\mathit{query\_source}(S', Q)$ is in the answer set, and "<" and "<=" are comparison built-ins. This modification can also be expressed with (ordinary) ELPs as introduced in Section 2, but is more involved then.

The setting of selecting a "best" source with a single selection result can be easily generalized to a setting with multiple, ranked selection results—in particular, to the computation of all outcomes with a cost valuation within a given distance d to a given value, as well as to the computation of the k best outcomes, for a given integer k, akin to *range queries* and *k-nearest neighbor queries*, respectively, in information retrieval. Such ranked computations can be orthogonally combined with the type of selection outcome (i.e., single source vs. up to a number of sources). Furthermore, they can be easily accomplished using the features of the underlying DLV system.

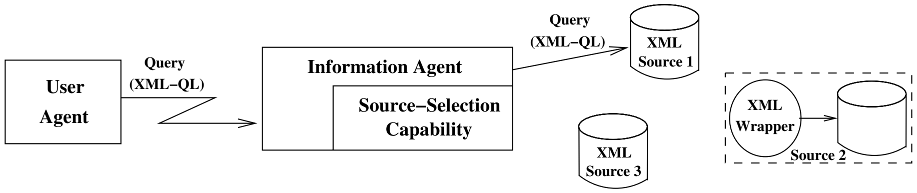

Fig. 2. Architecture of a simple agent-based source-selection system.

7 Implementation and application

7.1 Implementation

We have implemented our source-selection approach on top of the DLV system (Leone *et al.* 2006) and its front-end plp (Delgrande *et al.* 2001) for prioritized logic programs.[8] The evaluation of source-selection programs proceeds in three steps: (i) the set of all selection inputs for a query Q is computed from $R(Q)$, Π_{qa}, Π_{sd}, and Π_{dom}, using DLV (cf. Theorem 4 in Appendix D); (ii) a call to DLV calculates the priority relation $<$ from the set of selection inputs and Π_{sel}; and (iii) the answer sets of $(\Pi_Q, <)$ are determined by employing plp and DLV. Note that this three-step approach might appear to be overly complex, given that computing a selection answer set is feasible in polynomial time with an NP oracle (see Section 6.3 and Theorem 5 in Appendix D), and one might wonder why DLV (which can handle Σ_2^P-complete problems) is called several times. The reason for proceeding in this fashion is that it actually greatly improves the performance since, due to built-in optimization techniques of DLV, groundings can be kept smaller.

The entire process is implemented as an ECLiPSe Prolog program, which served as a rapid prototyping language, and is independent of the actual query language. For XML-QL queries, however, a query parser, written in C++, for generating the low-level representation $R(Q)$ of a query Q has been developed. A query parser for SQL queries is also available (Schindlauer 2002) and further languages can be deployed in the same way.

We have also "agentized" the source-selection system using the *IMPACT* agent platform (Subrahmanian *et al.* 2000), enabling the realization of source-selection agents which may also issue the execution of XML-QL queries on XML data sources. A generic agent-based source-selection setup, as implemented in *IMPACT*, is shown in Figure 2. Data are stored in XML databases, and queries are posed in an XML query-language such as XML-QL. Some of the databases may be wrapped from non-XML data sources. A query is handed over to an information agent, which has to pick one of several databases that comply with the same (universal) schema to answer the query.

The architecture in Figure 2 is only one of several possible agent-based architectures; others may be as follows:

- there may be multiple information agents in a system, avoiding a centralization bottleneck;

[8] Details about DLV and plp can also be found at `http://www.dlvsystem.com` and `http://www.cs.uni-potsdam.de/ torsten/plp`, respectively.

- the source-selection capability may be realized not in terms of a special source-selection agent, but being part of a more powerful mediator agent; or
- the sources may be accessed through specialized wrapper agents, which control access and might refuse requests.

7.2 An application for movie databases

As an application domain, we considered the area of movie databases, and we have built an experimental environment for source selection in this domain, using the prototype implementation described above.

7.2.1 Movie sources

We used the *Internet Movie Database* (*IMDb*) as the main source for raw data, as well as the *EachMovie Database* provided by Compaq Computer Corporation,[9] to generate a suite of XML movie databases. To this end, (parts of) the large databases were wrapped offline to XML, using a DTD (provided in Appendix A) which we modeled from a set of relevant movie concepts captured by the *Open Directory Project*.[10] The XML databases we constructed are the following:

RandomMovies (RM): This source contains data about numerous movies, randomly wrapped from the IMDb. Besides title and language information (always having value “English”), each item comprises, where available, entries containing genre classification, the release date, the running time, review ratings, the names of the two main actors, directors, and screenwriters, as well as details about the soundtrack (listing, in some cases, the name of the composer of the soundtrack).

RandomPersons (RP): Like RM, RandomPersons is derived from the IMDb, containing randomly wrapped data about numerous actors, directors, screenwriters, and some composers. Besides names, person data comprise the date and country of birth, and a biography, and may, as for RM, again be incomplete.

EachMovie (EM): Wrapped from Compaq’s EachMovie Database, this source stores English movies plus ratings. For most entries, it provides genre information, and for half of them a release date (after 1995). It has no information about actors, directors, soundtracks, etc., however.

Hitchcock (HC): Wrapped from the IMDb, this source stores all movies directed by Alfred Hitchcock, in the format of RandomMovies but with all involved actors listed. For each person, it also contains information (if available) about the date and country of birth, and a biography.

KellyGrant (KG): Similar to HC, this source stores the titles of all movies in which either Grace Kelly or Cary Grant were actors, as well as the names of all persons involved.

Horror60 (H60): Being the last of our databases, H60 is a collection of horror movies from the 1960s, as found in the IMDb. Movie and person data are as before, but almost no soundtrack or composer information is stored.

[9] These two databases are available at `http://www.imdb.org` and `http://www.research.compaq.com/SRC/eachmovie`, respectively.

[10] See `http://dmoz.org`.

The information about these databases is stored in the source-description program Π_{sd}, using the predicates introduced in Section 5. For illustration, we list some elements of this program, modeling one of the sources, and refer to Eiter *et al.* (2003) and Fink (2002) for a detailed account of the complete program Π_{sd}.

Example 9
For providing information about database KellyGrant, the program Π_{sd} contains the following facts:

$$\begin{array}{l}
source(s\_KellyGrant);\\
up(s\_KellyGrant);\\
data\_format(s\_KellyGrant, xml);\\
update\_frequency(s\_KellyGrant, low);\\
specialized(s\_KellyGrant, \text{“}Kelly\text{”});\\
specialized(s\_KellyGrant, \text{“}Grant\text{”});\\
covers(s\_KellyGrant, \text{“}Movie\text{”}, low);\\
covers(s\_KellyGrant, c, high), \text{ for } c \in \{\text{“}ReleaseDate\text{”}, \text{“}Person\text{”}, \text{“}BirthDate\text{”},\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{“}Actor\text{”}\};\\
covers(s\_KellyGrant, fifties, high);\\
covers(s\_KellyGrant, sixties, high);\\
\neg relevant(s\_KellyGrant, p), \text{ for } p \in \{seventies, eighties, nineties, twothousands\}.
\end{array}$$

Informally, Π_{sd} expresses that KellyGrant is an XML source which is (currently) up and rarely updated. It is specialized in topics “*Kelly*” and “*Grant*”, and has high coverage about persons, especially actors, and their birth dates, but provides low coverage about movies in general. However, it highly covers the release dates of the stored movies, most of which are from the fifties and sixties. Further information about KellyGrant is derived from default rules like the ones given below, stating that English is the default language for all sources:

$$\begin{array}{rl}
source\_language(S, \text{“}English\text{”}) \leftarrow & source(S),\\
& not\ \neg source\_language(S, \text{“}English\text{”});\\
\neg source\_language(S, \text{“}English\text{”}) \leftarrow & source\_language(S, L), L \neq \text{“}English\text{”}.
\end{array}$$

7.2.2 *Domain knowledge*

The ontology part of the domain knowledge, Π_{dom}, includes the facts

$$\begin{array}{l}
class(O), \text{ for } O \in \{\text{“}MovieDB\text{”}, \text{“}Movie\text{”}, \text{“}Director\text{”}, \text{“}Actor\text{”}, \text{“}Screenwriter\text{”},\\
\qquad\qquad \text{“}Composer\text{”}, \text{“}Person\text{”}, \text{“}Soundtrack\text{”}, \text{“}Review\text{”}\},
\end{array}$$

as they may be extracted from the XML DTD, and the fact

$$synonym(\text{“}Personalia\text{”}, \text{“}Person\text{”}).$$

The attributes of the concept “*Movie*” are given by the facts

$$\begin{array}{l}
class\_att(\text{“}Movie\text{”}, att), \text{ where } att \in \{title, alternativeTitles, genre, releaseDate,\\
\qquad\qquad\qquad\qquad\qquad\qquad runningTime, language, review\}.
\end{array}$$

For example, a concrete instance of "*Movie*" is given by $instance(m12, \text{“}Movie\text{”})$. For further details, cf. Eiter *et al.* (2003) or Fink (2002).

The background part of Π_{dom} serves to formalize "common-sense" knowledge of the application domain, which is an important source of information for the selection process. This part is usually quite extensive. On the one hand, it contains rules capturing typical relationships between ontological concepts, and, on the other hand, it comprises "well-known" instances of these concepts. For space reasons, we only show a few rules of Π_{dom} here. We note in passing that this part also implements a simple form of reasoning about time, viz. reasoning about *decades*, by associating every year since 1920 its corresponding decade.

Example 10

Some (typical) rules from the background knowledge are:

$$
\begin{aligned}
s_1: \quad & instance(P, \text{“}Director\text{”}) \leftarrow directed(P, M); \\
s_2: \quad & involved(P, M) \leftarrow directed(P, M); \\
s_3: \quad & life\_period(P, B, E) \leftarrow instance(P, \text{“}Person\text{”}), not\ dead(P), \\
& \qquad att\_val(P, birthDate, B_1), current\_year(E), \\
& \qquad calender\_year(B_1, B); \\
s_4: \quad & possible\_genre(M, G) \leftarrow involved(P, M), default\_genre(P, G), \\
& \qquad not\ defined\_genre(M).
\end{aligned}
$$

Intuitively, rules s_1 and s_2 infer, from a role *acted* between a person and a movie, that the corresponding person is an actor and that he or she is involved in the movie. Rule s_3 assigns a life period to a person from his or her birth date, while rule s_4 infers a possible genre for a movie, if an involved person and his or her default genre are known.

