In: Proc. Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR 2002), Toulouse, France. Morgan Kaufman, 2002.

A Generic Approach for Knowledge-Based Information-Site Selection

Thomas Eiter, Michael Fink, Giuliana Sabbatini, and Hans Tompits

Institut für Informationssysteme, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria
e-mail: {eiter,michael,giuliana,tompits}@kr.tuwien.ac.at

Abstract

With the advent of the World Wide Web, a vast number of heterogenous information sources has become available. In order to access and process these data, suitable tools and methods for building an information infrastructure are necessary. One task for this purpose is the selection of relevant information sources in automated query answering. In this paper, we present an approach for information-site selection based on the answer set programming paradigm. We use extended logic programs to represent rich descriptions of the information sources, an underlying domain theory, and queries in a formal language. Moreover, we use extended logic programs also for declarative query analysis and query abstraction. The central part of our architecture are *site-selection programs*, representing both qualitative and quantitative criteria. Because of the structured nature of data items, the semantics of such programs, given in terms of prioritized logic programs, must carefully respect implicit context information in site-selection rules, and furthermore combine it with possible user preferences. An experimental prototype of the overall approach has been implemented using the `dlv` KR system and its `plp` front-end for prioritized logic programs.

1 Introduction

With the advent of the World Wide Web, a wealth of information has become available to a large group of users. However, the distribution of the information sources, their heterogeneity and diverse ways of accessibility have raised the need for suitable tools and methods for building an information infrastructure. A promising approach is based on multi-agent systems, in which special *information agents* (or *middle agents* [10]) provide various services, including finding, selecting, and querying relevant information sources.

The potential of knowledge-based approaches—especially of logic programming—for developing reasoning components of intelligent information agents is outlined in [14, 16, 29]. In this paper, we pursue this further and present a declarative approach to information source selection. Here, the problem is, given a query of the user, which out of a collection of information sources should the agent select for answering the query, such that the utility of the answer, in terms of quality of the result and other criteria (e.g., costs), is as large as possible for the user? Obviously, a sensible solution to this problem is nontrivial and involves various aspects such as elementary properties of the information sources, knowledge about their contents, and knowledge about the application domain.

Our approach incorporates such aspects, and makes several contributions, which are briefly highlighted as follows.

(1) We use extended logic programs [21] to represent rich descriptions of the information sources, an underlying domain theory, and queries in a formal language. We perform query analysis by logic programs and compute *query abstractions*. Here, we consider XML-QL [13], but other query languages could be handled as well.

(2) At the heart, a declarative *site selection program* represents *both qualitative and quantitative* criteria (such as site preference and costs). Its rules may access information provided by the other programs, including occurrences of objects and values in the query. For instance, a rule r_1 may state that a query about actors's awards (where actors

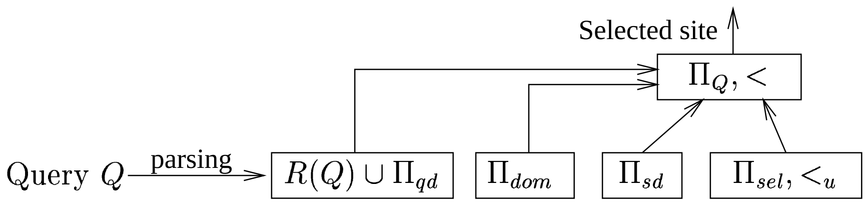

Figure 1: General Architecture of a Selection Base $\mathcal{S} = \langle \Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u \rangle$ for Site Selection.

and awards are objects) should be posed to information source s_1.

(3) We consider the interesting and, to our knowledge, novel issue of *contexts* in logic programs. Structured data items require a careful definition of the selection program semantics. If, besides r_1 as above, rule r_2 says that source s_2 is in general best for awards, then r_1 should have priority over r_2 and s_1 should be queried. Note that this priority is not based on inheritance (cf. [7]), but on the *context* of the occurrence of awards, which is determined by the access path in the query. Furthermore, implicit priorities, derived from context information, must be combined with explicit user preferences from the selection policy, and arising conflicts must be resolved.

Generally speaking, the main advantage of using extended logic programs under the answer set semantics for dealing with the site-selection task considered here is given by their declarative nature, which provides a clear semantics for knowledge representation and reasoning, both under qualitative as well as quantitative criteria. Furthermore, this formalism is capable of handling incomplete information, modeling some form of nonmonotonic reasoning which, arguably, is an inherent feature of the current problem domain. As well, changes in the specification of the site-selection process are easily incorporated by modifying or adding suitable rules or constraints, without the need for re-designing the given program, as might 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 for providing more advanced features for the domain knowledge.

2 Outline of the Approach

Our approach is based on the *answer set programming* paradigm [28], in which problems are represented by extended logic programs (ELPs) [21], possibly augmented with priorities (e.g., [6, 11, 25]) and weak constraints [8], and solutions are obtained from the answer sets of the programs. We assume the reader familiar with answer sets; for further background, see e.g., [21, 28, 4].

Let us illustrate our conception with the following scenario.

