The Dynamic Information Integration Model[0]

Anisoara Nica and Elke Angelika Rundensteiner
Dept. of Elect. Eng. and Computer Science
University of Michigan
1301 Beal Avenue, Ann Arbor, MI 48109-2122
anica | rundenst@eecs.umich.edu
fax: 313-763-1503

Abstract

Challenging issues for processing queries specified over large-scale information spaces (e.g., Digital Libraries or the World Wide Web) include the diversity of the information sources in terms of their structures, query interfaces and search capabilities, as well as the dynamics of sources continuously being added, removed or upgraded. Query processing can no longer be done in a static fashion against a well-defined schema (which assumes high integration). Rather an interactive query processing strategy that adapts its behavior to the system resources at hand is needed. In this paper, we give an innovative solution for query planning in such environments. The foundation of our solution is the Dynamic Information Integration Model (DIIM) which supports the specification of not only content but also capabilities of resources without requiring the establishment of a uniform integration schema. Besides the development of the DIIM model, contributions of this paper include: (1) introduction of the notion of fully specified queries that are semantically equivalent to a loosely-specified query (we show that our concept is a consistent and natural extension to the concept of full disjunction); (2) translation algorithm of a loosely-specified query into a set of semantically equivalent precise query plans that can be executed against the current configuration of available sources; the steps of the translated query plans are consistent with the binding patterns of query templates of the individual sources (capability descriptions in DIIM) and with possible interrelationships between two or more information sources (expressed as join constraints in DIIM); (3) search restriction algorithm for optimizing query processing by pruning the search space into the *relevant subspace* of a query based on information source descriptions; the search space of the translation algorithm is thus restricted to a query relevant subspace; and (4) the proofs of correctness for both the search restriction and translation algorithms that show that the plans obtained by the query planning process correspond to semantically equivalent query plans.

Keywords - Multi- and Heterogeneous Information Systems, Loosely-Specified Queries, Query Refinement, Query Templates, Planning.

[0]This research has been funded in part by the joint NSF/ARPA/NASA Digital Libraries Initiative under CERA IRI-9411287, and by NSF under grants RIA #IRI-9309076 and NYI #IRI-9457609.

1 Introduction

In recent years, more and more information has become available in on-line digital formats such as Digital Libraries and the World Wide Web which, while offering unique opportunities to everyone of making use of this wealth of information, also introduces many new challenges to the database community. These include: (1) a large number of heterogeneous data sources and other information resources available in the system, (2) a continuously changing configuration of sources joining or leaving the system or changing their interfaces and content, and (3) a wide diversity in data structures as well as query interfaces ranging from fixed APIs to declarative languages.

Typical users of these new systems are not database experts, hence they are not familiar with nor willing to utilize the rigid interfaces and query languages offered by today's DBMS systems. One key challenge that thus needs to be addressed is to allow such users to query the information space without having to:

1. be aware of which information sources the data is located in;
2. know the exact query language used by each source (otherwise one would have to learn multiple such languages);
3. utilize structured query languages (such as SQL) requiring knowledge about the particular schema of the source accessed;
4. deal with differences between structured and unstructured information sources;
5. know how to rewrite a query into subqueries, each targeting only one information source; and
6. know how to combine data from more than one source into one composite response.

While current DBMS systems have some built-in support for processing queries across distributed (typically SQL) databases and combining query results, they do not properly address the above challenge by effectively forcing users to specify queries in their particular query language (i.e., SQL). This clearly is not satisfactory for many users.

Information-retrieval (IR) based systems, such as WAIS and WWW engines [Cor95b, Cor95a], typically allow specification of vague queries, that is, users can list search terms in subjects they are interested in without having to formulate a precise query. Unfortunately, these systems typically work well only with unstructured data, i.e., textual documents, and perform textual (possibly sophisticated) matching of the search words within each text object. They do not split queries across diverse sources nor do they combine information from two separate sources to form composite results. WWW search engines, for example, simply return all individual items that seem to match one or more of the desired search terms.

The goal of our work is to provide a solution to the above problems that offers both the advantages of IR based systems (to allow the user to specify high-level queries) and of DBMS-based systems (to automatically process such queries against a set of distributed possibly structured sources). This requires us to solve the following issues:

1. develop a language allowing queries ranging from vaguely specified queries (i.e., simply a list of terms) to more precisely specified ones (i.e., specifying the particular source from which the data is to be taken, or the join to be performed to combine different sets of data);

2. develop query semantics that specify the notion of the "correct" meaning and hence "correct" answers for such type of vague queries (clearly, performing pairwise natural joins between all sources ala universal relation semantics [Ull89] is neither too meaningful nor too practical in the context of modern information spaces with hundreds of information sources);

3. develop algorithms that map a loosely-specified query into precise well-specified query plans based on defined query refinement semantics (so that the later can be executed against the set of available sources in the system);

4. develop optimization techniques to prune the set of potentially correct equivalent query plans in order to allow the approach to scale up.

The four problems listed above are the focus of this paper, and we make contributions to all of them in this work.

Our overall approach at tackling these problems, and making them sufficiently well-defined and manageable, is based on our proposal of the Dynamic Information Integration Model (DIIM). We assume that all information sources in our space, may they be fully structured like an SQL database management system, partially structured like SGML files [Gol91], or even an application program like **finger**, register their existence as well as a description of their content and capabilities with a knowledge base [ABD+96]. For this purpose, we have developed the Dynamic Information Integration Model which allows us to capture information source descriptions such as relationships among the information sources (specified as join constraints, integrity constraints and partial and complete information descriptions), and information sources capabilities i.e., the types of queries they can answer to (modeled as query templates with binding patterns).

The DIIM query language we propose permits high-level query requests (e.g., that don't specify from where data is to be retrieved or how information from different sources is to be combined) that the system refines into fully specified queries which can be satisfied by the available information sources. A loosely-specified user query such as "Find addresses of on-line journal collections having some articles in the area of *Meteorology* or related subjects" could be very easily expressed in the DIIM query language without having

to specify from where the data is to be retrieved or how pieces of information are to be combined to answer this query.

Our proposed query semantics define a natural way of refining a vague query by imposing how the query is to be rewritten into more restrictive queries that are consistent with both DIIM source descriptions and the original query definition. A refined query uses join consistencies among sources and at the same time is in agreement with their query interfaces. We introduce the notion of *connected relation* as a natural extension of the concept of full disjunction [GL94]. In the default case when only natural joins are defined in DIIM we show that, the semantics of the two concepts (connected relation and full disjunction) are equivalent.

We develop two algorithms for query processing in the context of DIIM semantics: (1) the *search restriction* algorithm and (2) the *query refinement* planner.

First, because our target information spaces contain a large number of sources some of which are likely to be unrelated, we propose a *search restriction* algorithm that will identify what the subspace of the overall information space is that a query is addressed to. We define the notion of *relevant subspace* of a query and give an algorithm for computing it. The purpose of the algorithm is twofold. On the one hand, it optimizes query processing by computing the relevant subspace. On the other hand, it also recognizes if a query has no answer in the current system configuration in which case the relevant subspace is empty.

Second, we present the query planning algorithm. The planning algorithm exploits our proposed semantics for query refinement to refine a loosely-specified user query into well-defined fully-specified queries that can be evaluated in the current system configuration. The algorithm based on regression planning algorithm [RN95] incrementally finds possible plans (corresponding to refined queries) for the given query.

We give proofs of correctness for both the search restriction and translation algorithms that show that the plans obtained by the query planning process are consistent with our definition of fully specified queries that are semantically equivalent to a given user query.

The rest of the paper is organized as follows. Section 2 gives an overview of the DIIM system, while Section 3 presents an example from the digital library world which we will use throughout this paper. Section 4 defines the overall framework for our approach, namely, the DIIM model. The DIIM query language and semantics are introduced in Section 5 as well as a comparison of our query semantics to the full disjunction [GL94] for a set of relations. Sections 6 and 7 describe our proposed algorithms for computing semantically-equivalent query plans for a user query. Namely, Section 6 gives an algorithm for computing the relevant subspace for a query while Section 7 describes the algorithms used for computing the set of semantically-equivalent query plans. Proofs of the correctness are given both in Sections 6 and 7. Section 8 presents related work and Section 9 concludes the paper.

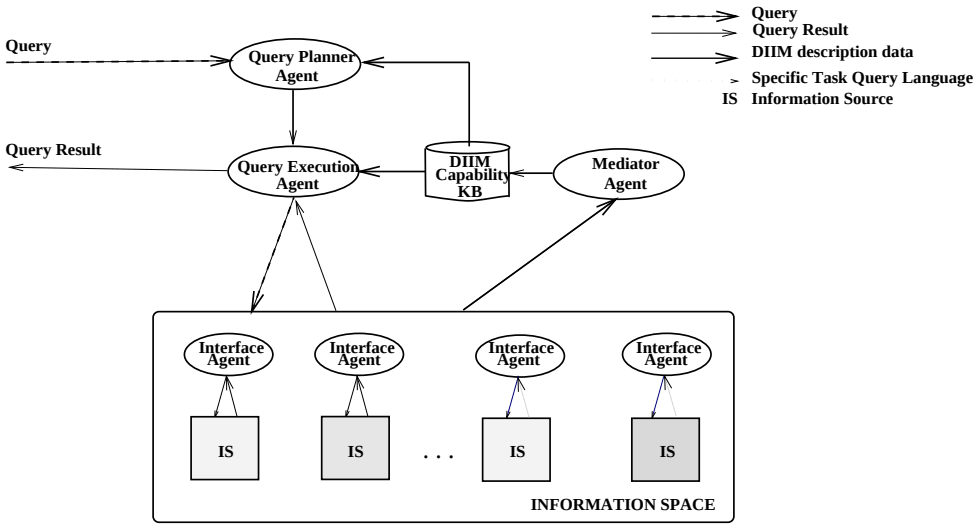

Figure 1: DIIM System Architecture

2 Overview of our Approach

Architectural overview of the DIIM system is given in Figure 1. There are three stages towards query processing: (1) external information sources join the system by registering their capability and content description relating them to other known sources if any, (2) a Mediator Agent keeps the knowledge base of these site descriptions and infers new facts as needed in the process of query planning and execution, and (3) a high-level user query posed to the system will initiate a query planning process that refines it into a set of "good" query plans given the current system configuration. The last stage is followed by the execution of one (or some) of the selected plans by either some system established ranking, or by asking the user for feedback based on refined queries.

The foundation of our information integration paradigm is DIIM (the Dynamic Information Integration Model). DIIM is used to construct information source descriptions using for example join constraints, integrity constraints and partial and complete information descriptions which are maintained in the knowledgebase by the mediator agent. The information sources also describe their capability, e.g., the types of queries they can answer to, which are modeled as query templates with binding patterns in DIIM.

As shown in Figure 1, the agents in our DIIM System Architecture are:

- **Mediator Agent** has the role of collecting external information source descriptions expressed in DIIM. The DIIM descriptions once processed are stored in the *capability knowledge base* to be available to other agents in the system for the purposes of query planning, evaluation and execution.

- **Query Planner Agent** uses the *capability knowledgebase* to generate the library of parameterized action schemas used for query planning. Given a user query, it applies a conditional partial-order

planning algorithm to obtain a set of partial-order query plans.

- **Query Execution Agent** uses the set of plans generated by the **Query Planner Agent** and the *capability knowledge base* to construct the query execution plans. As we will see, the descriptions of the external information sources could be used to prune some of the plans to avoid the retrieval of redundant data. In a real information space a user query has specified some global parameters concerning the result of the query. For example, a user could specify the number of desired query results or the maximum cost he is willing to pay. These kinds of parameters will be taken into account in this stage of query processing for incrementally computing the query results until the query parameters are satisfied.