Furthermore, the following facts are representations of specific movie-historic incidents (actually, they model information about Grace Kelly and the movie "Arsenic and Old Lace"):

$$
\begin{aligned}
& instance(perKelly, \text{“}Actor\text{”}); \\
& att\_val(perKelly, name, nameKelly); \\
& att\_val(perKelly, birthDate, 1929); \\
& att\_val(perKelly, dateOfDeath, 1982); \\
& prod\_period(perKelly, 1945, 1960); \\
& instance(nameKelly, name); \\
& att\_val(nameKelly, firstName, \text{“}Grace\text{”}); \\
& att\_val(nameKelly, firstName, \text{“}Patricia\text{”}); \\
& att\_val(nameKelly, lastName, \text{“}Kelly\text{”}); \\
& instance(m12, \text{“}Movie\text{”}); \\
& att\_val(m12, title, \text{“}Arsenic\ and\ Old\ Lace\text{”}); \\
& att\_val(m12, releaseDate, 1944); \\
& acted(perGrant, m12).
\end{aligned}
$$

7.2.3 Source-selection program

The experimental movie source-selection program fills several pages and is too complex to be listed and discussed here in detail. Therefore, similar as before, we only give an informal description, highlighting the most important aspects, and refer to Eiter *et al.* (2003) and Fink (2002) for more details.

Among the source-selection rules, default rules have lowest priority and are used only in the core part. They make default suggestions for query sources in case no other core source-selection rule is eligible. Some examples are the following default rules:

$$
\begin{array}{lrcl}
r_1: & query\_source(S,Q) & \leftarrow & default\_path(O,P,Q), \\
 & & & occurs(O,V), specialized(S,P); \\
r_2: & query\_source(s\_RandomMovies,Q) & \leftarrow & default\_class(O,\text{“}Movie\text{”},Q); \\
r_3: & query\_source(s\_RandomPersons,Q) & \leftarrow & default\_class(O,\text{“}Person\text{”},Q).
\end{array}
$$

The first rule is generic, whilst the others are specific. Informally, r_1 advises to query source S if it is specialized for P, where P is some path of a reference in the query that is compared to some value. For example, suppose P is instantiated with *LastName*. If some source is specialized for last names, then it is chosen unless a source-selection rule with higher priority is applicable. Similarly, the specific rules r_2 and r_3 suggest to select RandomMovies or RandomPersons if the query entails a reference under object “*Movie*” or “*Person*”, respectively.

Non-default core source-selection rules also appear in either generic or specific form:

$$
\begin{array}{lrcl}
r_4: & query\_source(S,Q) & \leftarrow & source(S),\ query(Q), \\
 & & & high\_coverage(S,Q); \\
r_5: & query\_source(S,Q) & \leftarrow & source(S),\ query(Q),\ special(S,Q); \\
r_6: & query\_source(s\_Hitchcock,Q) & \leftarrow & cref(O_1,\text{“}Person\text{”},\text{“}LastName\text{”},Q), \\
 & & & selects(O_1,equal,\text{“}Hitchcock\text{”}), \\
 & & & cref(O_2,\text{“}Person\text{”},\text{“}FirstName\text{”},Q), \\
 & & & selects(O_2,equal,\text{“}Alfred\text{”}).
\end{array}
$$

The generic rules r_4 and r_5 suggest to query any source that highly covers the query or is special for it, respectively. The specific rule r_6 advises to query the source Hitchcock if a query selects a person named Alfred Hitchcock. Note that *high_coverage* and *special* are auxiliary predicates, defined by auxiliary rules (see below).

Since no *cref* predicate and no default predicates occur in r_4 and r_5, there is no (direct) structural precedence between them and rule r_6, as well as between r_1, r_2, and r_3. The following user preferences explicitly establish preferences among them:

$$\begin{aligned} r_1(\_,Q,\_,\_,\_) &<_u r_4(\_,Q); & r_4(\_,Q) &<_u r_5(\_,Q);\\ r_1(\_,Q,\_,\_,\_) &<_u r_5(\_,Q); & r_4(\_,Q) &<_u r_6(Q,\_,\_);\\ r_2(Q,\_) &<_u r_5(\_,Q); & r_5(\_,Q) &<_u r_6(Q,\_,\_).\\ r_3(Q,\_) &<_u r_5(\_,Q); \end{aligned}$$

Auxiliary rules are used to define auxiliary predicates as well as to filter irrelevant sources:

$$\begin{aligned} a_1:&\quad & special(S,Q) &\leftarrow special\_topic(S,Q,T);\\ a_2:&\quad & special\_topic(S,Q,T) &\leftarrow inferred\_topic(Q,T),\ specialized(S,T);\\ a_3:&\quad & inferred\_topic(S,Q,T) &\leftarrow matchingMovie(Q,M), involved(P,M),\\ &&& \quad\ att\_val(P,name,N), att\_val(N,lastName,T);\\ a_4:&\quad & \neg query\_source(S,Q) &\leftarrow irrelevant(S,Q);\\ a_5:&\quad & irrelevant(S,Q) &\leftarrow cref(O,\text{“}MovieDB\text{”},\\ &&& \qquad\quad \text{“}Movie/ReleaseDate/Date\text{”},Q),\\ &&& \quad\ selects(O,equal,V),\ calender\_year(V,Y),\\ &&& \quad\ decade(Y,D), \neg relevant(S,D). \end{aligned}$$

Informally, rule a_1 states that a source is special for a query if a topic associated with the query exists for which it is special. Rule a_2 expresses that one way to associate a topic to a query is to infer a topic, like, e.g., realized in terms of rule a_3. Hence, if the query accesses a movie that is known and T is the last name of a person involved in it, then a source is concluded to be special for that query if it is specialized for T. Rule a_4 states that a source must not be queried if it is irrelevant for a query; in view of rule a_5, this is the case if the source is not relevant for the decade in which movie has been released.

Finally, the quantitative part of the source-selection program has weak constraints like the following:

$$\begin{aligned} w_1:&\quad \Leftarrow\ query\_source(S,Q), default\_class(O,T,Q),\\ &\qquad\ constructs(O,C,P), covers(S_1,T,high),\ not\ covers(S,T,high)\ [3:1];\\ w_2:&\quad \Leftarrow\ query\_source(S,Q),\ high\_covered\_topic(S_1,Q,T),\\ &\qquad\ decade\_name(T),\ not\ high\_covered\_topic(S,Q,T)\ [1:1]. \end{aligned}$$

Intuitively, w_1 assigns a penalty of 3 per concept T that is asked (resp., constructed) by the query to any answer set which selects a source that does not highly cover T while a source highly covering T exists. Similarly, w_2 assigns a penalty of 1 per decade that is associated to the query to any answer set which selects a source which does not highly cover this decade while some other highly covering source exists.

7.3 Experiments

We tested the above movie-application scenario by means of a number of natural user queries. More specifically, our tests involved 18 queries, some of which are the following (for the complete list of queries, cf. Eiter *et al.* (2003) or Fink (2002)):

Table 1. *Experimental Results for the Movie Application*

Query	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
Candi-dates	HC	RM, H60, HC, KG	KG	HC	RM, HC, KG	RM, HC, KG	RM, H60, HC, KG	RP
Best	HC	RM	KG	HC	RM	RM	RM	RP

q_1: *Which movies were directed by Alfred Hitchcock?*

q_2: *In which movies, directed by Josef von Sternberg, did Marlene Dietrich act?*

q_3: *In which year has the movie "Arsenic and Old Lace" been released?*

q_4: *In which movies, directed by Alfred Hitchcock, did Marlene Dietrich act?*

q_5: *In which film noirs did Marilyn Monroe act?*

q_6: *In which movies did Laurel and Hardy act in 1940?*

q_7: *Which movies where Frank Sinatra appeared in have a soundtrack composed by Elmer Bernstein?*

q_8: *When was James Dean born?*

The formulation of these queries in XML-QL is straightforward (as a matter of fact, q_1 is expressed by the XML-QL query of Example 1; for the formulation of all queries in XML-QL, cf. Eiter *et al.* (2003) or Fink (2002)).

Source selection for the considered queries was performed employing the movie databases described above as well as variants thereof. Each process took from a couple to up to tens of seconds, which is due to the size of the programs involved. However, performance was not a central issue here. Since our implementation and the used tools are unoptimized, there is a large potential for performance improvements. Also, the underlying solvers might gain efficiency in future releases.

7.3.1 Results

The results of the source selection process for q_1–q_8, using the above source descriptions, are shown in Table 1. Note that, by the semantics of source-selection programs, per selection answer set and query, a single source is chosen. Thus, query decomposition is not considered here, although our method for computing a query description allows for it in principle. The entries show the sources which are selected by the different answer sets, where the labels "Candidates" and "Best" refer to selection with optimization part dropped (i.e., qualitative selection only) and enabled, respectively.

The results can be informally explained as follows. For q_1, a specific core source-selection rule, r_6, which has highest preference, fires and HC is chosen, as expected.

For q_2, there is some background knowledge about Marlene Dietrich, but no source can be found as being special for this query, while generic default source-selection rules trigger for all sources. Nonetheless, RP and EM are recognized

Table 2. *Experimental Results for an Extended Selection Base*

Query	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
Candidates	HC	RM, RMN, H60, HC, KG	KG	HC	RM, RMN, HC, KG	RM, RMN, HC, KG	RM, RMN, H60, HC, KG	RP
Best	HC	RMN	KG	HC	RM, RMN	RM, RMN	RM	RP

as being irrelevant for q_2 and eventually discarded: q_2 asks for (resp., ranges over) concepts these sources are not relevant for (viz. *"Movie"* and *"Person"*, respectively). The best source among the candidates RM, H60, HC, and KG is RM, since it is the only one highly covering the concept asked for.

For q_3, since "Arsenic and Old Lace" is in the background knowledge (cf. above), and since Cary Grant acted in it, we would expect KG to be queried. Indeed, this is what actually happens. It is not a specific core source-selection rule that triggers the selection (Cary Grant does not explicitly appear in the query), rather Grant is inferred as a query topic from the background knowledge and, thus, the generic core selection rule suggesting to query KG has highest priority.

Query q_4 is a refinement of q_1; the same specific core source-selection rule, r_6, as for q_1 triggers.

Similar as for q_2, RM is chosen for q_5, q_6, and q_7, but for the former two, H60 is recognized as being irrelevant on different grounds: q_5 asks for film noirs, and so H60, which contains horror movies, is eliminated by reasoning over genre information, while q_6 involves movies from 1940, and thus H60, which contains only movies produced in the 1960s, is excluded by reasoning over decades.

Finally, RP is chosen for q_8, as expected: a specific default source-selection rule triggers for RP, which has precedence over generic default rules that would trigger for other sources.

7.3.2 Results with modified selection bases

In a slightly different scenario, RM is designed to have high coverage about composers and western movies, too, and a new random movie source, RANDOM-MOVIESNEW (RMN), similar to RM, but with less coverage about genres, release dates, composers, and western movies, while having high coverage about directors, dramas, and comedies, is introduced. Respective changes to the source descriptions and the addition of a specific default source-selection rule for RMN (similar to rule r_2 for RM in Section 7.2.3) and corresponding user preferences to the source-selection program yield an "extended" selection base, for which the results for q_1–q_8 are shown in Table 2.