Example 1 *Consider an information agent which knows information sources s_1, s_2, and s_3 about movies, containing XML data. Assume that the following query (in XML-QL [13]) is handed to the agent, which informally asks a source for 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>}
```

Suppose the agent knows that s_1 is especially accurate for information about directors, while s_2 has usually good coverage of information about person data; all which known about s_3 is that the site is not particularly reliable. In this situation, we would expect the agent to select s_1 for querying.

While the above example is simple, it shows that the selection process uses different kinds of knowledge. In our approach, this is reflected by the following components of a *selection base* $\mathcal{S} = \langle \Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u \rangle$ (see Figure 1):

Query description Π_{qd}: For any query Q as in Example 1, a high-level abstract description is extracted from a low-level (syntactic) representation $R(Q)$, given as a set of elementary facts, by applying a logic program Π_{qd} to $R(Q)$. Informally, the rules of Π_{qd} single out the essential parts of Q, such as occurrence of attributes and values in the query, comparison and joins, or access paths describing contexts.

Domain theory Π_{dom}: The agent's background knowledge about the application domain (e.g., movies) is represented in a logic program Π_{dom}. It includes a domain ontology, which is established by the knowledge engineer; under certain circumstances, semi-automatic support may be provided from meta-information about the data stored in information sites (e.g., an XML DTD).

Site description Π_{sd}: Information about the sites is represented in a logic program Π_{sd}, using special predicates for (i) *thematic aspects* concerning query topics, e.g., $\mathit{accurate}(S,T,V)$, stating that site S is accurate on topic T at value $V \in \{\mathit{high}, \mathit{medium}, \mathit{low}\}$; (ii) *cost aspects*, e.g., $\mathit{charge}(S,C)$, expressing that querying site S has cost C; and (iii) *technical aspects*, e.g., $\mathit{last\_update}(S,D)$, which gives the date D of the last update of site S. Besides facts, Π_{sd} may contain also rules, and in particular default rules for incomplete information.

Site-selection program Π_{sel}: The selection of information sites is specified using rules and constraints, which refer to predicates defined in the above programs. It comprises both *qualitative aspects*, given by rules and constraints, as well as *quantitative aspects* in terms of optimization criteria (concerning, e.g., cost or response time), which are represented using weak constraints [8].

The user may also define preferences between rules, in terms of a strict partial order, $<_u$. These preferences must be combined with implicit priorities derived from the *context* in which rules for site selection should be applied. We define a respective semantics, which resolves also possible conflicts that might arise.

Summarizing, given a query Q, the overall evaluation proceeds in three steps:

Step 1 (query representation): The input query Q is parsed and mapped into the internal query representation $R(Q)$.

Step 2 (qualitative selection): From $R(Q)$, 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 answer sets of $(\Pi_Q, <)$.

Step 3 (optimization): Among the candidates of Step 2, the one is chosen which represents the best solution with respect to the quantitative aspects of Π_{sel}, and the selected site is provided as output.

In this paper, we focus on Π_{sel} and the construction of $(\Pi_Q, <)$ (see Section 4). The next section provides some details concerning the query-description program Π_{qd}; a comprehensive discussion of the employed query abstraction method is reported in a companion paper [17].

3 Abstract Query Description

An important aspect of our approach is a meaningful description of a given query Q. For our purposes, we need a suitable representation of the constituents of Q in terms of predicates and objects. Simply mapping Q to logical facts reflecting its syntactic structure, which can be easily derived from the query grammar, will not serve our purposes. Rather, we need an *abstract* description which provides "relevant" information about Q, such as occurrence of an attribute or a value in Q, related to the scope in which it occurs. As an example, the value "Hitchcock" occurs in the query of Example 1 in a selection condition, which is performed on the attribute *LastName* reached by the access path $\mathit{Movie}/\mathit{Director}/\mathit{Personalia}$.

In our abstract query description, we adopt a general view in which a query consists of a *construct part*, a *where part*, and a *source part*, corresponding to the SQL *select*, *where*, and *from parts*, respectively. At a high-level description, the following predicates are used for describing a query Q, using surrogates for maximal access paths (starting from a root object) in Q from the low-level description:

$\mathit{access}(O,C,P,Q)$: This predicate describes that an item, which may be an attribute or a concept, is accessed under path P within the context of a concept C in the where-part of query Q, where O identifies a maximal path in Q with suffix "C/P". We call such a pair (C,P) a *context-reference pair in* Q.