3 Digital Library Example Scenario

Example 1 *We will use the following example in the rest of the paper, a small part of the reference space of the University of Michigan Digital Library (UMDL) system ([ABD+96]). The reference space contains a thesaurus database, a Broad System of Ordering (BSO) database, a general name authority index, an IEEE index of all the authors that published in an IEEE journal, and a metadata database containing information of various on-line journal collections maintaining information about the URL addresses, journal names and their subjects. The databases are stored at different external sites and have their own search engine and query interface that permit integration into the system via Interface Agents. Part of the description of this space is given throughout this paper.*

Given a user query "Give me at most ten addresses of on-line journal collections having some articles in the area of Meteorology", the system will have to use the information sources described above to formulate the answer. If the first step of going directly to the metadata database fails (i.e., no collection is classified under the Meteorology subject) then it should consider other alternatives for searching, like going to the thesaurus or BSO database for reformulating the query using a synonym or a broader term instead of Meteorology. If there are collections classified with Hydrography then a good plan in obtaining an answer to the query is: (1) take from the thesaurus database all the related terms for Meteorology; the list will contain Hydrography; (2) go back and use this list to extract all the collection addresses that have a subject in the related term list. Note that if the thesaurus database is not available or using the above plan gives no results then the system has to dynamically find new plans to process the query. Even in a small system like our example the number of possible query plans rises exponentially with the number of information sources that could be available at a given time. Hence, we have to exclude the solution of hardcoding fixed strategies for executing a query and instead we propose to look for more scalable approaches.

4 Dynamic Information Integration Model

A DIIM description of an information source E in $\mathcal{E}$ contains three types of information specifying its data structures, its query capabilities as well as the relationships with other exported relations that semantically express the operations allowed between information sources. The description of the information sources will be used first in the process of query refinement of a loosely-specified query into semantically-equivalent fully specified queries. This process is constrained by using the available information sources and by being consistent with the binding patterns (i.e., query capabilities) of the selected information sources. Second, the query optimization process of the resulting set of equivalent queries is based on the information descriptions like query templates, access costs and semantic relationships among information source contents (e.g., the knowledge that a site relation contains a fragment of another external site relation could be used to prune out some of the query plans that would result in redundant data).

4.1 Data Content Description in DIIM

For every site, the description of the local data is done using n-ary relations with attributes of either primitive or complex types. The model used is a combination of the nested relational data model [KS91] and the logic-based data model [Ull89]. A relation has a unique name and fixed arity where the attributes are not explicitly named but have the values determined by their positions. In our model, every attribute has a type which can be a primitive type represented by an unary relation or a complex type represented by a name of another relation. The primitive type-relations are virtual in the sense that their extent is unknown with real data being stored only in the external information sources. The model permits us to express a generalization hierarchy among the relations by using the integrity constraints as defined in Section 4.3[1].

Example 2 *In our Example 1 the external relation* **Conspectus** $(Url, Journal, Subject)$ *contains the* Url *of primitive type* **Address** *for collections of documents published in* $Journal$ *of complex type* **Publication** *with the attribute* $Subject$ *of primitive type* **ThesaurusTerm**. *The second attribute of the relation* **Conspectus** *is of complex type expressed by the fact that the relation* **Publication** *is defined as* **Publication**$(Title, Address)$ *with* $Title$ *of primitive type* **PublicationName** *and* $Address$ *of primitive type* **Address**.

The complete content descriptions for our example from Section 3 are:

- *The relation* **Nasa_Thesaurus**$(Tword, TwordEq)$ *contains pairs of words that are synonymous. Both* $Tword$ *and* $TwordEq$ *have the primitive type* **ThesaurusTerm**.
- *The relation* **Bso_Eq**$(Bword, BwordEq)$ *contains pairs of equivalent BSO terms.* $Bword$ *and* $BwordEq$ *have the primitive type* **BsoTerm**, *a subtype of* **ThesaurusTerm**.

[1] We make the convention throughout this paper to have the names of the relations starting with a capital letter.

- *The relation* **Bso_Broader**$(Bword, BwordB)$ *contains pairs of BSO terms with the second term one level broader than the first term.* $Bword$ *and* $BwordB$ *have the primitive type* **BsoTerm**.

- *The relation* **Bso_Narrower**$(Bword, BwordN)$ *contains pairs of BSO terms with the second term one level narrower than the first term.* $Bword$ *and* $BwordN$ *have the primitive type* **BsoTerm**. *The* **Bso** *relations are stored at the same site.*

- *The relation* **Name_Authority**$(Name, NameS)$ *contains pairs of synonymous names, where* $Name$ *and* $NameS$ *have the primitive type* **GeneralName**.

- *The relation* **Ieee_Authors**$(Author, Birthday)$ *contains all the authors that published in an IEEE journal.* $Author$ *is of primitive type* **AuthorName**, *a subtype of* **GeneralName**.

- *The relation* **Conspectus**$(Url, Journal, Subject)$ *contains the* Url *of primitive type* **Address** *for collections of documents published in* $Journal$ *of complex type* **Publication** *with the topic* $Subject$ *of primitive type* **ThesaurusTerm**. *The second attribute of the relation* **Conspectus** *is of complex type expressed by the fact that the relation* **Publication** *is defined as* **Publication**$(Title, Address)$ *with* $Title$ *of primitive type* **PublicationName** *and* $Address$ *of primitive type* **Address**.

Because in our model one can define complex objects, we must have a way to specify the "path expressions" to access the attributes of an object. Thus to define a subobject of a complex object, we use a binary relation with the same name as the type of the contained object starting with a lower-case letter. For example, to indicate that the object $Title$ of primitive type **PublicationName** is a subobject of the object $Journal$ of complex type **Publication**, we use the binary relation notation **publicationName**$(Journal, Title)$.

4.2 Capability Description in DIIM

An external site description must contain information about what kinds of queries a source can handle, i.e., *the query templates* the site could answer to. Formally, a query template is a parameterized query: assigning values to the parameters of the template results in a query that a target site could handle. Using this formalism, we can uniformly describe sites with fixed APIs as well as sites with declarative languages. For example, the Unix command "finger" could be describe as a relation **Finger**$(LastName, FirstName, UniqueName, Host)$ that supports only queries for which $Host$ is provided (i.e., *bound*) and one of the other attributes is provided as well. On the other hand, if a source is an SQL database, its relations support all possible combinations of bound/free arguments for a query.

In the query process, once the query template which is going to be used for extracting information from a source (a relation) is established, the execution will consist of a sequence of operations that must be carried out to answer the query corresponding to an instantiation of that query template. Thus, the capability

descriptions should contain both the query templates and a set of rules that define the sequence of operations to be executed for an instantiated template.

In our model, if $E \in \mathcal{E}$ is an n-ary relation, the query templates for E are denoted by a set $A^E = \{\alpha_1^E, \alpha_2^E, \ldots \alpha_n^E\}$ of adornments (or binding patterns) with $\alpha_k^E[i] \in \{b, f\}$ for all $i \in \{1, 2, \ldots, n\}$ (see [Ull89]). If the ith symbol of the adornment α_k^E is b then the ith argument of the relation must be bound when a query is asked. If the ith symbol of α_k^E is f then the ith argument is free and may be returned as result of executing the query [2].

Example 3 *For example, the adornment set for the relation* **Name_Authority**$(Name, NameS)$ *is the set* $\{[bf]\}$ *which describes the fact that the relation* **Name_Authority** *could only answer queries when the first argument is given as input and the second is free. For example, the query "Retrieve all the synonymous names for* $'JoeSmith'$*" is expressed as* **Name_Authority**$('Joe\ Smith', X)$ *which is an instantiation of the query template corresponding to the binding pattern* $[bf]$*. However, the query* **Name_Authority**$(X, 'Joe\ Smith')$ *with the second argument bound to a constant is not supported given the set* $\{[bf]\}$ *of binding patterns.*

Because we represent a primitive type as an unary relation, we assume that a type-relation E has the capability set $\{[b]\}$, i.e., we can check that a given object has the type E. The same assumption is made for the complex attributes represented by the binary relations. If a complex attribute c has an attribute c_a of type **A** then this is represented by the binary relation $\mathbf{a}(c, c\_a)$ having the capability set $\{[bf]\}$, i.e., we assume that given a complex object we can "extract" its components.

4.3 Constraint Description in DIIM

In order to generate the query plans and optimize the number of sites that need to be accessed to answer a query[3], the information source description also contains information characterizing the content in terms of its relationship with other sources in the system. In our model such relative descriptions are done in two ways: (1) constraints that relate different external relations and (2) integrity constraints for a site itself.

Throughout this paper we will use E to represent a relation from $\mathcal{E}$, $\bar{X}$ to represent a tuple of variables (unless it is specified that it is a *free tuple*, i.e., it may use either constants or variables), and $C(\bar{Y})$ a conjunction of arithmetic comparison predicates, i.e., built-in predicates. A built-in predicate has the form $X\theta Y$ where $\theta \in \{<, \leq, =, \geq, >\}$, X is a variable and Y is a variable or a constant.

[2] We make the observation here that a query template describes beside the directly supported queries (i.e., the queries obtained by instantiating the template) also the indirectly supported queries. An indirected supported query could be broken down into a directly supported query and a filter applied to the result of the directed supported query. For example, a query over the relation **Finger** with three arguments set $LastName$, $FirstName$ and $Host$ could be split into a directed supported query where $LastName$ and $Host$ is set followed by a filter that chooses only the results with the given $FirstName$.

[3] Actually, in the long-term we plan to incorporate actual cost functions into individual sources and utilize this to further optimize the interpreted query beyond the means of the number of sources used.

4.3.1 Integrity Constraints

Integrity Constraints define order and type constraints over the attribute set of a relation. Assume that E is an n-ary relation and $\bar{X}$ is a n-free tuple.

Definition 1 *The order constraint for the relation E is defined as:*

$$E(\bar{X}) \subseteq C(\bar{X}) \tag{1}$$

where $C(\bar{X})$ is a conjunction of built-in predicates over the set of variables $\bar{X}$.

Another form of integrity constraint defines the primitive or complex types of the attributes of a relation.

Definition 2 *The type constraint is specified as*

$$E(X_1, X_2, \ldots, X_n) \subseteq E_1(X_1), \ldots, E_i(X_i), \ldots, E_n(X_n) \tag{2}$$

where E is an n-ary relation, $(X_1, X_2, \ldots, X_n)$ is a n-tuple of variables, and E_i is a primitive or complex type-relation for all $1 \leq i \leq n$[4].

Expressions of type (2) could be used for modeling the type-subtype relationship (see *(ic8)* below): if T_1 is a subtype of T_2 then $T_1(X) \subseteq T_2(X)$, i.e., all tuples in T_1 (i.e., of type T_1) are also in T_2 (i.e., have type T_2).

Example 4 *For our example, the complete set of integrity constraints of type (2) is given below. The constraint (ic5) defines the attribute types for the relation* **Conspectus** *while the constraint (ic8) defines the subtype relationship between the primitive types* **Bso** *and* **ThesaurusTerm**.

(ic1) **Nasa_Thesaurus**$(Tword1, Tword2) \subseteq$ **ThesaurusTerm**$(Tword1)$, **ThesaurusTerm**$(Tword2)$

(ic2) **Bso_Eq**$(Bword1, Bword2) \subseteq$ **BsoTerm**$(Bword1)$, **BsoTerm**$(Bword2)$

(ic3) **Bso_Narrower**$(Bword1, Bword2) \subseteq$ **BsoTerm**$(Bword1)$, **BsoTerm**$(Bword2)$

(ic4) **Bso_Broader**$(Bword1, Bword2) \subseteq$ **BsoTerm**$(Bword1)$, **BsoTerm**$(Bword2)$

(ic5) **Conspectus**$(Url, Journal, Subject) \subseteq$ **Address**(Url), **Publication**$(Journal)$, **ThesaurusTerm**$(Subject)$

(ic6) **Name_Authority**$(Name1, Name2) \subseteq$ **GeneralName**$(Name1)$, **GeneralName**$(Name2)$

[4] The notation we use to express the integrity constrains "$\subseteq$" is natural if we think of these constraints as a way of expressing the relationship between the extents of an n-ary relation E and the type-relations: if the ith attribute of E is of type E_i then the object set corresponding to the projection of the ith attribute from the tuple set of relation E must be a subset of the E_i extent.

(ic7) **Ieee_Author**$(Author, Birthday) \subseteq$ **AuthorName**$(Author)$, **Date**$(Birthday)$

(ic8) **Bso**$(Bword) \subseteq$ **ThesaurusTerm**$(Bword)$

(ic9) **AuthorName**$(Author) \subseteq$ **GeneralName**$(Author)$

For the rest of the paper, we assume that the conjunctive expressions have a special format called $\mathcal{E}$-*normal form* defined below. Any safe rule-based conjunctive query using relations from $\mathcal{E}$ could be transformed into $\mathcal{E}$-*normal form* as discussed below.