The change does not influence the results for q_1, q_3, q_4, and q_8. This is intuitive, since the suitability of the chosen sources is unaffected. For the other queries, the new source RMN is a further candidate, as the generic default source-selection rule

Table 3. *Experimental Results for a Reduced Selection Base*

Query	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
Candi-dates	HC	H60, HC, KG	KG	HC	HC, KG,	HC, KG	H60, HC, KG	RP
Best	HC	KG	KG	HC	KG	HC, KG	KG	RP

is also applicable to it. By a similar reason as before, RM and RMN are better than the other candidates. For q_2, RMN is ranked above RM: it highly covers dramas, which is an inferred query topic, since drama is a default genre for Marlene Dietrich in the background knowledge. RM is ranked above RMN for q_7 since RM highly covers composers, a concept occurring in the query (which asks for the composer Elmer Bernstein). RM and RMN are ranked equal for q_5 and q_6: they highly cover actors, but the background knowledge has no information about Laurel and Hardy; and that Marilyn Monroe's default genre is comedy has no consequence for q_5, as it explicitly asks for film noirs.

As a further modification, we considered a "reduced" selection base where RM is down and, thus, cannot be queried. The results are given in Table 3. The candidate sources remain the same, except that RM is missing; thus, the change has no impact on q_1, q_3, q_4, and q_8, as one would expect. For q_6, the optimization part imposes no preference between HC and KG; interestingly, it selects KG as being best for q_2, q_5, and q_7. This is because the background knowledge entails information about the productive period of Dietrich, Monroe, and Bernstein, which occur in the queries. Thus, the 1950s and 1960s are inferred as being relevant topics for these queries, and KG, covering both decades highly, outranks HC and H60, which highly cover only one of the decades each.

8 Related work

The selection of data sources is a component in many information-integration systems (cf., e.g., Arens *et al.* (1993), Bayardo *et al.* (1997), Garcia-Molina *et al.* (1997), Singh *et al.* (1997), Genesereth *et al.* (1997), and Levy *et al.* (1996); see also Levy and Weld (2000) and references therein). However, most center around mappings between a global scheme and local schemes, on query rewriting, and on query planning to optimally reconstruct dispersed information. Our work, instead, is concerned with *qualitative* selection from different alternatives, based on rich meta-knowledge and a formal semantics respecting preference and context information involving heuristic defaults, which is not an issue there. Furthermore, no form of query description similar as in our method is considered in these approaches.

In the following subsection, we review some of the above mentioned information-integration systems in more detail. Afterwards, we discuss approaches bearing a closer relation to our work.

8.1 Information integration systems

SIMS (Arens *et al.* 1993; Arens and Knoblock 1992; Arens *et al.* 1996), short for *Services and Information Management for Decision Systems*, is a data integration system which exploits a semantic model of a problem domain to integrate information dispersed over various heterogeneous information sources. The latter are typically databases, or, more generally, knowledge bases. The domain model is formulated in the Loom knowledge-representation language (MacGregor and Bates 1987), and comprises a declarative description of the objects and activities possible in the specific domain. SIMS aims at providing the user a transparent access to the data, without being aware of the underlying heterogenous data sources. It accepts user queries in the form of a description of a class of objects about which information is required. Any such query over the domain model is mapped to a query over the information sources, by translating the concepts of the domain to corresponding concepts in the data models of the information sources; if a direct translation does not exist, a query rewriting is performed, and, if needed, multiple databases are accessed in a query plan. SIMS strives for singling out optimal query plans, for which aspects such as costs of accessing the different sources and combining the results returned are taken into account. This is apparently different from the contributions of our work, which is concerned with selecting a single information source among a set of candidate sources. Furthermore, aspects of incomplete information and nonmonotonic constructs to overcome it were not addressed in SIMS, nor a method similar to query description.

The Carnot project at MCC (Singh *et al.* 1997; Collet *et al.* 1991; Huhns and Singh 1992) was an early effort to provide a logically unifying view of enterprise-wide, distributed, and possibly heterogeneous data. The Carnot system has a layered architecture, whose top layer consists of semantic services providing a suite of tools for enterprise modeling, model integration, data cleaning, and knowledge discovery. The *Model Integration and Semantics Tool* (MIST) is used for creating mappings between local schemas and a common ontology expressed in Cyc (Lenat and Guha 1990) or in a specific knowledge representation language, which is done once at the time of integration. Besides relational databases, also knowledge-based systems (with an extensional part containing facts and an intensional part containing rules) may be integrated and, moreover, play a mediator role between applications and different databases. As an important feature, local database schemas remain untouched, and queries to them are translated to the global schema and back to (other) local schemas for data retrieval. Similar to SIMS, Carnot aims at providing a uniform and consistent view of heterogeneous data. A selection of information sources for query answering, based on similar criteria and methods as in our approach, is not evident.

InfoSleuth (Bayardo *et al.* 1997; Fowler *et al.* 1999; Nodine *et al.* 2003), which has its roots in Carnot, is an agent-based system for information discovery and retrieval in a dynamic, open environment, broadening the focus of database research to the challenge of the World-Wide Web. It extends the capabilities of Carnot to an environment in which the identities of the information sources need not be known at the time of generating the mapping. In this approach, agents are the constituents of the systems, whose knowledge and their relationships to each other are described in an InfoSleuth ontology. Decisions about user-query decompositions are based on a domain ontology, which is selected by the user and describes knowledge about the relationships of the data stored in the sources that subscribe to the ontology. As for selection of information sources, special broker agents provide, upon request, information about which resource agents (i.e., information sources behind them) should be accessed for specific information sought. The broker performs a *semantic matchmaking* of the user request with the service descriptions of the provider agents (which may be viewed as an advanced yellow-pages service), aiming at ruling out, by means of constraints (e.g., over the range of values, existing attributes, etc.), all sources which will return a nil result. To this end, it must reason over explicitly advertised information about agent capabilities to determine which agent can provide the requested services. The broker translates KIF statements into queries in the LDL++ deductive database language, which are submitted to an LDL++ engine for evaluation. In this way, rule-based matching is facilitated. Our approach differs significantly from InfoSleuth, and is in fact to some extent complementary to it. Indeed, the descriptions of constraints and other semantic criteria in InfoSleuth for selecting an information source are at a very low level. Even if the LDL++ language, which can emulate non-stratified negation via choice rules, is used for rule-based matchmaking, there is no special support for dealing with contexts, user preferences, or optimization constructs as in our approach. Furthermore, it is not evident that InfoSleuth agents are programmed using a declarative language which provides similar functionalities for discriminating among different sources compliant with the constraints. Instead, our formalism might be mapped to LDL++ by a suitable transformation and thus provide a plug-in module for realizing semantically richer and refined brokering in InfoSleuth with a well-defined, formal semantics and provable properties.

The Information Manifold (Kirk *et al.* 1995; Levy *et al.* 1995; Levy *et al.* 1996) is a system for browsing and querying multiple networked information sources. Its architecture is based on a rich domain model which enables the description of properties of the information sources, such as their addresses, the protocols used to access them, their structure, etc., using a combination of the CLASSIC description logic (Borgida *et al.* 1989), Horn rules, and integrity constraints. An external information source is viewed as containing extensions of a collection of relations, on which integrity constraints may be imposed, and which are semantically mapped by rules to the relations in the global knowledge base. Information sources may be associated with topics, allowing to classify the former along a hierarchy of topics in the domain model. This mechanism can be used for deciding retrieval of a source for related queries. Like in SIMS, the user may pose queries in a high-level language on the global schema, which are mapped to queries over the local sources. The Information Manifold focuses on optimizing the execution of a user query, accessing as few information sources as necessary, where relevance is judged on criteria involving the (static) semantic mapping, and on combining the results. However, no qualitative selection similar to the one in our approach is made, and, in particular, no user preferences or nonmonotonic rules (including default contexts) can be expressed by constructs in the language.

Infomaster (Genesereth *et al.* 1997) provides integrated access to multiple distributed heterogeneous information sources on the Internet, which gives the illusion of a centralized, homogeneous information system in a virtual schema. The system handles both structural and content translations to resolve differences between multiple data sources and the multiple applications for the collected data, where mappings between the information sources and the global schema are described by rules and constraints. The user may pose queries on the virtual schema, which are first translated to queries over base relations at the information sources and then further rewritten to queries over site relations, which are views on the base relations, by applying logical abduction. The core of Infomaster is a facilitator that dynamically determines an efficient way to answer the user's query employing as few sources as necessary and harmonizes the heterogeneities among these sources. However, like in the other information systems above, neither rich meta-data about the quality of information sources is considered, nor preferences or context information is used to heuristically discriminate between optional choices.

8.2 Other work

More related to our approach than the methods in the previous subsection is the work by Huffman and Steier (1995), which outlines an interactive tool for information specialists in query design. It relieves them from searching through data-source specifications and can suggest sources to determine trade-offs. However, no formal semantics or richer domain theories, capable of handling incomplete and default information, is presented.

Remotely related to our work are the investigations by Fuhr (1999), presenting a decision-theoretic model for selecting data sources based on retrieval cost and typical information-retrieval parameters.

Goto *et al.* (2001) consider a problem setting related to ours, where source descriptions include semantic knowledge about the source. In contrast to our work, however, a query is viewed merely as a set of terms, and a source description is a thesaurus automatically constructed from the documents of the source. A further thesaurus, WordNet (Fellbaum 1998), is used for the source evaluation algorithm, which is based on the calculation of weighted similarity measures. The main differences to our approach are that the selection method is not declarative and just numeric, semantic knowledge is limited to a thesaurus, and no further background knowledge, reasoning, or semantic query analysis is involved.

Semantic analysis of queries has been incorporated to document retrieval by Wendlandt and Driscoll (1991). Starting from conventional information-retrieval methods that accept natural-language queries against text collections and calculate similarity measures for query keywords, semantic modeling was introduced by trying to detect entity attributes and thematic roles from the query to the effect of a modified similarity computation. While richer ontological knowledge than thesauri is used, source descriptions have no semantic knowledge. Again, the approach is not declarative but numeric in nature, and neither rich domain theories nor automated reasoning is involved.

FAQ FINDER (Burke *et al.* 1995) is a natural-language question-answering system that uses files of frequently-asked questions (FAQ) as its knowledge base. It uses standard information-retrieval methods to narrow the search to one FAQ file and to calculate a term-vector metric for the user's question and question/answer pairs. Moreover, it uses a comparison of question types in a taxonomy derived from the query, and a semantic similarity score in question matching. The latter is calculated by passing through the hypernym links, i.e., is-a links, through WordNet.