query(Q): identifies an individual, independent query Q, taking into account that a query might be composed of several such subqueries.

occurs(O, V): this means that a value V is associated with a maximal path O in the (global) query. For example, *occurs*$(o_2,$ "*Alfred*"$)$ expresses that value *Alfred* is associated with the maximal path o_2.

selects(O, C, V): like *occurs*, but also specifies the type of relationship in terms of comparison operator C. For example, we have *selects*$(o_2,$ *equal*, "*Alfred*"$)$, where *equal* represents the equality operator.

constructs(O, I, P): specifies the maximal paths O which, by use of variables, also appear in the construct part of the query, as an item I under path P (which may be different from the path in the where part). In our example, we have *constructs*$(o_1,$ "*Movie*", " "$)$.

joins(O_1, O_2, C): records joins of (or within) queries between maximal paths O_1 and O_2 under comparison operator C. Our example has no join.

Example 2 *The complete high-level description of the query in Example 1 is given by the following set of facts:*

{ *query*(q_1),
access$(o_1,$ "*MovieDB*", "*Movie/Title*", $q_1)$,
access$(o_1,$ "*Movie*", "*Title*", $q_1)$,
access$(o_2,$ "*MovieDB*", "*Movie/Director/Personalia/FirstName*", $q_1)$,
access$(o_2,$ "*Movie*", "*Director/Personalia/FirstName*", $q_1)$,
access$(o_2,$ "*Director*", "*Personalia/FirstName*", $q_1)$,
access$(o_2,$ "*Person*", "*FirstName*", $q_1)$,
access$(o_3,$ "*MovieDB*", "*Movie/Director/Personalia/LastName*", $q_1)$,
access$(o_3,$ "*Movie*", "*Director/Personalia/LastName*", $q_1)$,
access$(o_3,$ "*Director*", "*Personalia/LastName*", $q_1)$,
access$(o_3,$ "*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*", " "$)$ }

They are obtained from the maximal paths MovieDB/Movie/Title and MovieDB/Movie/Director/Personalia/FirstName corresponding to o_1 *and* o_2*, respectively. Here, "MovieDB", "Movie", "Director", and "Person" are concepts, given in the ontology; furthermore, "Personalia" is known to amount to "Person".*

Note that the high-level description of Q does not depend on a particular query language. It is extracted from a low-level query description, $R(Q)$, which contains further structural details and can be generated by a query parser, by means of a stratified logic program Π_{qd} which imports the relevant ontology knowledge. The high-level description facts are given by the (unique) answer set of $\Pi_{qd} \cup Q(R)$. Further details are provided in [17].

4 Site Selection

In this section, we describe the formal details of site-selection programs, which represent the main part of our architecture. Roughly speaking, a site-selection program Π_{sel} is a prioritized logic program consisting of four parts, namely (i) a core unit Π^c_{sel}, containing the actual site-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. We define syntax and semantics of site-selection programs, and discuss some basic properties.

4.1 Syntax

The alphabet, $\mathcal{A}_{sel}$, of a site-selection program Π_{sel} in a selection base $\mathcal{S} = \langle \Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u \rangle$ consists of the following pairwise disjoint categories:

- the alphabets $\mathcal{A}_{qd}$, $\mathcal{A}_{sd}$, and $\mathcal{A}_{dom}$ of the query description Π_{qd}, the site description Π_{sd}, and the domain theory Π_{dom}, respectively;
- the selection predicate *query_site*(S, Q), expressing that site S is selected for evaluating query Q; and the predicates *default_object*(O, C, Q) and *default_path*(O, P, Q), which are projections of *access*(O, C, P, Q);
- a set $\mathcal{A}_{aux}$ of auxiliary predicates; and
- a set N of terms serving as *names* for rules.

Informally, the predicates *default_object*(O, C, Q) and *default_path*(O, P, Q) serve to specify a default status for selection rules depending on context-reference pairs matched in the query Q. For example, *default_object*$(O,$ "*Person*"$, Q)$ in the body of rule r says that r is eligible, if the concept *Person* occurs in

$$
\begin{array}{llll}
r_1\colon & \mathit{query\_site}(s_2, Q) & \leftarrow & \mathit{default\_object}(O, \text{“}\mathit{Person}\text{”}, Q);\\
r_2\colon & \mathit{query\_site}(s_1, Q) & \leftarrow & \mathit{selects}(O, \mathit{equal}, \text{“}\mathit{Hitchcock}\text{”}),\\
 & & & \mathit{access}(O, \text{“}\mathit{Director}\text{”}, \text{“}\mathit{Personalia/LastName}\text{”}, Q);\\
r_3\colon & \mathit{query\_site}(S, Q) & \leftarrow & \mathit{default\_path}(O, \text{“}\mathit{LastName}\text{”}, Q),\\
 & & & \mathit{default\_object}(O, T, Q),\\
 & & & \mathit{accurate}(S, T, \mathit{high});\\
r_4\colon & \mathit{high\_acc}(T, Q) & \leftarrow & \mathit{access}(O, T, P, Q),\\
 & & & \mathit{accurate}(S, T, \mathit{high});\\
r_5\colon & \mathit{high\_cov}(T, Q) & \leftarrow & \mathit{access}(O, T, P, Q),\\
 & & & \mathit{covers}(S, T, \mathit{high});
\end{array}
$$

$$
\begin{array}{lll}
c_1\colon & \Leftarrow & \mathit{query\_site}(S, Q),\\
 & & \mathit{high\_acc}(T, Q),\\
 & & \mathit{not}\ \mathit{accurate}(S, T, \mathit{high}) \quad [10:1];\\
c_2\colon & \Leftarrow & \mathit{query\_site}(S, Q),\\
 & & \mathit{high\_cov}(T, Q),\\