Definition 3 *We say that a logical rule* $r : Q(\bar{V}) := E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0)$ *is in* $\mathcal{E}$*-normal form if it has the following properties:*

P1. $\bar{V} \subseteq \bar{X}_0 \cup \bar{X}_1 \cup \ldots \cup \bar{X}_n$*, i.e., r is a safe rule.*

P2. $\bar{V}, \bar{X}_1, \ldots, \bar{X}_n$ *are tuples of variables and* $\bar{X}_0$ *is a free tuple.*

P3. All variables in $\bar{V} \cup \bar{X}_0 \cup \bar{X}_1 \cup \ldots \cup \bar{X}_n$ *are relevant to the query (i.e., any variable is used at least twice in r). If there are nonrelevant variables (i.e., anonymous variables), we replace them with '_'.*

P4. For any relevant variable X the body of the rule r contains its type relations according to the DIIM descriptions. That is, for each variable X that appears in a non-type relation as the jth argument $E(..., \underbrace{X}_{j}, ...)$ *in the body of the rule r, then* $E_j(X)$ *is also in the body of the rule r where* E_j *is the type of the jth argument defined in DIIM for the relation E. If X doesn't appear in a non-type relation then it must appear in a type relation.*

We say that an expression $E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0)$ is in $\mathcal{E}$-*normal form* if it has properties *P2* and *P4* from Definition 3. Throughout this paper, the examples use expressions that are not in $\mathcal{E}$-*normal form* as long as the context is very well understood. For example, an expression $E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0)$ could be transformed into an equivalent expression that has property *P2* if for all constants $c \in \bar{V} \cup \bar{X}_1 \cup \bar{X}_2 \cup \ldots \cup \bar{X}_n$ we do the following: (1) replace the constant c with a new variable (never used before) X_c^{new} and (2) add a new constraint $X_c^{new} = c$ to the expression $C(\bar{X}_0)$.

4.3.2 Join constraints

A join constraint relates a site to some other known site. This is done by defining how information across two disjoint sites could be integrated in a meaningful way.

Definition 4 *For two relations* E_1 *and* E_2 *in* $\mathcal{E}$*, we can define the join constraints of the form:*

$$E_1(\bar{X}), \phi_1(\bar{X}_0) \equiv_J E_2(\bar{Y}), \phi_2(\bar{Y}_0) \tag{3}$$

where the expressions ($E_1(\bar{X}), \phi_1(\bar{X_0})$), ($E_2(\bar{Y}), \phi_2(\bar{Y_0})$) *are in* $\mathcal{E}$*-normal form. The expressions* ϕ_1 *and* ϕ_2 *are conjunctions of built-in predicates, unary relations (representing the types of the variables) and binary relations used to represent path expressions. If* $(\bar{X} \cup \bar{X_0}) \cap (\bar{Y} \cup \bar{Y_0}) = \emptyset$ *we say that the join constraint is unconnected. Otherwise, the join constraint is said to be a connected join constraint.*

The semantics of a join constraint in Equation (3) is that the tuples of the two relations E_1 and E_2 could be combined using as join predicate the conjunction of ϕ_1 and ϕ_2. Specifically, if E_1 is an n-ary relation and E_2 is an m-ary relation and $\bar{X} = (X_1, \ldots, X_n)$, and $\bar{Y} = (Y_1, \ldots, Y_m)$, then the join constraint explicitly defines that the relations E_1 and E_2 could be joined using the join predicate c_{E_1,E_2} defined as follows:

$$c_{E_1,E_2} = \phi_1(\bar{X_0}) \ \wedge \ \phi_2(\bar{Y_0}) \ \wedge \{E_1.V = E_2.V^5 \mid \forall V \in (\bar{X} \cup \bar{X_0}) \cap (\bar{Y} \cup \bar{Y_0})\}. \tag{4}$$

Example 5 *The join constraints for our example are given below. The unconnected join constraint (j5) states that two tuples from the relations* **Conspectus** *and* **Ieee_Author** *respectively could be semantically combined (i.e., are join-consistent conforming to the join constraint (j5)) in a join tuple if and only if the complex attribute* $Journal$ *has the name 'ieee' in the* **Conspectus** *relation.*

(j1) **Nasa_Thesaurus**$(\_, Tword2) \equiv_J$ **Bso_Eq**$(Tword2, \_)$

(j2) **Nasa_Thesaurus**$(\_, Tword2) \equiv_J$ **Bso_Broader** $(Tword2, \_)$

(j3) **Nasa_Thesaurus**$(\_, Tword2) \equiv_J$ **Bso_Narrower** $(Tword2, \_)$

(j4) **Name_Authority**$(\_, Name2)$, **AuthorName**$(Name2) \equiv_J$ **Ieee_Authors**$(Name2, \_)$

(j5) **Conspectus**$(\_, Journal, \_)$, **publicationName** $(Journal, 'ieee') \equiv_J$ **Ieee_Authors**$(\_, \_)$

(j6) **Conspectus**$(\_, \_, Subject)$, **BsoTerm**$(Subject) \equiv_J$ **Bso_Eq**$(\_, Subject)$

(j7) **Conspectus**$(\_, \_, Subject)$, **BsoTerm**$(Subject) \equiv_J$ **Bso_Broader**$(\_, Subject)$

(j8) **Conspectus**$(\_, \_, Subject)$, **BsoTerm**$(Subject) \equiv_J$ **Bso_Narrower**$(\_, Subject)$

4.3.3 Partial and Complete Information

The partial and complete information description makes it possible to describe that a fragment of a relation is part of or equal to a fragment of another relation for all instances of the two relations.

Definition 5 *For two relations* E_1 *and* E_2 *the partial/complete information constraint*

$$E_1(\bar{X}), \phi_1(\bar{X_0}) \ \theta \ E_2(\bar{Y}), \phi_2(\bar{Y_0}) \tag{5}$$

[5] I.e., the "natural" join attributes are defined in the expression 3 by using the same variables names in the lefthand and righthand sides.

specifies that[6]*:*

$$\pi_{\bar{Y}\cap\bar{X}}(\sigma_{\phi_1(\bar{X}_0)}E_1(\bar{X}))\ \theta\ \pi_{\bar{Y}\cap\bar{X}}(\sigma_{\phi_2(\bar{Y}_0)}E_2(\bar{Y})) \tag{6}$$

where θ *is* $\subset$ *or* $=$ *for the partial or complete information constraint, respectively.*

Example 6 *The partial and complete information description will be used by the Query Evaluation Agent in the process of optimization and execution of the query plans. For example, let's assume that a new site joins the reference space having the relation* **Technical_Authors**$(Name, Company, Journal)$*, an author index with authors that published in engineering journals, their work addresses and the journal they published in. The fact that for all authors found in the* **Ieee_Authors** *relation there exists an entry with the same author name in* **Technical_Authors** *could be expressed using the complete information description as in (5).*

- *(ci1)* $\underbrace{\textbf{Ieee\_Authors}(Name, \_)}_{E_1(\bar{X})}, \underbrace{\textbf{AuthorName}(Name)}_{\phi_1(\bar{X}_0)} = \underbrace{\textbf{Technical\_Authors}(Name, \_, Journal)}_{E_2(\bar{Y})},$
  $\underbrace{\textbf{Publication}(Journal), \textbf{publicationName}(Journal, Y), Y =' ieee'}_{\phi_2(\bar{Y}_0)}$

When executing a query plan that has to access the relation **Technical_Authors** *in order to retrieve the author names for a given* $Journal$*, the complete information description (ci1) states that the relation* **Ieee_Authors** *could be accessed instead (we obtain the same result as going to* **Technical_Authors** *relation) if the* $Journal$ *name is* $'ieee'$*. The decision should be made based on the costs associated with the two relations.*

5 Query Semantics in DIIM

5.1 DIIM Query Language

A query in DIIM is formulated by defining a set of constraints over the domain concepts (e.g., ontology) that are to participate in the query-result.

Definition 6 *A DIIM query is formulated as a logical rule in* $\mathcal{E}$*-normal form as defined in Section 4.3, Definition 3:*

$$Q(\bar{V}) : -E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0) \tag{7}$$

where $C(\bar{X}_0)$ *is a conjunction of built-in predicates,* $E_i \in \mathcal{E}$ *for* $1 \leq i \leq n$ *are relations and* $\bar{V} \subseteq \bar{X}_0 \cup \bar{X}_1 \cup \ldots \cup \bar{X}_n$ *a set of variables defining the query-result IDB relation*[7]*.*

With this syntax, users can either simply list attributes (concepts) of interest they want to retrieve without giving any constraints, or they can make the query more precise by adding constraints as built-in predicates

[6] σ represents the selection operator and π represents the projection operator.

[7] For naive users, a query interface could be designed as a filled-in form that could be automatically translated into the internal system query language. The filled-in form would contain the ontology of terms as well as built-in predicates.

(query (8) in Example 7) or imposing an information source from which to acquire the data (query (9) in Example 7).

Example 7 *Here are two examples of queries of type (7):*

(1) The query "Retrieve URL addresses of on-line collections, their subjects and the participating authors for 'ieee' journals" is expressed in DIIM:

$$Q(U, S, A) : -\textbf{Address}(U), \textbf{BsoTerm}(S), \textbf{AuthorName}(A), \textbf{Publication}(J), \textbf{publicationName}(J, \textit{'ieee'}) \quad (8)$$

(2) If the user wants to impose an information source relation to be used for his request[8] *then the relation could be specified in the body of the rule. For example, the query "Retrieve URL addresses of on-line collections, their subjects and the participating authors that are found in the information source exporting the relation* **Technical_Authors***" is expressed in DIIM:*

$$Q(U, S, A) : -\textbf{Address}(U), \textbf{BsoTerm}(S), \textbf{AuthorName}(A), \textbf{Technical\_Authors}(A, \_, \_) \quad (9)$$

5.2 Query Specification Refinement

5.2.1 Steps of Query Processing

The DIIM query language permits loosely-specified queries which in general cannot be evaluated given the permissible adornments for the external relations that appear in the body of the query rule. For example, if the query rule is expressed using only type relations (like in the above query (8) in Example 7) then the answer to the query cannot be constructed by directly accessing the type relations. The reason is that primitive type relations are virtual, i.e. their extents are not explicitly stored[9]. Thus the DIIM system will first "refine" a query rule into the set of semantically-equivalent feasible query rules by combining data from existing information sources when meaningful. These equivalent refined query rules will be shown to be consistent with the integrity and join constraints declared in DIIM.

There are three stages of query execution in DIIM: (1) *Refinement Stage.* Given a loosely-specified user query, the system will use the information source descriptions to refine the query into a semantically-equivalent set of queries in the form of query access plans. All generated plans will be guaranteed to have feasible orders for their steps (subgoals), i.e., all needed bound variables for a subgoal are obtained from previous subgoals or are bound by an equality built-in predicate. (2) *Optimization Stage.* We then identify the minimum set of query rules from the complete set of refined query rules. At this stage we will use partial and complete information descriptions to eliminate from the complete set of refined rules, the queries

[8] This option could come from the past experience with the same kind of query.

[9] Complex type relations could be stored if they are relations exported by an information In this case the objects of complex types are identifiers of objects stored at the information sources exporting the complex type relations.

that are contained in some other queries. (3) *Query-Execution-Plan Generation Stage.* The result of the Optimization Stage, i.e., the set of query access plans, is used to generate query execution plans.

In this section we introduce the notion of *connected relation* as being the result relation defined by a loosely-specified query that cannot be evaluated given its definition. We show that the connected relation is a natural generalization of the notion of the relation defined by a feasible query rule and we generalize the well-known containment theorem for loosely-specified queries.

5.2.2 Extended Notion of Containment

We give next some basic definitions that we use in the rest of the paper.

Definition 7 *Let r_1 and r_2 be two conjunctive query rules with ordinary and built-in subgoals. We say that r_1 is **contained** in r_2, $r_1 \subseteq r_2$, if the relation for the head predicate defined by r_1 is contained in the relation for the head predicate defined by r_2 for all instances of the relations that appear in the bodies of r_1 and r_2.*

Definition 8 *Let r_1 and r_2 be two conjunctive query rules in $\mathcal{E}$-normal form where every symbol '_' was replaced with a unique variable name. A **containment mapping** h from r_2 to r_1 is a symbol mapping from the symbols of the query r_2 to the symbols of the query r_1 with h equal to identity for constants, relation names and built-in predicate symbols. Moreover, h maps the head of the query r_2 into the head of the query r_1 and every ordinary subgoal of r_2 into an ordinary subgoal of r_1.*

To show that one conjunctive query is contained in another conjunctive query, we have the following well known theorem (see [Ull89]).