Recent proposals for Web-based information retrieval built on ontology-based agents which search for, maintain, and mediate relevant information for a user or other agents are discussed by Luke *et al.* (1997), Sim and Wong (2001), and Chen and Soo (2001). More specifically, Sim and Wong (2001) describe a society of software agents where query-processing agents assist users in selecting Web pages. They search for URLs using search engines and ontological WordNet relations for query specialization or generalization to keep the number of located, relevant URLs within given limits. An architecture for ontology-based information-gathering agents appears also in the work of Chen and Soo (2001), but here special domain search engines and Web documents are used as well. Ontologies are represented in a usual object-oriented language, and queries are partial instances of ontological concepts.

9 Conclusion

In this paper, we have presented a knowledge-based approach for information-source selection, using meta-knowledge about the quality of the sources for determining a "best" information source to answer a given query, which is posed in a formal query language (as considered here, XML-QL). We have described a rule-based language for expressing source-selection policies in a fully declarative way, which supports reasoning tasks that involve different components such as background and ontological knowledge, source descriptions, and query constituents. Furthermore, the language provides a number of features which have proven valuable in the context of knowledge representation, viz. the capability of dealing with incomplete information, default rules, and preference information.

We have developed a novel method for automated query analysis at a generic level in which interesting information is distilled from a given query expression in a formal query language, as well as an approach to preference handling in source selection, which combines implicit rule priorities, given by the context of rule applications, and explicit user preferences. As pointed out previously, context-based rule application is a different concept as inheritance-based reasoning—to the best of our knowledge, no similar approach for handling default-context rules has been considered before. We presented a formal model-theoretic semantics of our approach, which is based on the answer-set semantics of extended logic programs. Furthermore, we analyzed semantical and computational properties of our approach, where we showed that source-selection programs possess desirable properties which intuitively should be satisfied. We emphasize that for other, related approaches no similar results are evident, since lacking a formal semantics makes them less accessible to reason about their behavior.

The results that we have obtained in the implementation of the experimental movie application are encouraging, and suggest several directions for further work. One issue concerns the supply of rich background and common-sense knowledge. The coupling with available ontology and common-sense engines via suitable interfaces is suggestive for this purpose. Extensions of logic programs under the answer-set semantics allowing such a coupling have been realized, e.g., by Eiter *et al.* (2004, 2005b, 2005a). Also, other recent efforts aim at mapping description logics underlying different ontology languages to logic programs (Grosof *et al.* 2003; Motik *et al.* 2003; Swift 2004.

Another direction for further work involves the application of our results in the context of information integration and query systems. They might be valuable for enriching semantic brokering in open agent-based systems, but also for more traditional closed systems in which information sources must be manually registered. In particular, the advanced information-integration methods, employing extended logic programming tools, developed within the INFOMIX project is a natural candidate for incorporating a heuristic source-selection component.[11]

Our results are also relevant for adaptive source selection which is customized, e.g., by user profiles. This subject is important for realizing personalized information systems in a dynamic environment, which, to a large extent, involve user preferences and reasoning with incomplete information and defaults, as well as dynamic updates of source descriptions.

Acknowledgments

We would like to thank the referees for their helpful and constructive comments which helped improving the presentation of this paper. This work was partially supported by the Austrian Science Fund (FWF) under grants P13871-INF and Z29-INF, as well as by the European Commission under projects FET-2001-37004 WASP, IST-2001-33570 INFOMIX, and the IST-2001-33123 CologNeT Network of Excellence.

Appendix A The XML DTD for the movie databases

```
<!ELEMENT MovieDB (Movie|Actor|Director|Screenwriter|
                   Composer|Person|Award|Filmfestival)*>
<!ELEMENT Movie (Title,AlternativeTitle*,ReleaseDate?,
                 RunningTime?,Culture?,LeadingRole*,Role*,Actor*,
                 Director*,Screenwriter*,Soundtrack*,Review*,Award*)>
```

[11] See `http://sv.mat.unical.it/infomix/` for details about INFOMIX.

```
<!ATTLIST Movie
          Genre (Action|Animation|Classic|Comedy|CowboyWestern|
                 CultMovie|Documentary|Experimental|FilmNoir|
                 Horror|Romance|SciFiFantasy|Series|Silent|Travel|Other)
         #IMPLIED Language CDATA "English">
<!ELEMENT Person (FirstName*,LastName,BirthDate?,Country?,Biography?)>
<!ATTLIST Person ID ID #REQUIRED Gender (male|female) #IMPLIED>
<!ELEMENT Award (AwardTitle,Date,AwardType?,AwardCategory?)>
<!ELEMENT Character (#PCDATA)>
<!ELEMENT Filmfestival (#PCDATA)>
<!ELEMENT Actor (Award*)>
<!ATTLIST Actor Personalia IDREF #REQUIRED>
<!ELEMENT Director (Award*)>
<!ATTLIST Director Personalia IDREF #REQUIRED>
<!ELEMENT Screenwriter (Award*)>
<!ATTLIST Screenwriter Personalia IDREF #REQUIRED>
<!ELEMENT Composer (Award*)>
<!ATTLIST Composer Personalia IDREF #REQUIRED>
<!ELEMENT Soundtrack (Title,Composer*,Award*)>
<!ELEMENT Biography (#PCDATA)>
<!ELEMENT AlternativeTitle (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT LastName (#PCDATA)>
<!ELEMENT BirthDate (Date)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT AwardCategory (#PCDATA)>
<!ELEMENT AwardType (#PCDATA)>
<!ELEMENT AwardTitle (#PCDATA)>
<!ELEMENT ReleaseDate (Date)>
<!ELEMENT RunningTime (#PCDATA)>
<!ELEMENT LeadingRole (Character,Award*)>
<!ATTLIST LeadingRole Actor IDREF #REQUIRED>
<!ELEMENT Role (Character,Award*)>
<!ATTLIST Role Actor IDREF #REQUIRED>
<!ELEMENT Review (ReviewText,Rating?)>
<!ELEMENT ReviewText (#PCDATA)>
<!ELEMENT Rating (#PCDATA)>
<!ELEMENT Culture (#PCDATA)>
```

Appendix B Query description

In what follows, we provide details about the query description predicates and the query-analysis program.

B.1 Low-level predicates

The query and its syntactic subqueries are named by constants (e.g., $q_1, q_2, \ldots$). The facts $R(Q)$ are formed using the following predicates:

- $\mathit{sub\_query}(Q', Q)$: Q' is a structural subquery of query Q (possibly itself a subquery);
- $\mathit{query\_cand}(Q)$: identifies the overall query;
- $\mathit{source}(S, Q)$: query Q accesses source S;
- $\mathit{db\_name}(S)$: source S is a database;
- $\mathit{whereRef}(O, T, P, Q)$: an IRP O references an item under element T and remaining path P in the where part of query Q;
- $\mathit{subpath}(O, T_1, P_1, T_2, P_2)$: the path T_1/P_1 is a direct subpath of T_2/P_2 in the IRP O;
- $\mathit{whereRefCmp}(O_1, R, O_2)$: the items of IRPs O_1 and O_2 are compared using operator R;
- $\mathit{whereCmp}(O, R, V)$: the item of IRP O is compared to value V using operator R;
- $\mathit{consRef}(O, T, P)$: the item of IRP O is constructed under element T and remaining path P in the (answer) construction part of query Q.

$R(Q)$ must respect that query languages may allow for nested queries. However, in a query expression, an outermost query as the "root" of nesting should be identifiable, as well as structural (syntactic) subqueries of it. They are described using *query_cand* and *sub_query*, respectively.

Along an IRP, item references relative to a position are captured by the *whereRef* predicate, and suffix inclusions for this IRP are stored as *subpath* facts. The predicates *whereRefCmp* and *whereCmp* mirror the comparison of two items and the comparison of an item with a value, respectively. Items that occur in the construction part of a query are also identified by an IRP and stored using *consRef*.

Example 11
The low-level representation $R(Q)$ of the query in Example 1 contains

$\mathit{sub\_query}(q_2, q_1)$, $\mathit{query\_cand}(q_1)$, $\mathit{source}(\text{“}\mathit{MovieDB}\text{”}, q_2)$, and $\mathit{db\_name}(\text{“}\mathit{MovieDB}\text{”})$,

and, e.g., for the third IRP, o_3, which references "*LastName*", the facts:

$\mathit{whereRef}(o_3, \text{“}\mathit{LastName}\text{”}, \text{“ ”}, q_2)$;
$\mathit{whereRef}(o_3, \text{“}\mathit{Personalia}\text{”}, \text{“}\mathit{LastName}\text{”}, q_2)$;
$\mathit{whereRef}(o_3, \text{“}\mathit{Director}\text{”}, \text{“}\mathit{Personalia/LastName}\text{”}, q_2)$;
$\mathit{whereRef}(o_3, \text{“}\mathit{Movie}\text{”}, \text{“}\mathit{Director/Personalia/LastName}\text{”}, q_2)$;
$\mathit{whereRef}(o_3, \text{“}\mathit{MovieDB}\text{”}, \text{“}\mathit{Movie/Director/Personalia/LastName}\text{”}, q_2)$;
$\mathit{subpath}(o_3, \text{“}\mathit{LastName}\text{”}, \text{“ ”}, \text{“}\mathit{Personalia}\text{”}, \text{“}\mathit{LastName}\text{”})$;
$\mathit{subpath}(o_3, \text{“}\mathit{Personalia}\text{”}, \text{“}\mathit{LastName}\text{”}, \text{“}\mathit{Director}\text{”}, \text{“}\mathit{Personalia/LastName}\text{”})$;
$\mathit{subpath}(o_3, \text{“}\mathit{Director}\text{”}, \text{“}\mathit{Personalia/LastName}\text{”}, \text{“}\mathit{Movie}\text{”},$
$\text{“}\mathit{Director/Personalia/LastName}\text{”})$;

$subpath(o_3, \text{“}Movie\text{”}, \text{“}Director/Personalia/LastName\text{”}, \text{“}MovieDB\text{”},$
$\quad \text{“}Movie/Director/Personalia/LastName\text{”});$
$whereCmp(o_3, equal, \text{“}Hitchcock\text{”}).$

The complete low-level representation $R(Q)$ of the query is given in Appendix C (cf. also Eiter *et al.* (2003) or Fink (2002)).

B.2 High-level predicates

The following high-level description predicates are defined:

- $query(Q)$: identifies an “independent” (sub-)query Q (i.e., Q is executable on some source), which is, moreover, not a purely syntactic subquery (i.e., which is not embraced by a sourceless query Q' merely restructuring the result of Q; for details, cf. the explanation of rules qa_8–qa_{12} of Π_{qa} below);
- $cref(O, C, P, Q)$: states that (C, P) is a CRP for Q via IRP O in the where-part of Q;
- $occurs(O, V)$: the value V is associated with an IRP O in the overall query;
- $selects(O, R, V)$: like *occurs*, but details the association with a comparison operator R;
- $constructs(O, I, P)$: states that the item of IRP O, by use of a variable, also appears in the construct-part of the global query, as an item I under path P (which may be different from the path in the where-part);
- $joins(O_1, O_2, R)$: records (theta-)joins of (or within) queries between IRPs O_1 and O_2 under comparison operator R.