 & & \mathit{not}\ \mathit{covers}(S, T, \mathit{high}) \quad [5:1];
\end{array}
$$

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

Figure 2: Site-Selection Program $(\Pi_{sel}, <_u)$ from Example 3.

the access path O and there is no other rule s that refers to some context-reference pair (C', P') matched in Q. These defaults are semantically realized using a suitable rule ordering.

We assume that $\mathcal{A}_{sel}$ generates a function-free language; rules are constructed as usual. We use upper-case letters for variables and lower-case letters for constants. We write $r(X_1, \ldots, X_n)$ to indicate that rule r has variables $X_1, \ldots, X_n$. As well, $H(r)$ denotes the head of r and $B(r)$ its body. The set of all literals built from the atoms in $\mathcal{A}_\ell$, where $\ell \in \{qd, sd, dom, aux, sel\}$, is denoted by Lit_ℓ. For 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 some atom A. Furthermore, we use an injective function $n(\cdot)$ which assigns to each rule $r(X_1, \ldots, X_n) \in \Pi_{sel}$ a name $n(r) \in N$ of form $t(X_1, \ldots, X_n)$, such that $n(\cdot)$ naturally extends to Π_{sel} by $n(s) = t(g_1, \ldots, g_n)$ for each ground instance $s = r(g_1, \ldots, g_n)$ of r. To ease notation, we also write n_r instead of $n(r)$.

Definition 1 *A site-selection program over $\mathcal{A}_{sel}$ is a tuple $(\Pi_{sel}, <_u)$, where*

(i) Π_{sel} *is a collection of rules over $\mathcal{A}_{sel}$ consisting of the following parts:*

(a) *the* core unit *Π^c_{sel} containing rules of form*

$$\mathit{query\_site}(S, Q) \leftarrow L_1, \ldots, L_m, \mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n;$$

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

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

(c) *an* optimization part *Π^o_{sel} containing weak constraints* [8] *of form*

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

where L_0 is either a literal from Lit_{aux} or is of form $\neg\,\mathit{query\_site}(\cdot, \cdot)$, and $L_i \in Lit_{sel}$ for $1 \leq i \leq n$, and $w, l \geq 1$ are integers. The number w is the weight *and l is the* priority level *of the weak constraint in (c).*

(ii) *The relation $<_u$ is a strict partial order (i.e., an irreflexive and transitive relation) between names of rules in Π^c_{sel}, whose elements are called* user defined preferences. *If $n_r <_u n_s$ then s is said to have* preference *over r.*

The rules in the core unit Π^c_{sel} serve for selecting a site, based on information from the domain description Π_{dom}, the site description Π_{sd}, the query description $R(Q) \cup \Pi_{qd}$, and possibly from auxiliary rules.

The latter may be used, e.g., for evaluating complex conditions. By $<_u$, preference of site selection can be expressed. The weak constraints in Π^o_{sel} are used to filter answer sets under *quantitative conditions.* Informally, they work as follows: From the answer sets of the weak-constraint free part of a program Π, prune first those where the sum of weights of violated constraints in the highest priority level is not minimal, then those where the sum of weights of violated constraints in the next lower level is not minimal, and so on. We refer the reader to [8] for a precise formal definition.

Example 3 *Consider some site-selection program* $(\Pi_{sel}, <_u)$ *for our movie domain, depicted in Figure 2. Intuitively, rule* r_1 *advises to choose site* s_2 *if the query involves persons and no more specific rule is eligible. Furthermore, rule* r_2 *says to choose site* s_1 *if the query contains an explicit select on the movie director Hitchcock. Rule* r_3 *demands to choose a site if, on some query access path, "LastName" is accessed under some concept* T *(with arbitrary intermediate access path), and the site is highly accurate for* T*. Rules* r_4 *and* r_5 *define auxiliary predicates which hold on concepts* T *appearing in the query* Q *such that some site with high accuracy and coverage, respectively, for* T *exists. The weak constraints* c_1 *and* c_2 *state penalties for choosing a site that has not high accuracy (weight 10) or coverage (weight 5), respectively, for a concept in the query while such a site exists. Finally,* $n_{r_1(Q,\_)} <_u n_{r_3(Q,\_,\_,\_)}$ *expresses that on the same query, instances of rule* r_3 *are preferred to those of* r_1.

4.2 Semantics

The semantics of a site-selection program $(\Pi_{sel}, <_u)$ in a selection base $\mathcal{S} = \langle \Pi_{qd}, \Pi_{dom}, \Pi_{sd}, \Pi_{sel}, <_u \rangle$ on a query Q is given by the concept of an *answer set* of $(\Pi_{sel}, <_u)$, which is defined as a preferred answer set of a prioritized logic program $\mathcal{E}(\mathcal{S}, Q) = (\Pi_Q, <)$ associated with $\mathcal{S}$ and Q. The program Π_Q contains ground instances of rules and constraints in Π_{sel}, and further rules ensuring that a single site is selected per query and 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. In that, we must suitably combine preference information—and in particular handle conflicts. For this, we use a cautious conflict elimination policy.

We commence the formal details with the following notation: For any rule r, its *defaultization*, r^Δ, is given by $H(r) \leftarrow B(r), not\, \neg H(r)$.