Theorem 1 *Let r_1 and r_2 be two conjunctive queries with built-in subgoals for which the following holds:*

1. *There is a containment mapping h from the symbols of r_2 to the symbols of r_1 that satisfies Definition 8. That is, for each $R(\bar{X})$ an ordinary subgoal of r_2, $h(R(\bar{X}))(= R(h(\bar{X})))$ is an ordinary subgoal of r_1; and if $Q_2(\bar{V_2})$ is the head of the query r_2 then $h(Q_2(\bar{V_2}))$ is the head of the query r_1, i.e., $h(Q_2(\bar{V_2})) = Q_1(h(\bar{V_2})) = Q_1(\bar{V_1})$.*

2. *For each C a built-in subgoal of r_2, $h(C)$ is implied by the built-in subgoals of r_1.*

Then r_1 is contained in r_2, i.e., $r_1 \subseteq r_2$.

Definition 7 applies to two queries r_1 and r_2 that could be evaluated given their definitions. For the purpose of our work, we extend the notion of ***containment*** for two queries that *cannot* be evaluated given their definitions (examples of such queries are query rules (8) and (9 in Example 7).

Definition 9 *Let r_1 and r_2 be two conjunctive query rules with ordinary and built-in subgoals. We say that r_1 is a **restriction** of r_2 if there exists a containment mapping h from the symbols of r_2 to the symbols of r_1 that satisfies Definition 8 plus for each built-in subgoal C of r_2, $h(C)$ is implied by built-in subgoals of r_1.*

If queries r_1 and r_2 could be evaluated over $\mathcal{E}$ and r_1 is a restriction of r_2 then r_1 is contained in r_2 by Theorem 1. Our purpose here is to extend the containment definition for two queries that could not be directly evaluated against $\mathcal{E}$, i.e., those high-level vague queries permitted in our system. We will define the result of such a query to be the relation defined by the union of all relations obtained from the "good" rewritten rules of the original queries and show that Theorem 1 still applies for this extended definition.

We now extend Definition 9 for two conjunctive expressions in *$\mathcal{E}$-normal form.*

Definition 10 *Let ϕ_1 and ϕ_2 be two conjunctive expressions in $\mathcal{E}$-normal form with ordinary and built-in subgoals. We say that ϕ_1 is a* ***restriction*** *of ϕ_2 if there exists a containment mapping h from the symbols of ϕ_2 to the symbols of ϕ_1 with the properties defined in Definition 8 plus for each built-in subgoal C of ϕ_2, $h(C)$ is implied by the built-in subgoals of ϕ_1.*

The next definition introduces the notion of a partition defined by a relevant variable of a query r by being the set of subgoals that participate in a defined DIIM join constraint in query r.

Definition 11 *Let r be a conjunctive query rule with built-in subgoals in $\mathcal{E}$-normal form. Let X be a variable in the variable set of r. We define the* ***partition*** *of the query r relative to a variable X by:*

$part_r(X) = \{p(\bar{Y}) \mid$ ***(I)*** *$p(\bar{Y})$ is a subgoal of r such that $X \in \bar{Y}$ or*
(II) *there exists $q(\bar{Y'}) \in part_r(X)$ such that there exists a join constraint as in Equation (3) $R(\bar{V}), \phi(\bar{V_0}) \equiv_J R'(\bar{Z}), \phi'(\bar{Z_0})$ such that the expression $R(\bar{V}), \phi(\bar{V_0}), R'(\bar{Z}), \phi'(\bar{Z_0})$ is a restriction (according to Def. 10) of the body of query-rule r and it contains $p(\bar{Y})$ and $q(\bar{Y'})$ as subgoals $\}$.*

If $\bar{X}$ is a subset of the variable set of r, we define $part_r(\bar{X}) = \cup_{X \in \bar{X}} part_r(X)$.

Example 8 *Let r be:*

$Q(U, A) :-$ **Ieee_Author**$(A, \_)$, **AuthorName**(A), **Conspectus**(U, J, T), **Address**(U), **Publication**(J), **publicationName**(J, N), **PublicationName**(N), $N =$ *'ieee'*, **ThesaurusTerm**$(T), T =$ *'greenhouse effect'*.

$part_r(U) =$ {**Conspectus**(U, J, T), **Ieee_Author**$(A, \_)$, **AuthorName**(A), **Conspectus**(U, J, T), **Address**(U), **Publication**(J), **publicationName**(J, N), **PublicationName**(N), $N =$*'ieee'* , **ThesaurusTerm**(T) }.

For constructing $part_r(U)$ we start with the subgoals **Conspectus**(U, J, T) *and* **Address**(U) *that satisfy property* **(I)** *from Definition 11. Then the subgoals* **Ieee_Author**$(A, \_)$, **AuthorName**(A), **Publication**(J), **publicationName**(J, N), **PublicationName**(N), $N =$ *'ieee'*, *and* **ThesaurusTerm**(T) *are included in $part_r(U)$ because they satisfy property* **(II)**: *there exists* **Conspectus**(U, J, T) *in $part_r(U)$ and the join constraint (j5) such that the expression* **Conspectus**$(X_1, Journal, X_2)$, **Publication**$(Journal)$, **publicationName**$(Journal, X)$, **PublicationName**(X) $X =$*'ieee'*,

Ieee_Authors(X_3, X_4) *and* **AuthorName**(X_3) *is a restriction (according to Def. 10) of the body of rule r.*

5.2.3 The Connected Query Rule Concept

We now give our connected rule definition that describes the constraints we impose for refining a vague user query into a more precise rules computable given our system resources. The proposed planning algorithm is then guaranteed to generate a set of plans that correspond to the set of connected rules for the given user query. In other systems, the user is typically required to specify his queries directly in the connected rule format and the system just executes them [PGMU95, PGMW95, GMHI+95].

Definition 12 ***Connected Query Rule in DIIM***

Let Q be a query in $\mathcal{E}$-normal form defined by the logical rule r:

$$r: \quad Q(\bar{V}) :- E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0) \tag{10}$$

We say that a rule r^c defined by:

$$r^c: \quad \underbrace{Q(\bar{V}) :- E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0)}_{r}, R_1(\bar{Y}_1), R_2(\bar{Y}_2), \ldots, R_k(\bar{Y}_k) \tag{11}$$

is a ***connected rule*** *for the query (10) if and only if the following conditions hold:*

1. ***Restriction:*** *r^c is a restriction of r (according to Def. 9) such that the containment mapping is the identity function and the set of built-in subgoals of r^c is equal to the set of built-in subgoals of r.*

2. ***Minimum Joins:*** *A relation from $\mathcal{E}$ appears only once in the body of the query rule r^c.*

3. ***Feasible:*** *r^c has a $\mathcal{E}$-feasible order for its subgoals (i.e., there is a feasible order for the subgoals of r^c such that the adornment for each subgoal $R(\bar{X})$ is permissible (given the adornment set of R) and correct. I.e., all the bound variables are obtained from the previous subgoals or are bound by a built-in subgoal of the form $X = c$ with c being a constant).*

4. ***Minimum Rewriting:*** *$part_{r^c}(\bar{V} \cup \bar{X}_0 \cup \bar{X}_1 \cup \ldots \cup \bar{X}_n) = \{E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0), R_1(\bar{Y}_1), R_2(\bar{Y}_2), \ldots, R_k(\bar{Y}_k)\}$, i.e., all subgoals are part of the partition generated over r^c by the original set of variables of query r.*

5. ***Connected:*** *The rule r^c represents connections among the relations in $\mathcal{E}$ that are consistent with the initial query. Formally, the following property must hold:*

   For any variable $X \in \bar{V} \cup \bar{X}_1 \cup \ldots \cup \bar{X}_n$, $part_{r^c}(X) \supseteq \{E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), R_1(\bar{Y}_1), R_2(\bar{Y}_2), \ldots, R_k(\bar{Y}_k)\}$, i.e., the partition of any variable contains all subgoals of r with the possible exception of the built-in subgoals ($C(\bar{X}_0)$ is not necessary in $part_{r^c}(X)$).

6. ***Maximal***: *r^c is maximal with the above properties. I.e., $\nexists r'^c$ with the above properties such that $r^c \subseteq r'^c$ and $r^c \neq r'^c$.*

Definition 13 *For a given query rule r, we define the set of the equivalent query rules $\bar{r}_{/\mathcal{E}}$ as being the set of its connected rules.*

Observation 1 *If r is a query rule that has a connected rule equal to itself then $\bar{r}_{/\mathcal{E}} = \{r\}$ following from* ***Restriction*** *and* ***Maximal*** *properties of Definition 12. Assume that there exists another connected query rule r^c such that $r^c \neq r$. From Definition 9 and property* ***Restriction*** *we have that $r^c \subseteq r$. But, because r is itself a connected query rule, this contradicts the* ***Maximal*** *property. Then the assumption that there exists a connected query rule not equal to r is false. We will call such a query rule a connected query rule.*

The above definition formalizes the fact that a refined ***connected rule*** corresponding to a query must represent some connections among its subgoals and at the same time be a restriction of the original query. This semantics of how a user query could be refined in order to be evaluated is imposed to assure that not arbitrary unrelated items are put together (since there would be an exponential number of such possible combinations) all likely to be meaningless. Note that our model makes it possible to semantically describe the join constraints that must be used in the process of rule generation.

Observation 2 *Let r^c be a connected rule in $\bar{r}_{/\mathcal{E}}$. If $E(\bar{X})$ is a subgoal in the body of r^c and there exists an integrity constraint of the form given in Equation 1, $E(\bar{X}) \subset C_E(\bar{X})$ then r^c is equivalent to the augmented query rule obtained by adding $C_E(\bar{X})$ to the body of the query r^c. Indeed, an integrity constraint of the form given in Equation 1 holds for any instance of the relation E which implies that it must hold for the tuples of E that are used in evaluating the relation defined by r^c. Following this observation, it is safe to assume that $\bar{r}_{/\mathcal{E}}$ contains the augmented connected query rules obtained by adding built-in subgoals corresponding to DIIM integrity constraints and for which the conjunction of the built-in subgoals is satisfiable (i.e., there are no contradictions).*

Example 9 *Let* **Ieee_Author**$(\_, BD) \subset BD \geq$ *'1950' be a DIIM integrity constraint. Then the rule*

$Q(U, A) :-$ **Ieee_Author**(A, D), **Date**$(D), D >$ *'1945'*, **AuthorName**(A), **Conspectus**(U, J, T), **Address**(U), **Publication**(J), **publicationName**$(J,$ *'ieee'*$)$, **ThesaurusTerm**$($*'greenhouse effect'*$)$

is equivalent to

$Q(U, A) :-$ **Ieee_Author**(A, D), **Date**$(D), D >$ *'1945'*, **AuthorName**(A), **Conspectus**(U, J, T), **Address**(U), **Publication**(J), **publicationName**$(J,$ *'ieee'*$)$, **ThesaurusTerm**$($*'greenhouse effect'*$)$, $D >$ *'1950'*

I.e., we can add this additional constraint to the query to refine it.

Definition 14 *Let $\bar{r}$ be a set of rules $r_1, r_2, \ldots, r_n$, having the same head*[10]*. Then the* ***relation defined by*** *$\bar{r}$ is the union of the relations defined by $r_1, r_2, \ldots, r_n$. Let r be a conjunctive query and $\bar{r}_{/\mathcal{E}}$ the set of its connected query rules. We call the* ***connected relation*** *$\mathcal{CR}_{/\mathcal{E}}(r)$ defined by r the relation defined by $\bar{r}_{/\mathcal{E}}$.*

Definition 15 *Given two query sets $\bar{r_1}$ and $\bar{r_2}$ such that the queries in each set have the same head, we say that $\bar{r_1}$ is* ***contained*** *in $\bar{r_2}$ and write $\bar{r_1} \sqsubseteq \bar{r_2}$ iff the relation defined by $\bar{r_1}$ is contained in the relation defined by $\bar{r_2}$*[11]*.*

The following theorem is a generalization of the containment theorem (Theorem 1) that holds for the connected relations defined by two loosely-specified queries.