Example 12
For the query in Example 1, we have $query(q_1)$ but not $query(q_2)$, since the embracing query q_1 has no source and merely structures the result of q_2. The following *cref* facts result from o_1 and o_3:

$$\begin{array}{l}
cref(o_1, \text{“}MovieDB\text{”}, \text{“}Movie/Title\text{”}, q_1);\\
cref(o_1, \text{“}Movie\text{”}, \text{“}Title\text{”}, q_1);\\
cref(o_3, \text{“}MovieDB\text{”}, \text{“}Movie/Director/Personalia/LastName\text{”}, q_1);\\
cref(o_3, \text{“}Movie\text{”}, \text{“}Director/Personalia/LastName\text{”}, q_1);\\
cref(o_3, \text{“}Director\text{”}, \text{“}Personalia/LastName\text{”}, q_1);\\
cref(o_3, \text{“}Person\text{”}, \text{“}LastName\text{”}, q_1).
\end{array}$$

Here, “*MovieDB*”, “*Movie*”, “*Director*”, and “*Person*” are concepts given by the ontology, and “*Personalia*” is known to be a synonym of “*Person*” (cf. Appendix B.3 for further discussion).

The fact $occurs(o_3, \text{“}Hitchcock\text{”})$ states that value *Hitchcock* is associated with o_3. This is detailed by $selects(o_3, equal, \text{“}Hitchcock\text{”})$, where *equal* represents equality. For the *constructs* predicate, the fact $constructs(o_1, \text{“}Movie\text{”}, \text{“ ”})$ is included. There are no *joins* facts since the query has no join. The complete high-level description is given in Appendix C (cf. also Eiter *et al.* (2003) or Fink (2002)).

B.3 Query-analysis program

The query-analysis program Π_{qa} is composed of the following groups of rules. The first rules enlarge the low-level predicate *subpath* as follows:[12]

$$\begin{aligned}
qa_1 :\quad & subpath(O,T_1,P_1,T_3,P_3) \leftarrow subpath(O,T_1,P_1,T_2,P_2),\\
& \qquad\qquad\qquad\qquad\qquad\quad subpath(O,T_2,P_2,T_3,P_3);\\
qa_2 :\quad & subpath(O,T,P_1,T_2,P_2) \leftarrow subpath(O,L,P_1,T_2,P_2), synonym(L,T);\\
qa_3 :\quad & subpath(O,T_1,P_1,T,P_2) \leftarrow subpath(O,T_1,P_1,L,P_2), synonym(L,T).
\end{aligned}$$

Rule qa_1 expresses transitivity for elements occurring in paths, and qa_2 and qa_3 deal with synonyms, which is imported ontological knowledge; *synonym* applies to all pairs of synonymous element names (e.g., names of IDREF attributes[13]).

The following two rules define useful projections of low-level predicates:

$$\begin{aligned}
qa_4 :\quad & has\_source(Q) \leftarrow source(\_,Q);\\
qa_5 :\quad & is\_sub\_query(Q) \leftarrow sub\_query(Q,\_).
\end{aligned}$$

Using them, an auxiliary predicate *iquery_cand* is defined for candidates which may satisfy the *query* predicate; these are the overall query and subqueries having a database or a document as its source:

$$\begin{aligned}
qa_6 :\quad & iquery\_cand(Q) \leftarrow query\_cand(Q);\\
qa_7 :\quad & iquery\_cand(Q) \leftarrow is\_sub\_query(Q), source(Z,Q), db\_name(Z).
\end{aligned}$$

Concerning the high-level predicates, independent, separate queries are specified by respecting the nesting structure:

$$\begin{aligned}
qa_8 :\quad & query(Q) \leftarrow top\_query(Q,Q);\\
qa_9 :\quad & top\_query(Q,Q) \leftarrow iquery\_cand(Q), not\ is\_sub\_query(Q);\\
qa_{10} :\quad & top\_query(Q,Q) \leftarrow iquery\_cand(Q), sub\_query(Q,S), source(Z,S);\\
qa_{11} :\quad & top\_query(S,Q) \leftarrow sub\_query(S,Z), iquery\_cand(S), top\_query(Z,Q),\\
& \qquad\qquad\qquad\qquad not\ has\_source(Z);\\
qa_{12} :\quad & top\_query(S,Q) \leftarrow sub\_query(S,Z), not\ iquery\_cand(S),\\
& \qquad\qquad\qquad\qquad top\_query(Z,Q).
\end{aligned}$$

Rule qa_8 expresses the property that a query is considered to be independent if it is the topmost independent query of itself. This is the case if the query is a candidate for a separate query and it is either the outermost query (dealt with by Rule qa_9) or a direct structural subquery of a query to a source (expressed by Rule qa_{10}). Moreover, qa_{10} intuitively states that a candidate query nested within another query is viewed as a separate query only if the nesting was not for purely syntactic reasons, i.e., it has its own source. In case of a purely syntactic subquery, or if a nested query is not a candidate for a separate query, its topmost

[12] In a clean separation of $R(Q)$ and the high-level description, a fresh predicate would be in order here. However, it is convenient and economic to re-use the predicate *subpath*, as it is only enlarged.

[13] If in a DTD an attribute is declared of type IDREF, this means that its value is the identifier of another element.

independent query is the one of the embracing query, as taken care of qa_{11} and qa_{12}, respectively.

The next rules define the remaining high-level description predicates. The auxiliary predicate *has_constructs* guarantees that at least one *constructs* fact is generated for each context reference constructed in the query answer.

$$\begin{array}{lrl}
qa_{13}: & \mathit{cref}(O,T,P,Q) & \leftarrow \mathit{whereRef}(O,T,P,S), \mathit{class}(T), \\
& & \quad \mathit{top\_query}(S,Q); \\
qa_{14}: & \mathit{cref}(O,T,P,Q) & \leftarrow \mathit{whereRef}(O,L,P,S), \mathit{synonym}(L,T), \\
& & \quad \mathit{class}(T), \mathit{top\_query}(S,Q); \\
qa_{15}: & \mathit{constructs}(O,T,P) & \leftarrow \mathit{consRef}(O,T,P), \mathit{class}(T); \\
qa_{16}: & \mathit{constructs}(O,T,P) & \leftarrow \mathit{consRef}(O,L,P), \mathit{synonym}(L,T), \mathit{class}(T); \\
qa_{17}: & \mathit{has\_constructs}(O) & \leftarrow \mathit{consRef}(O,T,P), \mathit{class}(T); \\
qa_{18}: & \mathit{has\_constructs}(O) & \leftarrow \mathit{consRef}(O,L,P), \mathit{synonym}(L,T), \mathit{class}(T); \\
qa_{19}: & \mathit{constructs}(O,\text{`` ''},\text{`` ''}) & \leftarrow \mathit{consRef}(O,\_,\_), \mathit{not}\ \mathit{has\_constructs}(O); \\
qa_{20}: & \mathit{occurs}(O,V) & \leftarrow \mathit{whereCmp}(O,C,V); \\
qa_{21}: & \mathit{selects}(O,C,V) & \leftarrow \mathit{whereCmp}(O,C,V); \\
qa_{22}: & \mathit{joins}(O_1,O_2,C) & \leftarrow \mathit{whereRefCmp}(O_1,C,O_2).
\end{array}$$

Note that some rules reference the ontology predicate *class*. A fact *class*(e) should exist in (or being entailed by) the domain ontology for all elements e that are considered to be concepts.

When queries are joined over CRPs, then some of the occurrence, selection, and construction information of one CRP is also valid for the other. Hence, we can build a form of a closure over joined CRPs, which is expressed by the following rules:

$$\begin{array}{lrl}
qa_{23}: & \mathit{constructs}(O_1,T,P) & \leftarrow \mathit{joins}(O_1,O_2,\mathit{equal}), \mathit{constructs}(O_2,T,P); \\
qa_{24}: & \mathit{constructs}(O_2,T,P) & \leftarrow \mathit{joins}(O_1,O_2,\mathit{equal}), \mathit{constructs}(O_1,T,P); \\
qa_{25}: & \mathit{occurs}(O_1,V) & \leftarrow \mathit{joins}(O_1,O_2,C), \mathit{occurs}(O_2,V); \\
qa_{26}: & \mathit{occurs}(O_2,V) & \leftarrow \mathit{joins}(O_1,O_2,C), \mathit{occurs}(O_1,V); \\
qa_{27}: & \mathit{selects}(O_1,C,V) & \leftarrow \mathit{joins}(O_1,O_2,\mathit{equal}), \mathit{selects}(O_2,C,V); \\
qa_{28}: & \mathit{selects}(O_2,C,V) & \leftarrow \mathit{joins}(O_1,O_2,\mathit{equal}), \mathit{selects}(O_1,C,V); \\
qa_{29}: & \mathit{selects}(O_1,\mathit{notequal},V) & \leftarrow \mathit{joins}(O_1,O_2,\mathit{notequal}), \mathit{selects}(O_2,\mathit{equal},V); \\
qa_{30}: & \mathit{selects}(O_2,\mathit{notequal},V) & \leftarrow \mathit{joins}(O_1,O_2,\mathit{notequal}), \mathit{selects}(O_1,\mathit{equal},V).
\end{array}$$

We remark that, as easily seen, the rules of Π_{qa} form a locally stratified logic program, and thus $\mathit{Ont} \cup \Pi_{qa} \cup R(Q)$ has a unique answer set.

Example 13
Let us consider how the high-level fact $\mathit{cref}(o_3,\text{“}\mathit{Person}\text{”},\text{“}\mathit{LastName}\text{”},q_1)$ is derived in Π_{qa}, given $R(Q)$ of the query in Example 1.

Since $\mathit{query\_cand}(q_1)$ is in $R(Q)$, we obtain, by qa_6, $\mathit{iquery\_cand}(q_1)$. Since the fact $\mathit{is\_sub\_query}(q_1)$ is not derivable, qa_{10} yields $\mathit{top\_query}(q_1,q_1)$ (i.e., stating that q_1 is independent). Next, we can derive $\mathit{is\_sub\_query}(q_2)$ by means of qa_5, and thus

$iquery\_cand(q_2)$ in view of qa_7, given that $R(Q)$ includes the facts $sub\_query(q_2, q_1)$, $source(\text{“}MovieDB\text{”}, q_2)$, and $db\_name(\text{“}MovieDB\text{”})$. Since q_1 has no source (i.e., $has\_source(q_1)$ is not derivable), we can derive $top\_query(q_2, q_1)$ from qa_9. The fact $cref(o_3, \text{“}Person\text{”}, \text{“}LastName\text{”}, q_1)$ is now derived by means of qa_{14}, making use of $whereRef(o_3, \text{“}Personalia\text{”}, \text{“}LastName\text{”}, q_2)$ from $R(Q)$, together with the facts $synonym(\text{“}Personalia\text{”}, \text{“}Person\text{”})$ and $class(\text{“}Person\text{”})$ from the ontology, and the derived fact $top\_query(q_2, q_1)$. Note that $cref(o_3, \text{“}Personalia\text{”}, \text{“}LastName\text{”}, q_1)$ is not derivable, as $class(\text{“}Personalia\text{”}) \notin Ont$.