Definition 2 *Let* $\mathcal{S} = \langle \Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u \rangle$ *be a selection base and* Q *a query. Then, the program* Π_Q *contains all ground instances of the rules and constraints in* $\Pi^{aux}_{sel} \cup \Pi^o_{sel}$*, as well as all ground instances of the following rules:*

1. *the defaultization* r^Δ *of* r*, for each* $r \in \Pi^c_{sel}$*;*
2. *the* structural rule

   $$\neg query\_site(S, Q) \;\leftarrow\; query\_site(S', Q), S \neq S';$$

   and
3. *the* default context rules

   $$\begin{aligned} default\_object(O, T, Q) &\leftarrow access(O, T, \_, Q); \\ default\_path(O, P, Q) &\leftarrow access(O, \_, P, Q). \end{aligned}$$

Intuitively, the defaultization makes the selection rules in Π_{sel} defeasible; the structural rule enforces that only one source is selected; and the default context rules define the two default predicates. We remark that, since we have no function symbols in our language, Π_Q is finite, and its size depends on the constants appearing in Π_{qd}, $R(Q)$, Π_{sd}, and Π_{dom}.

For $\mathcal{S} = \langle \Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u \rangle$ and query Q, we call any answer set of $\Pi = \Pi_{qd} \cup R(Q) \cup \Pi_{sd} \cup \Pi_{dom}$ a *selection input of* $\mathcal{S}$ *for* Q. The set of all selection inputs of $\mathcal{S}$ for Q is denoted by $Sel(\mathcal{S}, Q)$. For $I \in Sel(\mathcal{S}, Q)$, we define

$$\begin{aligned} I_{def} &= I \,\cup \\ &\cup\; \{default\_object(o, c, q) \mid access(o, c, p, q) \in I\} \\ &\cup\; \{default\_path(o, p, q) \mid access(o, c, p, q) \in I\}. \end{aligned}$$

Notice 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 site description Π_{sd} and the domain knowledge Π_{dom} have unique answer sets. In our framework, this is ensured if, for instance, these components are represented by stratified programs.

Definition 3 *A rule* $r \in \Pi_Q$ *is* relevant *for* Q *iff there is some* $I \in Sel(\mathcal{S}, Q)$ *such that* $B^\dagger(r)$ *is true in* I_{def}*, where* $B^\dagger(r)$ *results from* $B(r)$ *by deleting each element which does not contain a predicate symbol from* $\mathcal{A}_{qd} \cup \mathcal{A}_{sd} \cup \mathcal{A}_{dom} \cup \{default\_object,\ default\_path\}$*.*

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

We continue with the construction of the preference relation $<$, used for interpreting a site-selection program

$(\Pi_{sel}, <_u)$ relative to a selection base $\mathcal{S}$ 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 $\preceq_c$ represents the first step in the construction of $<$, transforming structural context information into explicit preferences, in virtue of the following specificity conditions:

- relative to the same maximal path and query, rules that mention default concepts or a context-reference pair matching the query are considered more specific than rules mentioning default paths;
- default contexts for concepts are assumed to be more specific than default contexts for attributes; and
- rules which have in the same maximal path more detailed context-reference pairs are considered more specific than rules with less such objects.

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 priorities $n_r \trianglelefteq n_s$ which lie on a cycle $n_r \trianglelefteq n_s \trianglelefteq n_{s_1} \trianglelefteq \cdots \trianglelefteq n_{s_n} \trianglelefteq n_r$, resulting in $<'$. Finally, taking the transitive closure of $<'$ yields $<$. The formal definition of relation $<$ is as follows.

Definition 4 *Let $\mathcal{S}$ be a selection base, Q a query, and Π_Q as in Definition 2. We introduce the following binary relations over the names of rules in Π_Q: For $r, s \in \Pi_Q$,*

1. *$n_r \preceq_c n_s$ iff r, s are relevant for Q, $r \neq s$, and one of (P_1)–(P_3) holds:*

   (P_1) *$default\_path(o_1, p_1, q)$ is a member of $B(r)$, and we have either $default\_object(o_2, t_2, q) \in B(s)$ or $access(o_2, t_2, p_2, q) \in B(s)$;*

   (P_2) *$default\_object(o_1, t_1, q)$ belongs to $B(r)$ and $access(o_2, t_2, p_2, q)$ is in $B(s)$;*

   (P_3) *$access(o, t_1, p_1, q)$ is a member of $B(r)$, $access(o, t_2, p_2, q)$ is in $B(s)$, and t_1/p_1 is a subpath of t_2/p_2;*

2. *$n_r \trianglelefteq n_s$ iff $n_r <_u n_s$ and r, s are relevant for Q, or $n_r \preceq_c n_s$;*

3. *$n_r <' n_s$ iff $n_r \trianglelefteq n_s$ but not $n_s \trianglelefteq^* n_r$.*

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

Let us illustrate this definition with an example.

Example 4 *Reconsider $(\Pi_{sel}, <_u)$ from Example 3. Suppose the domain ontology contains the concepts "MovieDB", "Actor", "Movie", "Director", and "Person", and that "Personalia" amounts to "Person". Furthermore, let us assume we have a unique selection input I for the query Q of Example 1, containing the following facts from the site description:*

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

as well as the following elements resulting from the query description and the default context rules:

$$default\_path(o_3, \text{“}LastName\text{”}, q_1);$$
$$default\_object(o_2, \text{“}Person\text{”}, q_1);$$
$$default\_object(o_3, \text{“}Person\text{”}, q_1);$$
$$default\_object(o_3, \text{“}Director\text{”}, q_1);$$
$$selects(o_3, equal, \text{“}Hitchcock\text{”});$$
$$access(o_2, \text{“}Person\text{”}, \text{“}FirstName\text{”}, q_1);$$