Theorem 2 *Let r_1 and r_2 be two conjunctive queries. If r_1 is a restriction of r_2 (according to Def. 9) with the same set of built-in subgoals and $\bar{r_2}_{/\mathcal{E}}$ is not empty then $\mathcal{CR}_{/\mathcal{E}}(r_1)$ is contained in $\mathcal{CR}_{/\mathcal{E}}(r_2)$, i.e. $\bar{r_1}_{/\mathcal{E}} \sqsubseteq \bar{r_2}_{/\mathcal{E}}$.*

Proof: Let t be a tuple from $\mathcal{CR}_{/\mathcal{E}}(r_1)$. Then there exists a connected rule r_1^c of r_1 such that t is in the relation defined by r_1^c. We must show that there exists r_2^c a connected rule for r_2 such that $t \in r_2^c$. We claim that r_1^c is a connected rule for r_2 except possible the property **Maximal**. From Definitions 12 and 9 and the hypothesis, we have that r_1^c is a restriction of r_1 (property ***Restriction***) which is a restriction of r_2. Then r_1^c has all the properties from Definition 12 relative to r_2 except possible the property ***Maximal***. If there exists $r_2^c \in \bar{r_2}_{/\mathcal{E}}$ such that $r_1^c \subseteq r_2^c$ then t is a tuple in the relation defined by r_2^c. If no such rule exists then r_1^c is itself a connected rule for r_2. In both cases we have that $t \in \mathcal{CR}_{/\mathcal{E}}(r_2)$.

Observation 3 *If r_1 and r_2 are two connected query rules then $\bar{r_1}_{/\mathcal{E}} = \{r_1\}$ and $\bar{r_2}_{/\mathcal{E}} = \{r_2\}$ and Theorem 2 is equivalent to Theorem 1. In this case the extended definition of containment is equivalent to the original containment definition as given in Definition 7.*

The following theorem shows that the full disjunction is the default case for our concept of connected relation when all (arbitrary) natural joins are permitted between sources. For obvious reasons, we developed in this paper the tightened hence more practical definition of combining data from disparate sources to construct answers to a query.

Theorem 3 *Let $\mathcal{E}$ contain a set of relations $\{R_1, R_2, \ldots, R_n\}$ over a set of primitive attribute types $\{E_1, E_2, \ldots, E_m\}$. The relations accept any adornment and the set of join constraints is defined to be all the natural joins. Let $\mathcal{R}$ be the "natural" full disjunction ([GL94], [RU96]) of the relations $R_1, R_2, \ldots, R_n$.*

[10] $\bar{r}$ is a datalog program defining the same intensional relation over the variables $\bar{V}$.I.e., if $\bar{r} = \{r_1, r_2, \ldots, r_n\}$ then the heads for the rules $r_1, r_2, \ldots$ and r_n are $Q_1(\bar{V}), Q_2(\bar{V}), \ldots$ and $Q_n(\bar{V})$, respectively, defining relations over the variables $\bar{V}$.

[11] Note that Definition 15 is stronger than the following definition: $\bar{r_1}$ is contained in $\bar{r_2}$ iff $\forall r_1 \in \bar{r_1}, \exists r_2 \in \bar{r_2}$ such that r_1 is contained in r_2.

Let $\{E_{i_1}, E_{i_2}, \ldots, E_{i_k}\} \subseteq \{E_1, E_2, \ldots, E_m\}$ *be a subset of attribute types and* $r_{E_{i_1},E_{i_2},\ldots,E_{i_k}}$ *be the following query rule:*

$$r_{E_{i_1},\ldots,E_{i_k}} : Q(\bar{X}) : -E_{i_1}(X_1), E_{i_2}(X_2), \ldots, E_{i_k}(X_k) \tag{12}$$

where $\bar{X} = (X_1, X_2, \ldots, X_k)$.

Then $\mathcal{CR}_{/\mathcal{E}}(r_{E_{i_1},\ldots,E_{i_k}}) = \pi_{X_1,X_2,\ldots,X_k}(\sigma_{X_1 \neq nil \wedge \ldots \wedge X_k \neq nil}\mathcal{R})$.

Proof "$\supseteq$". The full disjunction $\mathcal{R}$ of the relations $R_1, \ldots, R_n$ contains all tuples of the relations combining them as a single tuple whenever they match. Let $t \in \pi_{X_1,X_2,\ldots,X_k}(\sigma_{X_1 \neq nil \wedge \ldots \wedge X_k \neq nil}\mathcal{R})$. Then there exists $t' \in \mathcal{R}$ such that nonnull components of t' contain (at least) the attributes corresponding to $X_1, \ldots, X_k$. Then there exists a connected subset $\{R_{j_1}, \ldots, R_{j_l}\}$ of the relation set $\{R_1, \ldots, R_n\}$ such that (1) schema of the connected set $\{R_{j_1}, \ldots, R_{j_l}\}$ contains the attributes corresponding to $X_1, \ldots, X_k$ and (2) there exist tuples $t_1 \in R_{j_1}, \ldots, t_l \in R_{j_l}$ which are join-consistent (nonnull) for any attribute corresponding to $X_1, \ldots, X_k$ and agree with each others and with t on these attributes. Let r^c be the rule (in $\mathcal{E}$-normal form):

$$r^c : Q(\bar{X}) : -R_{j_1}(\bar{Y_1}), \ldots, R_{j_l}(\bar{Y_l}), E_{i_1}(X_1), E_{i_2}(X_2), \ldots, E_{i_k}(X_k), E_{m_1}(V_1), \ldots, E_{m_s}(V_s) \tag{13}$$

where $\{X_1, \ldots, X_k\} \cup \{V_1, \ldots, V_s\} = \bar{Y_1} \cup \ldots \cup \bar{Y_l}$ and $E_{m_1}, \ldots, E_{m_s}$ are the types of the attributes corresponding to $V_1, \ldots, V_s$, respectively. The relation defined by r^c contains t (the body of the rule r^c generates t'). We show that r^c is a connected rule (with possible exception of **Maximal** property) for the rule $r_{E_{i_1},\ldots,E_{i_k}}$. By construction r^c has the properties **Restriction** and **Minimum Joins** of Definition 12. From hypothesis, r^c has the property **Feasible**. The properties **Minimum Rewriting** and **Connected** are true following the fact that the set $\{R_{j_1}, \ldots, R_{j_l}\}$ is a connected subset of the relation set. If r^c does not have the property **Maximal** then there exists a connected rule r'^c of $r_{E_{i_1},\ldots,E_{i_k}}$ such that $r^c \subseteq r'^c$, which implies that t is in the relation defined by r'^c. If no such rule exists, then r^c is itself a connected rule for $r_{E_{i_1},\ldots,E_{i_k}}$. In both cases, we have that $t \in \mathcal{CR}_{/\mathcal{E}}(r_{E_{i_1},\ldots,E_{i_k}})$.

"$\subseteq$". Let $t \in \mathcal{CR}_{/\mathcal{E}}(r_{E_{i_1},\ldots,E_{i_k}})$. Then there exists a connected rule r^c for the rule $r_{E_{i_1},\ldots,E_{i_k}}$ such that t is in the relation defined by r^c. By Definition 12 there exists a subset $\{R_{j_1}, \ldots, R_{j_l}\}$ of the relation set such that r^c is defined as following (in $\mathcal{E}$-normal form):

$$r^c : Q(\bar{X}) : -R_{j_1}(\bar{Y_1}), \ldots, R_{j_l}(\bar{Y_l}), E_{i_1}(X_1), E_{i_2}(X_2), \ldots, E_{i_k}(X_k), E_{m_1}(V_1), \ldots, E_{m_s}(V_s) \tag{14}$$

where $\{X_1, \ldots, X_k\} \cup \{V_1, \ldots, V_s\} = \bar{Y_1} \cup \ldots \cup \bar{Y_l}$ and $E_{m_1}, \ldots, E_{m_s}$ are the types of the attributes corresponding to $V_1, \ldots, V_s$, respectively. From the property **Connected** of the Definition 12, we have that the subset $\{R_{j_1}, \ldots, R_{j_l}\}$ is a connected subset of the relation set $\{R_1, \ldots, R_n\}$. Moreover, there exist

$t_1 \in R_{j_1}, \ldots, t_l \in R_{j_l}$ such that $t_1, \ldots, t_l$ are join-consistent and they agree (and are nonnull) with t on the attributes corresponding to $X_1, \ldots, X_k$. Then, according to the definition of the full disjunction $\mathcal{R}$ the tuple t' obtained by padding t with null for all other attributes of the relation $\mathcal{R}$, must be subsumed by a tuple $t'' \in \mathcal{R}$. Thus, t is obtained by projecting t'' on the attributes corresponding to $X_1, X_2, \ldots, X_k$ and $t'' \in \sigma_{X_1 \neq nil \wedge \ldots \wedge X_k \neq nil}\mathcal{R}$ given that all the components corresponding to $X_1, \ldots, X_k$ are nonnull. Then, $t \in \pi_{X_1, X_2, \ldots, X_k}(\sigma_{X_1 \neq nil \wedge \ldots \wedge X_k \neq nil}\mathcal{R})$.

6 Restriction to the Relevant Subspace of a Query

Given a potentially big information space including most probably unrelated sources, a user query targets only a small subspace of the sources and thus, the query process should be limited to that subspace. We call the set of relevant sites **the relevant subspace** of the query. In this section we give a formal definition as well as an algorithm for computing the relevant subspace for a given query that depends on both the content of the information sources and their capabilities (i.e., query templates). If the relevant subspace of a query is empty then the query cannot be answered in the current system configuration. However, the smaller the subspace we consider during query processing the better the performance of the planning algorithm can be expected to be.

Example 10 *Consider our Example 2 space with the addition of the relation* **Finger**($LastName$, $FirstName$, $UniqueName$, $Host$) *as defined in Section 4.2. A query "Find all the email addresses for the authors that published in an IEEE journal" could not be answered by the system because the only source for the email addresses is the relation* **Finger** *which requires the $Host$ name as input that in turn could not be "discovered" somewhere else. Our goal is to automatically determine if a query could be answered in the current system configuration and, if yes, what information sources are relevant for a given query, i.e. its relevant subspace.*

Definition 16 *For a relation E with arity n, we define the set of join constraints of its i'th attribute relative to the relation E, $C_E(i) = \{j_{E,E'} \mid j_{E,E'}$ is a DIIM connected join constraint defined for the relations E and E', with $E' \in \mathcal{E}$ such that i'th attribute is a join attribute $\}$.*

Example 11 *In the join constraint (j6) (see Example 5), the join attribute for the relation* **Conspectus** *is the third attribute of type $BsoTerm$ while for the relation* **Bso_Eq** *it is the second attribute of the same type. Note that unconnected join constraints have no join attributes.*

Definition 17 ***Capability Graph*** *Given a set of external information source descriptions $\mathcal{E}$ containing DIIM capability and content descriptions, we define a directed graph $\mathcal{G}_\mathcal{E} = (\mathcal{N}, \mathcal{D})$ with $\mathcal{N}$ the set of nodes and $\mathcal{D}$ the set of edges, as follows:*

- *Step 0.* $\mathcal{N} = \emptyset, \mathcal{D} = \emptyset$

- *Step 1. For every type-relation $T \in \mathcal{E}$, we add a new node to $\mathcal{N}$: $\mathcal{N} = \mathcal{N} \cup \{(T, [bf])\}$.*

- *Step 2. For every relation $E \in \mathcal{E}$ and for every $\tilde{\alpha} \in \tilde{A}^E$ ($\tilde{A}^E$ is the minimal adornment set for E[12]), we add a new node to the node set $\mathcal{N}$: $\mathcal{N} = \mathcal{N} \cup \{(E, \tilde{\alpha})\}$.*

  The binary relations used to access the components of the complex type attributes have the binding pattern $[bf]$ as stated in Section 4 and we add only a node corresponding to this adornment set to $\mathcal{N}$.

- *Step 3. For every non-type relation $E \in \mathcal{E}$ for which a DIIM integrity constraint of the form (2) exists: $E(X_1, X_2, \ldots, X_n) \subseteq E_1(X_1), \ldots, E_n(X_n)$ (in our model defining the types of the attributes), and for all $\tilde{\alpha} \in \tilde{A}^E$ and i such that $\tilde{\alpha}[i] = f$, we add an edge $(E, \tilde{\alpha}) \stackrel{1, C_E(i)}{\longrightarrow} (E_i, [bf])$ to $\mathcal{D}$. In Figure 2 the edges added at this step are called "Output edges" as they represent the objects that could be outputs from the non-type relations.*

- *Step 4. For every type-relation $T \in \mathcal{E}$ for which a DIIM integrity constraint of the form (2) exists: $T(X) \subseteq T_1(X)$ (i.e., defining subtype relationship between T and T_1), we add an edge $(T, [bf]) \stackrel{1, \emptyset}{\longrightarrow} (T_1, [bf])$ to $\mathcal{D}$.*

- *Step 5. For every non-type relation $E \in \mathcal{E}$ for which a DIIM integrity constraint of the form (2) exists: $E(X_1, X_2, \ldots, X_n) \subseteq E_1(X_1), \ldots, E_n(X_n)$ and for all $\tilde{\beta} \in \tilde{A}^E$ and j such that $\tilde{\beta}[j] = b$, we add an edge $(E_j, [bf]) \stackrel{j, C_E(j)}{\longrightarrow} (E, \tilde{\beta})$ to $\mathcal{D}$.*

  In the Figure 2 the edges added at step 5 are called "Input edges" as they represent the input objects used to instantiate a query template for a non-type relation.