Appendix C Query-representation for Example 1

The low-level representation $R(Q)$ of the query in Example 1 comprises the following facts:

$db\_name(\text{“}MovieDB\text{”});$
$query\_cand(q_1);$
$sub\_query(q_2, q_1);$
$source(\text{“}MovieDB\text{”}, q_2);$
$whereRef(o_1, \text{“}Title\text{”}, \text{“ ”}, q_2);$
$whereRef(o_1, \text{“}Movie\text{”}, \text{“}Title\text{”}, q_2);$
$whereRef(o_1, \text{“}MovieDB\text{”}, \text{“}Movie/Title\text{”}, q_2);$
$subpath(o_1, \text{“}Title\text{”}, \text{“ ”}, \text{“}Movie\text{”}, \text{“}Title\text{”});$
$subpath(o_1, \text{“}Movie\text{”}, \text{“}Title\text{”}, \text{“}MovieDB\text{”}, \text{“}Movie/Title\text{”});$
$whereRef(o_2, \text{“}FirstName\text{”}, \text{“ ”}, q_2);$
$whereRef(o_2, \text{“}Personalia\text{”}, \text{“}FirstName\text{”}, q_2);$
$whereRef(o_2, \text{“}Director\text{”}, \text{“}Personalia/FirstName\text{”}, q_2);$
$whereRef(o_2, \text{“}Movie\text{”}, \text{“}Director/Personalia/FirstName\text{”}, q_2);$
$whereRef(o_2, \text{“}MovieDB\text{”}, \text{“}Movie/Director/Personalia/FirstName\text{”}, q_2);$
$subpath(o_2, \text{“}FirstName\text{”}, \text{“ ”}, \text{“}Personalia\text{”}, \text{“}FirstName\text{”});$
$subpath(o_2, \text{“}Personalia\text{”}, \text{“}FirstName\text{”}, \text{“}Director\text{”}, \text{“}Personalia/FirstName\text{”});$
$subpath(o_2, \text{“}Director\text{”}, \text{“}Personalia/FirstName\text{”}, \text{“}Movie\text{”},$
$\quad \text{“}Director/Personalia/FirstName\text{”});$
$subpath(o_2, \text{“}Movie\text{”}, \text{“}Director/Personalia/FirstName\text{”}, \text{“}MovieDB\text{”},$
$\quad \text{“}Movie/Director/Personalia/FirstName\text{”});$
$whereRef(o_3, \text{“}LastName\text{”}, \text{“ ”}, q_2);$
$whereRef(o_3, \text{“}Personalia\text{”}, \text{“}LastName\text{”}, q_2);$
$whereRef(o_3, \text{“}Director\text{”}, \text{“}Personalia/LastName\text{”}, q_2);$
$whereRef(o_3, \text{“}Movie\text{”}, \text{“}Director/Personalia/LastName\text{”}, q_2);$
$whereRef(o_3, \text{“}MovieDB\text{”}, \text{“}Movie/Director/Personalia/LastName\text{”}, q_2);$
$subpath(o_3, \text{“}LastName\text{”}, \text{“ ”}, \text{“}Personalia\text{”}, \text{“}LastName\text{”});$
$subpath(o_3, \text{“}Personalia\text{”}, \text{“}LastName\text{”}, \text{“}Director\text{”}, \text{“}Personalia/LastName\text{”});$
$subpath(o_3, \text{“}Director\text{”}, \text{“}Personalia/LastName\text{”}, \text{“}Movie\text{”},$
$\quad \text{“}Director/Personalia/LastName\text{”});$

$subpath(o_3,$"*Movie*","*Director / Personalia / LastName*";
"*MovieDB*","*Movie / Director / Personalia / LastName*");
$whereCmp(o_2, equal,$"*Alfred*");
$whereCmp(o_3, equal,$"*Hitchcock*");
$consRef(o_1,$"*Movie*"," ");
$consRef(o_1,$"*MovieList*","*Movie*").

The high-level description, except for auxiliary predicates and the completion of the subpath predicates, is given by the following facts:

$query(q_1)$;
$cref(o_1,$"*MovieDB*","*Movie / Title*"$, q_1)$;
$cref(o_1,$"*Movie*","*Title*"$, q_1)$;
$cref(o_2,$"*MovieDB*","*Movie / Director / Personalia / FirstName*"$, q_1)$;
$cref(o_2,$"*Movie*","*Director / Personalia / FirstName*"$, q_1)$;
$cref(o_2,$"*Director*","*Personalia / FirstName*"$, q_1)$;
$cref(o_2,$"*Person*","*FirstName*"$, q_1)$;
$cref(o_3,$"*MovieDB*","*Movie / Director / Personalia / LastName*"$, q_1)$;
$cref(o_3,$"*Movie*","*Director / Personalia / LastName*"$, q_1)$;
$cref(o_3,$"*Director*","*Personalia / LastName*"$, q_1)$;
$cref(o_3,$"*Person*","*LastName*"$, q_1)$;
$occurs(o_2,$"*Alfred*"$), occurs(o_3,$"*Hitchcock*");
$selects(o_2, equal,$"*Alfred*");
$selects(o_3, equal,$"*Hitchcock*");
$constructs(o_1,$"*Movie*"," ").

Appendix D Further properties of source-selection programs

In order to realize the construction of $\mathscr{E}(\mathscr{S}, Q)$ in terms of a single logic program, we introduce a set N of constants serving as names for rules, and a new binary predicate $pref(\cdot,\cdot)$, defined over N, expressing preference between rules. The extended vocabulary $\mathscr{A}_{sel} \cup \{pref(\cdot,\cdot)\} \cup N$ is denoted by $\bar{\mathscr{A}}_{sel}$. We furthermore assume an injective function $n(\cdot)$ which assigns to each rule $r \in \Pi_Q$ a name $n(r) \in N$. To ease notation, we also write n_r instead of $n(r)$. Finally, Lit_{pref} denotes the set of all literals having predicate symbol *pref*. Note that $Lit_{pref} \cap Lit_{sel} = \emptyset$.

Theorem 4
Let $\mathscr{S} = (\Pi_{qa}, \Pi_{dom}, \Pi_{sd}, \Pi_{sel}, <_u)$ be a selection base, Q a query, and $\mathscr{E}(\mathscr{S}, Q) = (\Pi_Q, <)$. Furthermore, let $\Pi_{\mathscr{S}}(Q) = \Pi_{qa} \cup R(Q) \cup \Pi_{dom} \cup \Pi_{sd} \cup \Pi_Q$. Then, there exists a logic program $\Pi_{obj}(Q)$ over a vocabulary $\hat{\mathscr{A}} \supseteq \bar{\mathscr{A}}_{sel}$ such that every answer set X of $\Pi_{\mathscr{S}}(Q) \cup \Pi_{obj}(Q)$ satisfies the following conditions:

1. $X \cap Lit_{pref}$ represents $<$, i.e., $pref(n_r, n_{r'}) \in X$ iff $r < r'$; and
2. $X \cap Lit_{sel}$ is a selection answer set of $(\Pi_{sel}, <_u)$ for Q with respect to $\mathscr{S}$ iff X is a preferred answer set of the prioritized program $(\Pi_{\mathscr{S}}(Q) \cup \Pi_{obj}(Q), <)$.

Proof
We give a description of $\Pi_{obj}(Q)$ but omit a detailed argument that it satisfies the desired properties. Informally, $\Pi_{obj}(Q)$ consists of two parts: the first, Π_{rel}, is derived from $\Pi_{\mathscr{S}}(Q)$ and takes care of computing the relevant rules for Q, by utilizing weak constraints; the second part, Π_{pref}, is a (locally) stratified logic program determining the relations of Definition 8. We start with the construction of Π_{rel}.

For each predicate $p \in \mathscr{A}_{sel}$ and each $r \in \Pi_Q$, we introduce a new predicate $\overline{p}_r$ of the same arity as p. In addition, we introduce a new atom rel_r, informally expressing that rule r is relevant. If α is either a literal, a set of literals, a rule, or a program, then by $\lceil\alpha\rceil_r$ we denote the result of uniformly replacing each atom $p(x_1,,\ldots,x_n)$ occurring in α by $\overline{p}_r(x_1,\ldots,x_n)$.

For each $r \in \Pi_Q$, we define a program Π_r containing the following items:

1. each rule in $\lceil\Pi_{qa} \cup R(Q) \cup \Pi_{sd} \cup \Pi_{dom}\rceil_r$;
2. the rule $rel_r \leftarrow \lceil B^{\dagger}(r)\rceil_r$; and
3. the extended default-context rules

$$\overline{default\_class}_r(O,C,Q) \leftarrow \overline{cref}_r(O,C,\_,Q),$$
$$\overline{default\_path}_r(O,P,Q) \leftarrow \overline{cref}_r(O,\_,P,Q).$$

As easily checked, r is relevant for Q iff Π_r has some answer set containing rel_r. Now, Π_{rel} is defined as the collection of each of the programs Π_r, together with weak constraints of form

$$\Leftarrow \ not\ rel_r\ [1 : m+1], \tag{D 1}$$

for every $r \in \Pi_Q$, where m is the maximal priority level of the weak constraints occurring in Π^{o}_{sel}. Since, for any $r_1, r_2 \in \Pi_Q$ with $r_1 \neq r_2$, the programs Π_{r_1} and Π_{r_2} are defined over disjoint vocabularies, and given the inclusion of the weak constraints (D 1) in Π_{rel}, we obtain that Π_{rel} satisfies the following property:

$(*)$ for every answer set X of Π_{rel} and every $r \in \Pi_Q$, r is relevant for Q iff $rel_r \in X$.

These answer sets are used as inputs for the program Π_{pref}, which is defined next.