$$access(o_2, \text{“}Director\text{”}, \text{“}Personalia/FirstName\text{”}, q_1);$$
$$access(o_3, \text{“}Person\text{”}, \text{“}LastName\text{”}, q_1);$$
$$access(o_3, \text{“}Director\text{”}, \text{“}Personalia/LastName\text{”}, q_1).$$

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 $r_1(q_1, o_2)$, $r_2(q_1, o_3)$, $r_2(q_1, o_3)$, and $r_3(q_1, s_1, o_3, \text{“}D\text{”})$ (for brevity, we write here "D" for "Director"). 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 following pairs:

$$n_{r_1(q_1, o_2)} < n_{r_2(q_1, o_3)};$$

$$n_{r_1(q_1,o_3)} < n_{r_2(q_1,o_3)};$$
$$n_{r_3(q_1,s_1,o_3,\text{“}D\text{”})} < n_{r_2(q_1,o_3)}.$$

Note that for each of r_4 and r_5, there also exist two relevant instances. However, they have no influence on the above rule ordering. Informally, they are either unrelated to or "ranked between" the priority of $r_2(q_1, o_3)$ and the priority of the relevant instances of r_1 and r_3 (since the 'access' 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.

With respect to r_1 and r_3, the auxiliary relation $\trianglelefteq$ contains two further structural priorities, namely

$$n_{r_3(q_1,s_1,o_3,\text{“}D\text{”})} \preceq_c n_{r_1(q_1,o_2)}$$

and

$$n_{r_3(q_1,s_1,o_3,\text{“}D\text{”})} \preceq_c n_{r_1(q_1,o_3)}.$$

They are in conflict with the user preferences

$$n_{r_1(q_1,o_2)} <_u n_{r_3(q_1,s_1,o_3,\text{“}D\text{”})}$$

and

$$n_{r_1(q_1,o_3)} <_u n_{r_3(q_1,s_1,o_3,\text{“}D\text{”})},$$

respectively. This conflict is resolved in the resultant relation $<$ by removing these preferences.

Note that the final order $<$ in Definition 4 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; however, this may lead to nondeterministic choices. Namely, in general, multiple such cutsets exist, of which one must be chosen. Different choices lead to different orders $<$, which may lead to different results of the site selection program. Thus, unless a well-defined *particular* minimal cutset is singled out, by virtue of preference conflicts the result of the site 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 4.

Combining Definitions 2 and 4, we obtain the translation $\mathcal{E}(\cdot,\cdot)$ as follows:

Definition 5 *Let $\mathcal{S}$ be a selection base and Q a query. Then, the evaluation $\mathcal{E}(\mathcal{S}, Q)$ of $\mathcal{S}$ with respect to Q is given by the prioritized logic program $(\Pi_Q, <)$, where Π_Q and $<$ are as in Definitions 2 and 4, respectively.*

For defining answer sets of site-selection programs, in what follows we assume the approach of [11] as underlying preference framework; other approaches, like, e.g., [6], are also suitable for this purpose. Furthermore, for any program Π and any set S of literals, we write $\Pi \cup S$ to denote the program $\Pi \cup \{L \leftarrow \mid L \in S\}$.

Definition 6 *Let $\mathcal{S} = \langle \Pi_{qd}, \Pi_{sd}, \Pi_{dom}, \Pi_{sel}, <_u \rangle$ be a selection base, and let $\mathcal{E}(\mathcal{S}, Q) = (\Pi_Q, <)$ for query Q. Then, $S \subseteq Lit_{sel}$ is an* answer set of *$(\Pi_{sel}, <_u)$* for Q with respect to *$\mathcal{S}$ iff S is a preferred answer set of the prioritized logic program $(\Pi_Q \cup I, <)$, for some $I \in Sel(\mathcal{S}, Q)$.*

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

Example 5 *In our running example, $(\Pi_{sel}, <_u)$ has a unique answer set, S, for the query Q from Example 3 with respect to $\mathcal{S}$, containing $query\_site(s_1, q_1)$ (where q_1 represents Q). This atom is derived from the core rule $r_2(q_1, o_3)$, which has 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*

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

and adapt the corresponding user preference to

$$n_{r_1(Q,\_,\_)} <_u n_{r_3(Q,\_,\_,\_)},$$

then the weak-constraint free part of Π_{sel} has two preferred answer sets: one, S_1, is identical to S (where an application of $r_2(q_1, o_3)$ is preferred to an application of $r_1(q_1, o_3)$ given that $n_{r_1(q_1,o_3)} < n_{r_2(q_1,o_3)}$); in the other answer set, S_2, the rule $r_1(q_1, o_2)$ is applied and $query\_site(s_2, q_1)$ is derived. Informally, due to the replacement, the preference of $r_2(q_1, o_3)$ over $r_1(q_1, o_2)$ does no longer hold, since their 'access' predicates refer to different contexts ("…/FirstName" and "…/LastName", respectively). As a consequence, $r_1(q_1, o_2)$ has maximal preference like $r_2(q_1, o_3)$.

Given that S_1 has weight 5, caused by violation of $c_2(s_1, q_1, \text{“}Person\text{”})$, but S_2 has weight 10, caused by violation of $c_1(s_2, q_1, \text{“}Director\text{”})$, S_1 is the answer set of $(\Pi_{sel}, <_u)$ for Q.

4.3 Properties

Finally, we discuss some properties of our framework. Our first result deals with the adequacy of the evaluation method of site-selection programs with respect to the usual semantics of prioritized logic programs.

Recall that the construction of answer sets of site-selection programs is laid out in a modular fashion, depending on the input of several subprograms, where each program module is assigned with a particular representation task. However, since the predicate symbols occurring in the heads of rules in a site-selection program do not occur in rules from the query description, the site description, or the domain theory, from standard modularity results for logic programs [27] we easily obtain the following characterization:

Theorem 1 *Let $\mathcal{S} = \langle \Pi_{qd}, \Pi_{dom}, \Pi_{sd}, \Pi_{sel}, <_u \rangle$ be a selection base, let Q be a query, and let $\mathcal{E}(\mathcal{S}, Q) = (\Pi_Q, <)$. Then, S is an answer set of $(\Pi_{sel}, <_u)$ for Q with respect to $\mathcal{S}$ iff S is a preferred answer set of $(\Pi_Q \cup \Pi_{qd} \cup R(Q) \cup \Pi_{sd} \cup \Pi_{dom}, <)$.*

Furthermore, it is possible to emulate the construction of $\mathcal{E}(\mathcal{S}, Q)$ in terms of a *single* logic program under usage of dynamic priorities. Indeed, given the relevant rules for Q, the order relation $<$ can be computed by a stratified logic program $\Pi_<$ which computes the relations in Definition 4. On the other hand, the relevant rules for Q can be computed in a program Π_{rel} derived from $\Pi_{qd} \cup \Pi_{sd} \cup \Pi_{dom}$ using weak constraints. By combining the components, we obtain:

Theorem 2 *Let $\mathcal{S} = \langle \Pi_{qd}, \Pi_{dom}, \Pi_{sd}, \Pi_{sel}, <_u \rangle$ be a selection base. Then, there exists a logic program $\Pi_{\mathcal{S}} = \Pi_{qd} \cup \Pi_{dom} \cup \Pi_{sd} \cup \Pi_{sel} \cup <_u \cup \Pi_< \cup \Pi_{rel}$, such that for every query Q, the preferred answer sets of $\Pi_{\mathcal{S}} \cup R(Q)$ under dynamic preference $<$ correspond to the answer sets of $(\Pi_{sel}, <_u)$ for Q with respect to $\mathcal{S}$.*

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

Theorem 3 *Let S be an answer set of $(\Pi_{sel}, <_u)$ for query Q with respect to $\mathcal{S}$. Then, for any constant q it holds that*

$$|\{s \mid \mathit{query\_site}(q, s) \in S\}| \leq 1.$$

Intuitively, this result holds due to the presence of the structural rule

$$\neg \mathit{query\_site}(S, Q) \leftarrow \mathit{query\_site}(S', Q), S \neq S'$$

in Definition 2, which enforces that during the evaluation step no more than one site can be selected simultaneously.

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

Theorem 4 *Let S be an answer set of $(\Pi_{sel}, <_u)$ for query Q with respect to $\mathcal{S}$, and let r be some rule from the grounding of Π^c_{sel} for Q with respect to $\mathcal{S}$. Suppose that $B(r)$ is true in S but $H(r) \notin S$. Then, there is some s in the grounding of $\Pi^c_{sel} \cup \Pi^{aux}_{sel}$ for Q with respect to $\mathcal{S}$ such that*

1. *$B(s)$ is true in S;*
2. *$H(s) \in S$; and*
3. *either r and s are incompatible with respect to $<$, or else $r < s$ holds.*

This result follows from the observation that, under the above circumstances, if $B(r)$ is true in S but $H(r) \notin S$, then there must be some s in the grounding of $\Pi^c_{sel} \cup \Pi^{aux}_{sel}$ such that $H(s) = \neg H(r)$ and s fires during the construction of S. Moreover, the underlying preference semantics guarantees that we can choose s in such a way that either r and s are not related with respect to $<$, or else $r < s$ holds.

5 Related Work and Conclusion

Selection of data sources is an ingredient to several systems for information integration, e.g. [2, 5, 9, 20, 22, 24, 26]. However, they center around mappings from a global scheme to local schemes and vice versa, query rewriting, and planning, in order to optimally reconstruct dispersed information. Our concern is with the *qualitative* selection from different alternatives, based on rich meta-knowledge, which is not an issue there. More related is [23], which outlines an interactive tool for assisting information specialists in query design. It relieves them from searching through data source specifications and can provide suggestions for using sources to determine tradeoffs. However, no formal semantics or richer domain theories, capable of handling incomplete and default information, is presented. Remotely related to our work is [19], which presents a decision-theoretic model for selecting data sources based on retrieval cost and typical IR parameters.

Our approach to declarative information site selection, based on query analysis and contexts in logic programs, is novel and innovate in several respects. It is implemented on top of the `dlv` system [18, 15] and its front end `plp` [12] for prioritized logic programs, and employs the underlying Eclipse Prolog engine as a glueing frame. For testing, we developed a prototypical environment of a movie domain, which comprises (i) basic domain knowledge, (ii) XML sources containing movie data wrapped from the Internet Movie Database [1] and other movie related data sources, and

(iii) suitable site descriptions. Queries are formulated in XML-QL [13], and can be executed after site selection on the respective source. For generating the low-level representation $R(Q)$, for query Q, we have developed an XML-QL query parser, written in C++ using the standard tools `lex` and `yacc`. The implementation, as well as the full movie application, is too complex to be described here. In its current version, selection inputs are assumed to be unique, which is ensured if the site description Π_{sd} and the domain knowledge Π_{dom} yield unique answer sets. This is given, for example, if these components are stratified, or by adding suitable weak constraints. The implementation will be considered in more detail in an extended paper.