- *Step 6. For every unconnected join constraint j_{E_1,E_2} of the form (3) $E_1(\bar{X}_1), C_1(\bar{X}) \equiv_J E_2(\bar{Y}_1), C_2(\bar{Y})$, we add a new node (j_{E_1,E_2}, α) to $\mathcal{N}$, with $\alpha = [\underbrace{b \ldots b}_{n\,times} \underbrace{b \ldots b}_{m\,times}]$ where n and m are the arities of E_1 and E_2, respectively. For each attribute i of type E_i of the relation E_1 we add an edge $(E_i, [bf]) \stackrel{i, j_{E_1,E_2}}{\longrightarrow} (j_{E_1,E_2}, \alpha)$. For each attribute j of the type E_j of the relation E_2 we add an edge $(E_j, [bf]) \stackrel{n+j, j_{E_1,E_2}}{\longrightarrow} (j_{E_1,E_2}, \alpha)$ to the edge set $\mathcal{D}$. In the Figure 2 the edges added at this step are called "Join edges".*

Definition 18 ***Query Capability Graph*** *and* ***Relevant Subspace***

Given a query $r : Q(\bar{V}) : -E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0)$ in $\mathcal{E}$-normal form and the capability graph for $\mathcal{E}, \mathcal{G}_\mathcal{E} = (\mathcal{N}, \mathcal{D})$, we define the query capability graph for the query r, $\mathcal{G}_r = (\mathcal{N}_r, \mathcal{D}_r)$ and its $RelevantSubspace_r = (\mathcal{R}_r, \mathcal{JC}_r)$ as follows:

[12] Ullman defines in [Ull89] the adornment set and the partial order relation "$\leq$" over it as follows: if A is an adornment set and $\alpha, \beta \in A$ two adornments, we say that $\alpha \leq \beta$ iff β has b in every position where α has b. We now define the minimal adornment set $\tilde{A} \subseteq A$ to be the minimal set so that $\forall \alpha \in A$ there exists $\tilde{\alpha} \in \tilde{A}$ with $\tilde{\alpha} \leq \alpha$. The computation of $\tilde{A}$ is straightforward given that A is a finite set and it is clear that it is uniquely defined.

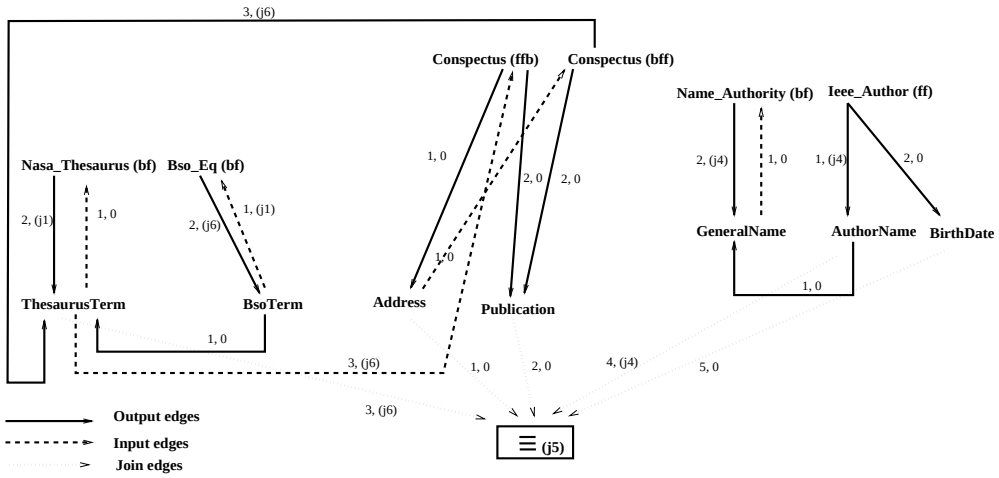

Figure 2: Capability Graph for Example 1.

In constructing the query capability graph corresponding to the query r, we start with the capability graph for $\mathcal{E}$ and mark the nodes that could be used in constructing connected rules. For each node N_i corresponding to a type-relation E_i the algorithm keeps a set of join constraints, denoted by $const(N_i)$, indicating first that an object of type E_i could be bound, and second what are the join constraints in which this object could participate. We have a special join constraint r indicating that an object of type E_i is bound by the query.

- *Step 1. $Marked = \emptyset$, $\forall N \in \mathcal{N}$, $const(N) = \emptyset$*

- *Step 2. For every variable $X \in \bar{X}_0 \cup \bar{X}_1 \cup \ldots \cup \bar{X}_n$ that is bound in the query, i.e., there exists a constraint $X = c$ where c is a constant in $C(\bar{X}_0)$ or there exists $X = Y$ and Y is bound in $C(\bar{X}_0)$, if $E_i(X)$ is the type of X in the query r and N_i is the node corresponding to E_i then (1) $Marked = Marked \cup \{N_i\}$ and (2) $const(N_i) = \{r\}$. The above construction is a representation of the fact that X could be used as an input for any E^α with the adornment $\alpha \in \tilde{A}^E$ such that $\alpha[i] = b$.*

- *Step 3. For every relation E that has a free adornment α (i.e., $\alpha[i] = f$ for all i) and its corresponding node N do (1) $Marked = Marked \cup \{N\}$ and (2) for every type-relation node N_i corresponding to the type-relation E_i denoting the type of the i'th attribute of E do (2.1) $const(N_i) = const(N_i) \cup C_E(i) \cup \{j_{E,E'} \mid j_{E,E'}$ is an unconnected join constraint $\}$, and (2.2) $Marked = Marked \cup \{N_i\}$. I.e., objects of type E_i could be obtained from E and they could participate in joins defined for E.*

- *Step 4. For every relation E and $\alpha \in \tilde{A}^E$ such that for every i with $\alpha[i] = b$, $const(N_i) \cap C_E(i) \neq \emptyset$ or $const(N_i) \supseteq \{r\}$ do (1) $Marked = Marked \cup \{N\}$ (N is the node corresponding to E^α) (2) for every type-relation node N_j corresponding to the type-relation E_j denoting the type of the j'th attribute of E do (2.1) $const(N_j) = const(N_j) \cup C_E(j) \cup \{j_{E,E'} \mid j_{E,E'}$ is an unconnected join constraint $\}$. (2.2)*

$Marked = Marked \cup \{N_j\}$. I.e., relation E could be used with the adornment α iff all the needed bound attributes are obtained from relations that are "allowed" to join with E.

- *Step 5. Repeat Step 4 until no more nodes could be marked.*

- *Step 6. For every join node $N = (j_{E,E'}, [\underbrace{bb...b}_{l}])$ in $\mathcal{N}$ for which $\forall i \in \{1, \ldots, l\}$, the type-relation node N_i[13] is marked and $const(N_i) \ni j_{E,E'}$ do: (1) $Marked = Marked \cup \{N\}$.*

- *Step 7. If there exists $E(\bar{X})$ in the body of r such that there is no node corresponding to E in the set $Marked$ then $RelevantSubspace(r) = (\emptyset, \emptyset)$.*

- *Step 8. Let's define $\mathcal{G}_{Marked}$ the subgraph corresponding to the marked set. $\mathcal{G}_r = (\mathcal{N}_r, \mathcal{D}_r)$ where*
  $\mathcal{N}_r = \{N \mid N \in \mathcal{N} \cap Marked$ and there exists E in the body of r such that N is in the same connected component of the subgraph $\mathcal{G}_{Marked}$ with a node corresponding to $E\}$
  $\mathcal{D}_r = \{e' \mid \exists e = (E_j, [bf]) \stackrel{j, C_E(j)}{\longrightarrow} (E, \tilde{\beta}) \in \mathcal{D}$ and $(E_j, [bf])$, $(E, \tilde{\beta}) \in \mathcal{N}_r$, $C_E(j) \cap const((E_j, [bf])) \neq \emptyset$,
  $e' = (E_j, [bf]) \stackrel{j, C_E(j) \cap const((E_j, [bf]))}{\longrightarrow} (E, \tilde{\beta})\} \cup$
  $\{e' \mid \exists e = (E, \tilde{\beta}) \stackrel{1, C_E(j)}{\longrightarrow} (E_j, [bf]) \in \mathcal{D}$ and $(E_j, [bf])$, $(E, \tilde{\beta}) \in \mathcal{N}_r$, $C_E(j) \cap const((E_j, [bf])) \neq \emptyset$,
  $e' = (E, \tilde{\beta}) \stackrel{1, C_E(j) \cap const((E_j, [bf]))}{\longrightarrow} (E, \tilde{\beta})\} \cup$
  $\{e \mid e = (E_i, [bf]) \stackrel{i, j_{E,E'}}{\longrightarrow} (j_{E,E'}, [b \ldots b]) \in \mathcal{D}$ such that $(j_{E,E'}, [b \ldots b]) \in \mathcal{N}_r\}$.
  $\mathcal{N}_r$ is a subgraph of $\mathcal{N}$ corresponding to the set of connected components determined by the relations used in r in the subgraph of the marked nodes. The sets of join constraints associated with edges are restricted to computed $const()$ sets.

- *Step 9. If $\mathcal{N}_r$ is an unconnected graph then $RelevantSubspace_r = (\emptyset, \emptyset)$. Else if $\mathcal{N}_r$ is a connected graph then $RelevantSubspace_r = (\mathcal{R}_r, \mathcal{JC}_r)$ where*
  $\mathcal{R}_r = \{(E, \alpha) \mid E$ is a nontype-relation such that $\exists N = (E, \alpha) \in \mathcal{N}_r\}$ and
  $\mathcal{JC}_r = \{j_{E,E'} \mid j_{E,E'}$ is a connected join constraint and $\exists \alpha, \beta$ such that $(E, \alpha) \in \mathcal{N}_r$ and $(E', \beta) \in \mathcal{N}_r$ or $j_{E,E'}$ is un unconnected join constraint such that $N = (j_{E,E'}, [b \ldots b]) \in \mathcal{N}_r\}$.

The above definition of the relevant subspace of a query gives a polynomial algorithm for recursively constructing a directed graph that will be used to generate connected rules for a given query. For a query and a capability graph corresponding to the current state of the information space $\mathcal{E}$, the query capability graph is defined by removing some relation-nodes if they never could be used to generate some data, because the input arguments needed could not be obtained from any other relations nor are they bound in the query.

[13] The edge $N_i \stackrel{i, j_{E,E'}}{\longrightarrow} (j_{E,E'}, [\underbrace{bb...b}_{l}])$ exists in $\mathcal{G}$.

We now give a theorem that establishes the connection between a ***connected rule*** (Def. 12) for a query and its capability graph (Def. 18). Intuitively, a connected rule corresponding to a query r could be represented by a connected graph that must be a subgraph of the query capability graph. If the later is not connected then there exists no subgraph of it containing all the relations from the query rule that is connected.