Let $pr(n,m)$ and $pr'(n,m)$ be new binary predicates, where n,m are names. Then, Π_{pref} consists of the following rules:

1. $pr(n_{r_1}, n_{r_2}) \leftarrow rel_{r_1}, rel_{r_2}$, for every $r_1, r_2 \in \Pi_Q$ such that either $r_1 <_u r_2$ or r_1, r_2 satisfy Conditions (O_1) or (O_2) of Definition 8;
2. $pr(n_{r_1}, n_{r_2}) \leftarrow rel_{r_1}, rel_{r_2}, subpath(o,t_1,p_1,t_2,p_2)$, for every $r_1, r_2 \in \Pi_Q$ such that $cref(o,t_1,p_1,q) \in B(r_1)$ and $cref(o,t_2,p_2,q) \in B(r_2)$; and
3. the rules

$$pr'(N_1,N_2) \leftarrow pr(N_1,N_2),$$
$$pr'(N_1,N_3) \leftarrow pr'(N_1,N_2), pr(N_2,N_3),$$
$$pref(N_1,N_2) \leftarrow pr(N_1,N_2), not\ pr'(N_2,N_1),$$
$$pref(N_1,N_3) \leftarrow pref(N_1,N_2), pref(N_2,N_3).$$

Obviously, Π_{pref} is a (locally) stratified program. Moreover, in view of Condition $(*)$, and since Π_{rel} is independent of Π_{pref} and $\Pi_{\mathscr{S}}(Q)$ is independent of $\Pi_{obj}(Q)$, for every answer set X of $\Pi_{\mathscr{S}}(Q) \cup \Pi_{obj}(Q) = \Pi_{\mathscr{S}}(Q) \cup \Pi_{rel} \cup \Pi_{pref}$, we have that (i) $pr(n_{r_1}, n_{r_2}) \in X$ iff $r_1 \trianglelefteq r_2$, (ii) $pr'(n_{r_1}, n_{r_2}) \in X$ iff $r_1 \trianglelefteq^* r_2$, and (iii) $pref(n_{r_1}, n_{r_2}) \in X$ iff $r_1 < r_2$. This proves Condition 1 of the theorem.

As for Condition 2, consider some answer set X of $\Pi_{\mathscr{S}}(Q) \cup \Pi_{obj}(Q)$. Since $\Pi_{\mathscr{S}}(Q)$ is independent of $\Pi_{obj}(Q)$, X is of form $Y \cup Y'$, where Y is an answer set of $\Pi_{\mathscr{S}}(Q)$ and Y' is a set of ground literals disjoint from Lit_{sel}. Hence, $X \cap Lit_{sel} = Y$. According to Theorem 1, Y is a selection answer set of $(\Pi_{sel}, <_u)$ for Q with respect to $\mathscr{S}$ iff Y is a preferred answer set of $(\Pi_{\mathscr{S}}(Q), <)$. But it is easily seen that the latter holds just in case $Y \cup Y'$ is a preferred answer set of $(\Pi_{\mathscr{S}} \cup \Pi_{obj}(Q), <)$. This proves the result. □

We note the following comments. First, Π_{rel} can be simplified by taking independence of subprograms of $\Pi_{\mathscr{S}}(Q)$ and possible uniqueness of answer sets for them into account. For example, if Π_{sd} has a unique answer set, then we may use in each program Π_r simply $\lceil \alpha \rceil_r = \alpha$, for each literal over $\mathscr{A}_{sd}$. In particular, if the program $\Pi_{qa} \cup R(Q) \cup \Pi_{dom} \cup \Pi_{sd}$ has a unique answer set (e.g., if this program is locally stratified), then we may simply take as Π_{rel} the program $\Pi_{qa} \cup R(Q) \cup \Pi_{dom} \cup \Pi_{sd}$ together with all rules $rel_r \leftarrow B^{\dagger}(r)$, for $r \in \Pi_Q$.

Second, the program $\Pi_{\mathscr{S}}(Q) \cup \Pi_{obj}(Q)$ in Theorem 4 represents, via preferred answer sets for a dynamic rule preference given by the atoms over *pref*, the selection answer sets of $(\Pi_{sel}, <_u)$ for Q. It can be easily adapted to a fixed program $\Pi'_{\mathscr{S}}$ such that, for any query Q, the dynamic preferred answer sets of $\Pi'_{\mathscr{S}} \cup R(Q)$ represent the selection answer sets of $(\Pi_{sel}, <_u)$ for Q (cf. Delgrande *et al.* (2003) for more details on dynamic preferences).

As for the complexity of source-selection programs, we can derive the following result as a consequence of Theorem 1.

Theorem 5

Given a query Q and the grounding of $\Pi_{\mathscr{S}}(Q) = \Pi_{qa} \cup R(Q) \cup \Pi_{dom} \cup \Pi_{sd} \cup \Pi_Q$, for a selection base $\mathscr{S} = (\Pi_{qa}, \Pi_{dom}, \Pi_{sd}, \Pi_{sel}, <_u)$, deciding whether $(\Pi_{sel}, <_u)$ has some selection answer set for Q with respect to $\mathscr{S}$ is NP-complete. Furthermore, computing any such selection answer set is complete for $\mathrm{FP}^{\mathrm{NP}}$.

Proof

Obviously, the groundings of the programs Π_{rel} and Π_{pref} in the proof of Theorem 4 are constructible in polynomial time from Q and the grounding of $\Pi_{\mathscr{S}}(Q)$, and so is the ground program Π', consisting of the groundings of $\Pi_{\mathscr{S}}(Q)$, Π_{rel}, and Π_{pref}. Furthermore, the preferred answer sets of $(\Pi', <)$ correspond to the selection answer sets of $(\Pi_{sel}, <_u)$. Since deciding whether a prioritized logic program (with no weak constraints) has a preferred answer set is NP-complete (Delgrande *et al.* 2003), it follows that deciding whether $(\Pi_{sel}, <_u)$ has a selection answer set for Q with respect to $\mathscr{S}$ is in NP. Note that the presence of weak constraints has no influence on the worst-case complexity of deciding the existence of (preferred) answer sets. Moreover, NP-hardness is immediate since the auxiliary rules can form any standard logic program.

From any answer set X of Π', an answer set of $(\Pi_{sel}, <_u)$ is easily computed. Computing such an X is feasible in polynomial time with an NP oracle, sketched as follows. First, compute the minimum vector of weak-constraint violations, v^*, i.e., the sum of weights of violated constraints at each level, using the oracle, performing binary search at each level, asking whether a violation limit can be obeyed. Then, build atom by atom an answer set X whose violation cost matches v^* using the NP oracle. Overall, this is possible in polynomial time with an NP oracle, hence the problem is in $\mathrm{FP}^{\mathrm{NP}}$.

The hardness for $\mathrm{FP}^{\mathrm{NP}}$ follows from a reduction given by Buccafurri *et al.* (2000), which shows how the lexicographic maximum truth assignment to a SAT instance, whose computation is well-known to be complete for $\mathrm{FP}^{\mathrm{NP}}$ (Krentel 1988), can be encoded in terms of the answer set of an ELP with weak constraints. □

Note that under *data complexity*, i.e., where the selection base $\mathscr{S}$ is fixed while the query Q (given by the facts $R(Q)$) may vary, the problems in Theorem 5 are in NP resp. $\mathrm{FP}^{\mathrm{NP}}$, since the grounding of $\Pi_{\mathscr{S}}(Q)$ is polynomial in the size of $\mathscr{S}$ and Q in this case. If, moreover, the size of Q is small and bounded by a constant, then the problems are solvable in *polynomial time,* since then the number of rules in the grounding of $\Pi_{\mathscr{S}}(Q)$ is bounded by some constant as well.

References

Abiteboul, S., Buneman, P. and Suciu, D. 2000. *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann, Los Altos.

Alferes, J., Pereira, L., Przymusinska, H. and Przymusinski, T. 2002. LUPS – A Language for Updating Logic Programs. *Artificial Intelligence 138,* 1–2, 87–116.

Apt, K., Blair, H. and Walker, A. 1988. Towards a Theory of Declarative Knowledge. See Minker (1988), 89–148.

Arens, Y., Chee, C., Hsu, C. and Knoblock, C. 1993. Retrieving and Integrating Data from Multiple Information Sources. *International Journal of Cooperative Information Systems 2,* 2, 127–158.

Arens, Y. and Knoblock, C. 1992. Planning and Reformulating Queries for Semantically-Modeled Multidatabase Systems. *Proceedings of the First International Conference on Information and Knowledge Managements*. 92–101.

Arens, Y., Knoblock, C. and Shen, W. 1996. Query Reformulation for Dynamic Information Integration. *Journal of Intelligent Information Systems 6,* 2–3, 99–130.

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving with Answer Sets.* Cambridge University Press.

Bayardo, R., Bohrer, B., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A. and Woelk, D. 1997. InfoSleuth: Semantic Integration of Information in Open and Dynamic Environments (Experience Paper). *Proceedings of the ACM SIGMOD International Conference on Management of Data* (*SIGMOD '97*). 195–206.

Borgida, A., Brachman, R. J., McGuinness, D. L. and Resnick, L. A. 1989. CLASSIC: A Structural Data Model for Objects. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (*SIGMOD '89*), J. Clifford, B. G. Lindsay, and D. Maier, Eds. ACM Press, 58–67.

Brewka, G. and Eiter, T. 1999. Preferred Answer Sets for Extended Logic Programs. *Artificial Intelligence 109,* 1–2, 297–356.

Buccafurri, F., Leone, N. and Rullo, P. 1996. Stable Models and their Computation for Logic Programming with Inheritance and True Negation. *Journal of Logic Programming 27,* 1, 5–43.

Buccafurri, F., Leone, N. and Rullo, P. 2000. Enhancing Disjunctive Datalog by Constraints. *IEEE Transactions on Knowledge and Data Engineering 12,* 5, 845–860.

Burke, R., Hammond, K. and Kozlovsky, J. 1995. Knowledge-Based Information Retrieval from Semi-Structured Text. *Working Notes of the AAAI '95 Fall Symposium, Series on AI Applications in Knowledge Navigation and Retrieval, Cambridge, MA*. 19–24.

Chen, Y.-J. and Soo, V.-W. 2001. Ontology-Based Information Gathering Agents. *Proceedings of the First Asia-Pacific Conference on Web Intelligence* (*WI 2001*), N. Zhong *et al.*, Ed. LNCS, subseries LNAI, vol. 2198. Springer, 423–427.

Collet, C., Huhns, M. and Shen, W.-M. 1991. Resource Integration using a Large Knowledge Base in Carnot. *IEEE Computer 24,* 12, 55–62.

Decker, K., Sycara, K. and Williamson, M. 1997. Middle-Agents for the Internet. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence* (*IJCAI '97*). Vol. 1. Morgan Kaufmann, 578–583.

Delgrande, J. and Schaub, T. 1994. A General Approach to Specificity in Default Reasoning. *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning* (*KR '94*). 146–157.

Delgrande, J., Schaub, T. and Tompits, H. 2001. plp: A Generic Compiler for Ordered Logic Programs. *Proceedings of the Sixth International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2001*), T. Eiter, W. Faber, and M. Truszczyński, Eds. LNCS, subseries LNAI, vol. 2173. Springer, 411–415.