Our ongoing work comprises several tasks. One is making our site selection method available for building multi-agent information systems. To this end, its *agentization* in the *IMPACT* agent platform [3] is in progress. Another task is coupling the approach with learning and program updates for adaptive source selection, and with query rewriting and planning. Also, the current setting of single site selection can be generalized to multiple site selection for parallel querying.

References

[1] The Internet Movie Database. `http://imdb.com`.

[2] Y. Arens, C. Chee, C. Hsu, and C. Knoblock. Retrieving and Integrating Data from Multiple Information Sources. *Int. J. of Cooperative Information Systems*, 2(2):127–158, 1993.

[3] K. Arisha, T. Eiter, S. Kraus, F. Ozcan, R. Ross, and V. Subrahmanian. IMPACT: A Platform for Collaborating Agents. *IEEE Intelligent Systems*, 14(2):64–72, 1999.

[4] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving with Answer Sets*. 2001. Draft. Available from `http://www.public.asu.edu/~cbaral/bahi/`.

[5] R. Bayardo, B. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. InfoSleuth: Semantic Integration of Information in Open and Dynamic Environments (Experience Paper). In *SIGMOD Conf. 1997*, pages 195–206, 1997.

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

[7] F. Buccafurri, N. Leone, and P. Rullo. Stable Models and their Computation for Logic Programming with Inheritance and True Negation. *J. of Logic Programming*, 27(1):5–43, 1996.

[8] F. Buccafurri, N. Leone, and P. Rullo. Enhancing Disjunctive Datalog by Constraints. *IEEE Trans. on Knowledge and Data Engineering*, 12(5):845–860, 2000.

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

[10] K. Decker, K. Sycara, and M. Williamson. Middle-Agents for the Internet. In *Proc. 15th Int. Joint Conf. on Artificial Intelligence* (*IJCAI'97*), volume 1, pages 578–583. Morgan Kaufmann, 1997.

[11] J. Delgrande, T. Schaub, and H. Tompits. Logic Programs with Compiled Preference. In W. Horn, editor, *Proc. 14th Europ. Conf. on Artificial Intelligence* (*ECAI 2000*), pages 392–398. IOS Press, 2000.

[12] J. Delgrande, T. Schaub, and H. Tompits. plp: A Generic Compiler for Ordered Logic Programs. In T. Eiter, W. Faber, and M. Truszczyński, editors, *Proc. 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'01*), pages 411–415. Springer, 2001.

[13] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A Query Language for XML. In *Proc. 8th International World Wide Web Conference* (*WWW8*), 1999. Computer Networks 31(11-16): 1155-1169. See also `http://www.w3.org/TR/NOTE-xml-ql/`.

[14] Y. Dimopoulos and A. Kakas. Information Integration and Computational Logic. *Computational Logic, Special Issue on the Future Technological Roadmap of Compulog-Net*, 2000. Available from `http://www.compulog.org/net/Forum/Supportdocs.html`.

[15] T. Eiter, W. Faber, N. Leone, and G. Pfeifer. Declarative Problem-Solving Using the DLV System. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 79–103. Kluwer Academic Publishers, 2000.

[16] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. Using Methods of Declarative Logic Programming

for Intelligent Information Agents. *Theory and Practice of Logic Programming*, 2001. To appear. Available from `http://xxx.lanl.gov/abs/cs.MA/0108008`.

[17] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. Declarative Query Abstraction for Selecting Information Sources. Manuscript (in preparation), 2002.

[18] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. The KR System dlv: Progress Report, Comparisons, and Benchmarks. In A. Cohn, L. Schubert, and S. Shapiro, editors, *Proceedings Sixth International Conference on Principles of Knowledge Representation and Reasoning* (*KR-98*), pages 406–417, June 2–4 1998.

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

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

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

[22] M. Genesereth, A. Keller, and O. Duschka. Infomaster: An Information Integration System. In *Proc. ACM SIGMOD Conference*, volume 26(2) of *SIGMOD Record*, pages 539–542, New York, 1997. ACM Press.

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

[24] M. Huhns and M. Singh. The Semantic Integration of Information Models. In *AAAI Workshop on Cooperation among Heterogeneous Intelligent Agents*, 1992.

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

[26] A. Levy, A. Rajaraman, and J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In T. Vijayaraman, A. Buchmann, C. Mohan, and N. Sarda, editors, *Proc. 22th Int. Conf. on Very Large Data Bases* (*VLDB'96*), pages 251–262. Morgan Kaufmann, 1996.

[27] V. Lifschitz and H. Turner. Splitting a Logic Program. In *Proceedings ICLP-94*, pages 23–38, Santa Margherita Ligure, Italy, June 1994. MIT-Press.

[28] A. Provetti and T. C. Son, editors. *Proceedings AAAI 2001 Spring Symposium on Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning, Stanford, CA* (*Workshop Technical Report SS-01-01*). AAAI Press, March 2001.

[29] F. Sadri and F. Toni. Computational Logic and Multi-Agent Systems: a Roadmap. *Computational Logic, Special Issue on the Future Technological Roadmap of Compulog-Net*, 2000. Avaliable from `http://www.compulog.org/net/Forum/Supportdocs.html`.