Theorem 4 ***Necessary Condition for Existence of a Connected Query Rule.*** *If the Query Capability Graph $\mathcal{G}_r$ for a query r is disconnected there exists no connected rule for the query, i.e., $\bar{r}_{/\mathcal{E}} = \emptyset$.*

Proof: Let r be:

$$r : Q(\bar{V}) : -E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0). \tag{15}$$

its query capability graph $\mathcal{G}_r$, and a connected rule r^c corresponding to it:

$$r^c : \quad Q(\bar{V}) : -E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0), R_1(\bar{Y}_1), R_2(\bar{Y}_2), \ldots, R_k(\bar{Y}_k) \tag{16}$$

r^c has a feasible order by property **Feasible** (Def. 12) in which the joins are executed. This means that each nontype-relation E in the body of r^c has an adornment α and for each attribute i a set of join constraints $C_{E/r^c}(i)$ that are used in the connected query rule r^c. Let's define the subgraph of $\mathcal{G}_r$ corresponding to r^c, denoted by $\mathcal{G}_{r^c}$. Then $\mathcal{G}_{r^c}$ has all the nodes (E, α) where E is a relation in r^c and α its adornment in a feasible order of r^c (for type-relation α is $[bf]$). It has edges with the edge labels restricted to $C_{E/r^c}(i)$ for each attribute i of E. By property **Minimum Rewriting** all relations from the body of the query r have a node in $\mathcal{G}_{r^c}$. According to the property **Connected**, $\mathcal{G}_{r^c}$ is connected. Thus, $\mathcal{G}_{r^c}$ is a connected subgraph of $\mathcal{G}_r$ that contains all the relations from the body of the query r. But $\mathcal{G}_r$ has been constructed to have only connected components corresponding to the relations from the body of the query r. Then existence of r^c proves that there is only one such connected component and thus $\mathcal{G}_r$ must be connected.

Example 12 *Let's consider in our example the query requesting the on-line collection addresses that have topics "greenhouse effect" and their authors. The query could be expressed in our language as follows:*

$$r : \quad Q(U, A) : -\textbf{Address}(U), \textbf{AuthorName}(A), \textbf{ThesaurusTerm}(\textit{'greenhouse effect'}) \tag{17}$$

The computation of $RelevantSubspace_r$ is shown below. The table summarizes the $const()$ sets for each step of the algorithm given in Definition 18 and the set $Marked$. The capability graph used is shown in Figure 2.

	ThesaurusTerm	*BsoTerm*	*Address*	*Publication*	*GeneralName*	*AuthorName*	*Date*	$Marked$
Step 1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
Step 2	r							*ThesaurusTerm*
Step 3								*Ieee_Author*
						j4, j5		*AuthorName*
							j5	*Date*
Step 4								*Conspectus(ffb)*
	j6,j5							
			j5					*Address*
				j5				*Publication*
					j4			*GeneralName*
	j1							*Nasa_Thesaurus*
Step 4								*Name_Authority*

Following the steps from Definition 18 we have that $\mathcal{R}_r = \{$(**Conspectus**, $[ffb]$), (**Name_Authority**, $[bf]$), (**Ieee_Author**, $[ff]$), (**Nasa_Thesaurus**, $[bf]$) $\}$ *and* $\mathcal{JC}_r = \{j1, j4, j5, j6\}$.

7 Dynamic Query Planning Process

7.1 Generation of the Library of Parameterized Action Schemas

Given a user query, the first stage in the query process is to compute the relevant subspace of the query, as defined in Section 6. Then we transform this DIIM subspace into a format processed by the planning algorithm (this format is referred to as parameterized action schemas). Thus, the Parameterized Actions (PAs) are generated only for the information sources that are relevant to a given user query, i.e., they could be used in the refine process. The planning algorithm generates query plans corresponding to the set of connected rules of a user query using only the information sources found in the system at the query time reflected by the fact that the library of actions is obtained only from the relevant subspace of the query. The generated query plans must not only define what external sites are to be used to answer the query, but they also must define a partial-order among the steps to be executed in the process of query execution. For the process of defining the consistent query plans in the DIIM system we thus use a regression planning algorithm based on the conditional partial-order planning algorithm (CPOP) defined in [RN95].

We now discuss our solution for transforming our given information source descriptions kept in the *capability knowledge base* (see Figure 1) expressed in DIIM model into a set of parameterized actions that can be used by the planning agent (see [RN95], [LS93])[14]. We start by making the observation that the act of obtaining

[14] We use a STRIPS-like language to describe parameterized actions.

data from an information source corresponds to a *sensing action* as defined in [RN95], i.e., we cannot know at the planning time what will be the output at the execution time. For every relation and for every one of its query templates, we define a new parameterized action (PA) which, at the execution time, corresponds to querying the relation with an instantiation of that query template.

We propose the following 2-step algorithm for automatically generating the parameterized action library given the restriction of the information space to the relevant subspace of the query:

Algorithm: Parameterized Action Set Generation

INPUT: $RelevantSubspace_r = (\mathcal{R}_r, \mathcal{JC}_r)$ the relevant subspace of the query r.

OUTPUT: *actions* the library of parameterized actions

Step 1: For every $(E, \alpha) \in \mathcal{R}_r$, E an n-ary relation and the adornment $\alpha \in \tilde{A}^E$ in its minimal adornment set, we define a *parameterized action* (PA) as following: Let $B(\alpha) = \{b_1, b_2, \ldots, b_k \mid \alpha[b_j] = b, 1 \leq j \leq k\}$ and $F(\alpha) = \{f_1, f_2, \ldots, f_{n-k} \mid \alpha[f_j] = f, 1 \leq j \leq n-k\}$ correspond to the positions where α has the value b and f, respectively. Let $E(X_1, X_2, \ldots, X_n) \subseteq E_1(X_1), \ldots, E_n(X_n)$ be an integrity constraint of the form (2) and $E(X_1, X_2, \ldots, X_n) \subseteq C(X_1, X_2, \ldots, X_n)$ be an integrity constraint of the form (1), with $C(X_1, X_2, \ldots, X_n)$ a conjunction of built-in predicates $X_l \theta_l Y_j$ with $X_l, Y_j \in \{X_1, X_2, \ldots, X_n\}$ for $l \in \{l_1, \ldots, l_h\} \subseteq \{1, \ldots, n\}$ and $j \in \{j_1, \ldots, j_h\} \subseteq \{1, \ldots, n\}$ and built-in predicates $X_m \vartheta_m Y_m$ with $X_m \in \{X_1, X_2, \ldots, X_n\}$ and Y_m a constant for $m \in \{m_1, \ldots, m_p\} \subseteq \{1, \ldots, n\}$ and $\theta_l, \vartheta_m \in \{<, \leq, =, \geq, >\}$. The *parameterized action* corresponding to (E, α) is defined by Equation (18). The precondition $(defined\ ?t\ E_i\ ?R)$ assures the fact that when the action will be executed $?t$ is bound to an object of type E_i after querying relation $?R$. The predicates $(\theta\ ?x\ ?y)$ will trigger search control rules at the planning time that will add these constraints to the plan constraints (see Step 5.7. in the algorithm *DynamicQueryPlanning* in Section 7.2 and [BCea95]) which must be satisfied at the execution time.

$$
\begin{array}{lll}
Op(& \text{ACTION:} & E^{\alpha} \\
 & \text{PARAMS:} & (?t_1, \ldots, ?t_n) \\
 & \text{PRECOND:} & (:and \\
 & & (:exists\ (relation\ ?R1)\ (defined\ ?t_{b_1}\ E_{b_1}\ ?R1)) \\
 & & \ldots \\
 & & (:exists\ (relation\ ?Rk)\ (defined\ ?t_{b_k}\ E_{b_k}\ ?Rk)) \\
 & & (new-object\ ?t_{f_1}) \ldots (new-object\ ?t_{f_{n-k}})\ ) \\
 & \text{EFFECT:} & (:and\ (relation\ E) \\
 & & (variable\ ?t_{f_1}) \ldots (variable\ ?t_{f_{n-k}}) \\
 & & (defined\ ?t_{f_1}\ E_{f_1}\ E) \ldots (defined\ ?t_{f_{n-k}}\ E_{f_{n-k}}\ E) \\
 & & (defined\ ?t_{b_1}\ E_{b_1}\ E) \ldots (defined\ ?t_{b_k}\ E_{b_k}\ E) \\
 & & (\theta_1\ ?t_{l_1}\ ?t_{j_1}) \ldots (\theta_h\ ?t_{l_h}\ ?t_{j_h})\ (\vartheta_1\ ?t_{m_1}\ Y_{m_1}) \ldots (\vartheta_p\ ?t_{m_p}\ Y_{m_p})\ )\ )
\end{array}
\tag{18}
$$

Example 13 *For the* **Conspectus** *relation, we have the adornment set* $\tilde{A}^{\textbf{Conspectus}} = \{[bff], [ffb]\}$. *For the adornment* $[ffb]$ *and using the integrity constraint (ic5), we obtain the parameterized action (19) by the first step of the Algorithm* **Parameterized Action Set Generation** *from Section 7.1.*

$$
\begin{array}{lll}
Op(& \textit{ACTION:} & \textbf{Conspectus}^{[ffb]} \\
 & \textit{PARAMS:} & (?J\ ?S\ ?A) \\
 & \textit{PRECOND:} & (:\ and \\
 & & (:\ exists\ (relation\ ?R2)\ (defined\ ?S\ \textbf{ThesaurusTerm}\ ?R2)) \\
 & & (new-object\ ?A)\ (new-object\ ?J)\ ) \\
 & \textit{EFFECT:} & (:\ and\ (relation\ \textbf{Conspectus}) \\
 & & (variable\ ?A)\ \ (variable\ ?J) \\
 & & (defined\ ?A\ \textbf{Address Conspectus}) \\
 & & (defined\ ?J\ \textbf{Publication Conspectus})\ )\ )
\end{array}
\tag{19}
$$

Step 2: For a given join constraint $j_{E,E'} \in \mathcal{JC}_r$ of the form (3) $E(\bar{X}), \phi(\bar{X}_0) \equiv_J E'(\bar{Y}), \phi'(\bar{Y}_0)$ we define a parameterized action as follows: Let $\bar{X} = (X_1, \ldots, X_n) \subseteq \bar{X}_0 = (X_1, \ldots, X_{n_0})$ and $\bar{Y} = (Y_1, \ldots, Y_m) \subseteq \bar{Y}_0 = (Y_1, \ldots, Y_{m_0})$. $\phi(\bar{X}_0)$ and $\phi'(\bar{Y}_0)$ are conjunctive expressions with built-in predicates and binary relations (corresponding to complex subobjects). For each built-in predicate $X\theta Y$ in $\phi(\bar{X}_0)$, $\phi'(\bar{Y}_0)$, X and Y variables in $\bar{X}_0$ and $\bar{Y}_0$, we construct a precondition $(\theta\ ?x\ ?y)$ ($?x, ?y$ are the parameters corresponding to variables X and Y, respectively). If Y is a constant, the precondition is $(\theta\ ?x\ Y)$. The preconditions corresponding to built-in predicates are implemented as *facts* that will add these constraints to the plan constraints (see Step 4 in the algorithm *DynamicQueryPlanning*). Binary relations $e(X, Y)$ in $\phi(\bar{X}_0)$, $\phi'(\bar{Y}_0)$ have corresponding preconditions $(e\ ?x\ ?y)$ that are implemented as *facts* and will be added to the plan constraints at the planning time.

$$
\begin{array}{lll}
Op(& \text{ACTION:} & j_{E,E'} \\
 & \text{PARAMS:} & (?x_1, \ldots, ?x_{n_0}, ?y_1, \ldots ?y_{m_0}) \\
 & \text{PRECOND:} & (:\ and\ (relation\ E)\ (relation\ E') \\
 & & \quad (defined\ ?x_1\ E_1\ E) \ldots (defined\ ?x_{n_0}\ E_{n_0}\ E) \\
 & & \quad (defined\ ?y_1\ E'_1\ E') \ldots (defined\ ?y_{m_0}\ E'_{m_0}\ E') \\
 & & (\theta_1\ ?z_1\ ?v_1) \ldots (\theta_s\ ?z_s\ ?v_s) \\
 & & (\vartheta_1\ ?t_1\ ?w_1) \ldots (\vartheta_p\ ?t_p\ ?w_p) \\
 & & (e_1\ ?s_1\ ?t_1) \ldots (e_l\ ?s_l\ ?t_l)\ \ ) \\
 & \text{EFFECT:} & (:\ and\ (join\ ?x_1\ E_1\ j_{E,E'}) \ldots (join\ ?x_{n_0}\ E_{n_0}\ j_{E,E'}) \\
 & & \quad (join\ ?y_1\ E'_1\ j_{E,E'}) \ldots (join\ ?y_{m_0}\ E'_{m_0}\ j_{E,E'})\ )\ )
\end{array}
\tag{20}
$$

Example 14 *We present here the parameterized action corresponding to* $(j5)$ *in Example 5.*

$$
\begin{array}{lll}
Op(& \textit{ACTION:} & j5_{\textbf{Conspectus,leee\_Author}} \\
 & \textit{PARAMS:} & (?ADD\ ?J\ ?S\ ?N\ ?A\ ?D) \\
 & \textit{PRECOND:} & (:and\ (relation\ \textbf{Conspectus})\ (relation\ \textbf{leee\_Author}) \\
 & & (defined\ ?ADD\ \textbf{Address Conspectus}) \\
 & & (defined\ ?J\ \textbf{Publication Conspectus}) \\
 & & (defined\ ?S\ \textbf{ThesaurusTerm Conspectus}) \\
 & & (defined\ ?A\ \textbf{AuthorName leee\_Author}) \\
 & & (defined\ ?D\ \textbf{Date leee\_Author}) \\
 & & (publicationName\ ?J\ ?N)\ (=\ ?N\ 'ieee')\ ) \\
 & \textit{EFFECT:} & (:and\ (join\ ?ADD\ \textbf{Address}\ j5_{\textbf{Conspectus,leee\_Author}}) \\
 & & (join\ ?J\ \textbf{Publication}\ j5_{\textbf{Conspectus,}\textit{leee\_Author}}) \\
 & & (join\ ?S\ \textbf{ThesaurusTerm}\ j5_{\textbf{Conspectus,leee\_Author}}) \\
 & & (join\ ?A\ \textbf{AuthorName}\ j5_{\textbf{Conspectus,leee\_Author}}) \\
 & & (join\ ?D\ \textbf{Date}\ j5_{\textbf{Conspectus,leee\_Author}})\ ) \\
)
\end{array}
\tag{21}
$$

As we discussed before, the join constraint explicitly defines some conditions to be met so the two relations could be joined. We model this semantics by imposing at the execution time that before the join is done the constraints should be met and that we know the tuples corresponding to the two relations could be joined.