Delgrande, J. P., Schaub, T. and Tompits, H. 2003. A Framework for Compiling Preferences in Logic Programs. *Theory and Practice of Logic Programming 3,* 2, 129–187.

Deutsch, A., Fernandez, M., Florescu, D., Levy, A. and Suciu, D. 1999. A Query Language for XML. *Computer Networks 31,* 11–16, 1155–1169.

Dimopoulos, Y. and Kakas, A. 2001. Information Integration and Computational Logic. *Computational Logic, Special Issue on the Future Technological Roadmap of Compulog-Net*, 105–135.

Eiter, T., Fink, M., Sabbatini, G. and Tompits, H. 2002a. On Properties of Update Sequences Based on Causal Rejection. *Theory and Practice of Logic Programming 2,* 6, 721–777.

Eiter, T., Fink, M., Sabbatini, G. and Tompits, H. 2002b. Using Methods of Declarative Logic Programming for Intelligent Information Agents. *Theory and Practice of Logic Programming 2,* 6, 645–719.

Eiter, T., Fink, M. and Tompits, H. 2003. A Knowledge-Based Approach for Selecting Information Sources. Tech. Rep. INFSYS RR-1843-03-14, 2003, Institut für Informationssysteme, Technische Universität Wien.

Eiter, T., Gottlob, G. and Mannila, H. 1997. Disjunctive Datalog. *ACM Transactions on Database Systems 22,* 3, 364–418.

Eiter, T., Ianni, G., Schindlauer, R. and Tompits, H. 2005a. A Uniform Integration of Higher-Order Reasoning and External Evaluations in Answer-Set Programming. *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence* (*IJCAI 2005*). Morgan Kaufmann.

Eiter, T., Ianni, G., Schindlauer, R. and Tompits, H. 2005b. Nonmonotonic Description Logic Programs: Implementation and Experiments. *Proceedings of the Twelfth International Conference on Logic for Programming, Artificial Intelligence and Reasoning* (*LPAR 2004*), F. Baader and A. Voronkov, Eds. LNCS, vol. 3452. Springer, 511–517.

Eiter, T., Lukasiewicz, T., Schindlauer, R. and Tompits, H. 2004. Combining Answer-Set Programming with Description Logics for the Semantic Web. *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning* (*KR 2004*), D. Dubois, C. Welty, and M.-A. Williams, Eds. Morgan Kaufmann, 141–151.

Faber, W., Leone, N. and Pfeifer, G. 2004. Recursive Aggregates in Disjunctive Logic Programs: Semantics and Complexity. *Proceedings of the Ninth European Conference on Logics in Artificial Intelligence* (*JELIA 2004*), J. J. Alferes and J. A. Leite, Eds. LNCS, subseries LNAI, vol. 3229. Springer, 200–212.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Fink, M. 2002. Declarative Logic-Programming Components for Information Agents. Ph.D. thesis, Institut für Informationssysteme, Technische Universität Wien, Austria.

Fowler, J., Perry, B., Nodine, M. H. and Bargmeyer, B. 1999. Agent-Based Semantic Interoperability in InfoSleuth. *SIGMOD Record 28,* 1, 60–67.

Fuhr, N. 1999. A Decision-Theoretic Approach to Database Selection in Networked IR. *ACM Transactions on Information Systems 17,* 3, 229–249.

Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J., Vassalos, V. and Widom, J. 1997. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems 8,* 2, 117–132.

Geerts, P. and Vermeir, D. 1993. A Nonmonotonic Reasoning Formalism using Implicit Specificity Information. *Proceedings of the Second International Workshop on Logic Programming and Nonmonotonic Reasoning* (*LPNMR '93*), L.-M. Pereira and A. Nerode, Eds. LNCS, subseries LNAI. Springer, 380–396.

Geerts, P. and Vermeir, D. 1995. Specificity by Default. *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty* (*ECSQARU '95*). LNCS, subseries LNAI, vol. 946. Springer, 207–216.

Gelfond, M. and Lifschitz, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing 9,* 3–4, 365–386.

Genesereth, M., Keller, A. and Duschka, O. 1997. Infomaster: An Information Integration System. *Proceedings of the ACM SIGMOD International Conference on Management of Data* (*SIGMOD '97*), J. Peckham, Ed. ACM Press, 539–542.

Goto, S., Ozono, T. and Shintani, T. 2001. A Method for Information Source Selection using Thesaurus for Distributed Information Retrieval. *Proceedings of the Pacific Asian Conference on Intelligent Systems 2001* (*PAIS 2001*). 272–277.

Grosof, B. N., Horrocks, I., Volz, R. and Decker, S. 2003. Description Logic Programs: Combining Logic Programs with Description Logics. *Proceedings of the Twelfth International World Wide Web Conference* (*WWW 2003*). ACM Press, 48–57.

Huffman, S. B. and Steier, D. 1995. A Navigation Assistant for Data Source Selection and Integration. *Working Notes of the AAAI '95 Fall Symposium Series on AI Applications in Knowledge Navigation and Retrieval, Cambridge, MA*. AAAI Press, 72–77.

Huhns, M. and Singh, M. 1992. The Semantic Integration of Information Models. *Proceedings of the AAAI Workshop on Cooperation among Heterogeneous Intelligent Agents.*

Inoue, K. and Sakama, C. 2000. Prioritized Logic Programming and Its Applications to Commonsense Reasoning. *Artificial Intelligence 123,* 1–2, 185–222.

Kirk, T., Levy, A., Sagiv, Y., and Srivastava, D. 1995. The Information Manifold. *Proceedings of the AAAI 2001 Spring Symposium on Information Gathering in Distributed Heterogeneous Environments.* AAAI Press, 85–91.

Kowalski, R. A. and Sadri, F. 1990. Logic Programs with Exceptions. *Proceedings of the Seventh International Conference on Logic Programming* (*ICLP '90*). MIT Press, 598–616.

Krentel, M. 1988. The Complexity of Optimization Problems. *Journal of Computer and System Sciences 36*, 490–509.

Laenens, E. and Vermeir, D. 1990. A Logical Basis for Object-Oriented Programming. *Proceedings of the Second European Workshop on Logics in Artificial Intelligence* (*JELIA '90*). LNCS, subseries LNAI. Springer, 317–332.

Lenat, D. B. and Guha, R. V. 1990. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S. and Scarcello, F. 2006. The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic*. To appear.

Levy, A., Rajaraman, A. and Ordille, J. 1996. Querying Heterogeneous Information Sources using Source Descriptions. *Proceedings of the Twentysecond International Conference on Very Large Data Bases* (*VLDB '96*), T. Vijayaraman, A. Buchmann, C. Mohan, and N. Sarda, Eds. Morgan Kaufmann, 251–262.

Levy, A., Srivastava, D. and Kirk, T. 1995. Data Model and Query Evaluation in Global Information Systems. *Journal of Intelligent Information Systems 5,* 2, 121–143.

Levy, A. and Weld, D. 2000. Intelligent Internet Systems. *Artificial Intelligence 118,* 1–2, 1–14.

Lifschitz, V. and Turner, H. 1994. Splitting a Logic Program. *Proceedings of the Eleventh International Conference on Logic Programming* (*ICLP '94*). MIT Press, 23–38.

Luke, S., Spector, L., Rager, D. and Hendler, J. 1997. Ontology-Based Web Agents. *Proceedings of the First International Conference on Autonomous Agents* (*Agents '97*), W. L. Johnson, Ed. 59–66.

MacGregor, R. and Bates, R. 1987. The LOOM Knowledge Representation Language. Tech. Rep. RS-87-188, Information Sciences Institute, University of Southern California. Project Web page `http://www.isi.edu/isd/LOOM/`.

Minker, J., Ed. 1988. *Foundations of Deductive Databases and Logic Programming.* Morgan Kaufman, Washington DC.

Motik, B., Volz, R. and Maedche, A. 2003. Optimizing Query Answering in Description Logics using Disjunctive Deductive Databases. *Proceedings of the Tenth International Workshop on Knowledge Representation meets Databases* (*KRDB 2003*), F. Bry, C. Lutz, U. Sattler, and M. Schoop, Eds. CEUR Workshop Proceedings, vol. 79. RWTH Aachen University, 39–50. `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-79/`.

Nodine, M., Ngu, A., Cassandra, A. and Bohrer, W. 2003. Scalable Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth. *IEEE Transactions on Knowledge and Data Engineering 15,* 5, 1082–1098.

Przymusinski, T. C. 1988. On the Declarative Semantics of Deductive Databases and Logic Programs. See Minker (1988), 193–216.

Sadri, F. and Toni, F. 2000. Computational Logic and Multi-Agent Systems: A Roadmap. *Computational Logic, Special Issue on the Future Technological Roadmap of Compulog-Net*, 1–31.

Schindlauer, R. 2002. Representation of SQL Queries for Declarative Query Analysis. M.S. thesis, Institut für Informationssysteme, Technische Universität Wien, Austria.

Sim, K. M. and Wong, P. T. 2001. Web-Based Information Retrieval using Agent and Ontology. In *Proceedings of the First Asia-Pacific Conference on Web Intelligence* (*WI 2001*), N. Zhong *et al*., Ed. LNCS, subseries LNAI, vol. 2198. Springer, 384–388.

Singh, M., Cannata, P., Huhns, M., Jacobs, N., Ksiezyk, T., Ong, K., Sheth, A., Tomlinson, C. and Woelk, D. 1997. The Carnot Heterogeneous Database Project: Implemented Applications. *Distributed and Parallel Databases 5,* 2, 207–225.

Subrahmanian, V., Bonatti, P., Dix, J., Eiter, T., Kraus, S., Ozcan, F. and Ross, R. 2000. *Heterogeneous Agent Systems: Theory and Implementation.* MIT Press.

Swift, T. 2004. Deduction in Ontologies via ASP. *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2004*), I. Niemelä and V. Lifschitz, Eds. LNCS, subseries LNAI, vol. 2923. Springer, 275–288.

Van Nieuwenborgh, D. and Vermeir, D. 2002. Preferred Answer Sets of Ordered Logic Programs. *Proceedings of the Eighth European Conference on Logics in Artificial Intelligence* (*JELIA 2002*), S. Flesca, S. Greco, G. Ianni, and N. Leone, Eds. LNCS, subseries LNAI, vol. 2424. 432–443.

Wendlandt, E. B. and Driscoll, J. R. 1991. Incorporating a Semantic Analysis into a Document Retrieval Strategy. *Proceedings of the Fourteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, A. Bookstein, Y. Chiaramella, G. Salton, and V. V. Raghavan, Eds. ACM Press, 270–279.

Wiederhold, G. 1993. Intelligent Integration of Information. *Proceedings of the ACM SIGMOD Conference on Management of Data* (*SIGMOD '93*). 434–437.