7.2 The Conditional Partial-Order Planning Algorithm for Query Planning

In this section we describe the planning algorithm we use to generate query plans for a given user query and a system configuration reflected in the parameterized action set.

Given the set of parameterized actions defined for our problem domain, we now propose a conditional partial-order planning algorithm (see Figure 3). The algorithm is a regression planning algorithm ([RN95]) that incrementally finds possible plans (corresponding to refined queries) for the given query. We have made several modifications to CPOP defined in [RN95] as required for our problem at hand. First, our algorithm performs a breadth-first search with the goal of finding all the plans ordered by the number of total steps[15]. Our algorithm will find only the plans that represent connected rules pruning out plans that are inconsistent or incomplete using the function SOLUTION(*plan*) (Step 2 in the algorithm) that checks for this condition. In the following subsections, we discuss how the function SOLUTION() is defined by expressing the connected rule properties in terms of partial-order plan characteristics.

Algorithm *MakeInitialPlan*

Input: A query rule in $\mathcal{E}$-*normal form* $r : Q(\bar{V}) :- R_1(\bar{X}_1), R_2(\bar{X}_2), \ldots, R_n(\bar{X}_n), C(\bar{X}_0)$
Output: The initial plan *plan* for *DynamicQueryPlanning* Algorithm
Step 1.

[15] In the AI literature, typically finding one single plan is considered sufficient, hence depth-first search is utilized [RN95]. For future work we plan to have more efficient ways of finding the set of query plans based on a real cost reflecting the information source transaction costs. If we assume that a query has a maximum total cost, the planning algorithm strategy would be a fixed-depth best-first search with the fixed-depth determined by the total cost constraint.

$plan = ($ STEPS: $\{S_1 : Op($ ACTION : START
PRECOND: $\emptyset$
EFFECT: $\emptyset)$
$S_2 : Op($ ACTION: FINISH
PRECOND: $\emptyset$
EFFECT: $\emptyset)\}$
ORDERINGS: $\{S_1 \prec S_2\}$
BINDINGS: $\emptyset$
CONSTRAINTS: $\emptyset$
LINKS: $\emptyset$
)

Step 2. **for all** built-in subgoals $X = c$ in $C(\bar{X}_0)$ such that c
is a constant, X is a variable and $E(X)$ its type in r **do**
S_1.EFFECT := S_1.EFFECT $\cup\{(defined\ X\ E\ query)\}$
plan.BINDINGS := *plan*.BINDINGS $\cup\{X = c\}$
end

Step 3. **for all** built-in subgoals $X_i = X_j$ in $C(\bar{X}_0)$, such that X_i and X_j are variables **do**
plan.BINDINGS := *plan*.BINDINGS $\cup\{X_i = X_j\}$
end

Step 4. **for all** variables X with $E(X)$ its type, in *plan*.BINDINGS indirectly bound to a constant **do**
S_1.EFFECT := S_1.EFFECT $\cup\{(defined\ X\ E\ query)\}$
end

Step 5. **for all** $Y \in \bar{V} \cup \bar{X}_0 \cup \ldots \cup \bar{X}_n$ with the type $E(Y)$
such that $\nexists R(\bar{X}), R'(\bar{Y})$ non-type relations in the body of the rule r
such that $Y \in \bar{X} \cap \bar{Y}$[16] **do**
S_2.PRECOND := S_2.PRECOND $\cup\{(:exists\ (?j)\ (join\ Y\ E\ ?j))\}$
end

Step 6. all new variables must be part of a join
S_2.PRECOND := S_2.PRECOND $\cup$ $\{(:forall\ (variable\ ?v)$
$(:exists\ (?j)\ (:exists\ (?E)\ (join\ ?v\ ?E\ ?j))))\}$
end

Step 7. **for all** non-type relations $R(\bar{X})$ in the body of the query rule r **do**
S_2.PRECOND := S_2.PRECOND $\cup\{(relation\ R))\}$
for all variables $X \in \bar{X}$ with $E(X)$ its type **do**
S_2.PRECOND := S_2.PRECOND $\cup\{(defined\ X\ E\ R))\}$
end
end

Step 8. **for all** built-in subgoals $Z\theta W$ with $\theta \in \{<, \leq, \geq, >\}$ in $C(\bar{X}_0)$ **do**
plan.CONSTRAINTS := *plan*.CONSTRAINTS $\cup\{Z\theta W\}$

7.3 Partial-Ordered Plan Description

A partial-order query plan obtained by the planning algorithm (Figure 3) consists of the following components: (1) A set $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ of plan steps which in our domain are parameterized actions. A parameterized action corresponds to an information source query that is an instantiation of a particular query template. (2) A set $\mathcal{O}$ of ordering constraints. $S_i \prec S_j \in \mathcal{O}$ means that step S_i must occur sometime (not necessarily immediately) before S_j. (3) A set $\mathcal{B}$ of variable binding constraints. A variable constraint is of the form $v = x$ where v is a variable and x is a variable or a constant. (4) A set $\mathcal{C}$ of built-in predicates of the form $Z\theta W$ where $\theta \in \{<, \leq, \geq, >\}$ that must be consistent to the binding constraints. (5) A set $\mathcal{L}$ of causal links. A causal link $S_i \xrightarrow{c} S_j$ means that S_i achieves the precondition c for the step S_j.

Algorithm *DynamicQueryPlanning* $(Q(\bar{V}) : -C(\bar{X}_0), R_1(\bar{X}_1), R_2(\bar{X}_2), \ldots, R_n(\bar{X}_n)$, *actions*, *plans*)
/* Generate consistent plans for the query $Q(\bar{V})$; *actions* is the library of parameterized actions generated from the relevant subspace of the query Q*/

- $plans = \emptyset$
- $MakeInitialPlan(Q(\bar{V}) : -C(\bar{X}_0), R_1(\bar{X}_1), R_2(\bar{X}_2), \ldots, R_n(\bar{X}_n), plan\ )$
- $q_{plans} := [plan]$
- **for all** $plan \in q_{plans}$ **do**:

  1. q_{plans} := REST$[q_{plans}]$ /* we discard *plan* from the queue of plans*/
  2. **if** SOLUTION(*plan*)
     $plans = plans \cup \{plan\}$
     continue
  3. choose a subgoal to be considered next S_{need} with an open condition c
  4. **if** c is a built-in predicate **then**
     if c is consistent with *plan*.BINDINGS and *plan*.CONSTRAINTS
     then
     plan.CONSTRAINTS = *plan*.CONSTRAINTS $\cup\{c\}$, q_{plans} = APPEND(*plan*, q_{plans})
     continue
  5. **for all** S_{add} from *actions* or *plan*.STEPS that has c_{add} as effect such that u = UNIFY(c, c_{add}, *plan*.BINDINGS), construct a new plan $plan_{S_{add}}$ as follows:
     1 $plan_{S_{add}} = plan$ /* we make a copy of the *plan* first */
     2 $plan_{S_{add}}$.BINDINGS = $plan_{S_{add}}$.BINDINGS $\cup\{u\}$
     3 $plan_{S_{add}}$.LINKS = $plan_{S_{add}}$.LINKS $\cup\{S_{add} \xrightarrow{c} S_{need}\}$
     4 $plan_{S_{add}}$.ORDERINGS = $plan_{S_{add}}$.ORDERINGS $\cup\{S_{add} \prec S_{need}\}$
     5 If S_{add} is a new parameterized action from *actions* then $plan_{S_{add}}$.ORDERINGS = $plan_{S_{add}}$.ORDERINGS $\cup\{\ S_1 \prec S_{add}, S_{add} \prec$ FINISH $\}$
     $plan_{S_{add}}$.STEPS = $plan_{S_{add}}$.STEPS $\cup\{S_{add}\}$
     6 Check if the plan is consistent, i.e., there is no contradiction in the ordering or binding constraints. If some inconsistency then **continue**
     7 Add to $plan_{S_{add}}$.CONSTRAINTS all built-in and binary relations predicates from the effect set of the step S_{add}
     8 Check all the constraints from $plan_{S_{add}}$.CONSTRAINTS against the new $plan_{S_{add}}$.BINDINGS set. If some inconsistency then **continue**.
  6. q_{plans} = APPEND($plan_{S_{add}}$, q_{plans}) and **continue**

Figure 3: The Conditional Partial-Order Planning Algorithm in DIIM.

The initial plan consists of the two special steps START and FINISH and the sets $\mathcal{S}, \mathcal{O}, \mathcal{B}, \mathcal{C}, \mathcal{L}$ defined using the algorithm *MakeInitialPlan*. The planning algorithm obtains only *complete* and *consistent* plans in the sense defined in [RN95]. A *complete* plan is one in which every precondition of every step is achieved by some other step, and no other step can possibly cancel out the condition. More formally, a step S_i achieves a precondition c of the step S_j if (1) $S_i \prec S_j$ and $c \in S_j$.EFFECT and (2) there is no step S_k such that $\neg c \in S_k$.EFFECT, where $S_i \prec S_k \prec S_j$ in some linearization of the plan. A *consistent* plan is one in which there are no contradictions in the ordering or binding constraints. A contradiction occurs when both $S_i \prec S_j$ and $S_j \prec S_i$ hold or both $v = x$ and $v = y$ hold with x and y different constants. We have extended the basic CPOP algorithm as described in [RN95] to handle the built-in predicate set $\mathcal{C}$ that permits query language and information source descriptions to include the conjunctions of built-in predicates. Step 4 of the algorithm verifies that the set of bindings and the set of built-in predicates are consistent and discards the plans that do not have this property.

7.4 Connected Rule Representation

The presented algorithm finds only plans that represent connected rules. We give here the necessary and sufficient conditions for a plan generated by the algorithm presented in Figure 3 to be a representation of a connected rule for a query. This condition is used by the SOLUTION() function (Step 2 in Figure 3) to check if a consistent and complete plan is a representation of a connected rule for that query.

Definition 19 ***Common-Bound Partial-Order Set (CB-set)*** *We say that a partial-order set $(P, <)$ is a Common-Bound (CB) set if and only is for every two elements $p_1, p_2 \in P$, there exists $p \in P$ so that either (1) $p < p_1$ and $p < p_2$ or (2) $p_1 < p$ and $p_2 < p$*[17].

Theorem 5 *A plan generated by the Algorithm DynamicQueryPlanning is a representation of a connected rule for the input query if and only if (1) $\forall c \in S_1.EFFECT$, $\exists S$ so that $S_1 \xrightarrow{c} S$ and (2) the set plan.$\mathcal{O}_{\mathcal{L}} = \{S_i \prec S_j \mid \exists S_i \xrightarrow{c} S_j \in plan.\mathcal{L}\}$ is a CB-set.*

Proof: / Property (1) is equivalent to property **Restriction** (see Def. 12) imposing that all the inputs are used in the query. Because the plan is complete and consistent ([RN95]), it is assured that all the variables will be obtained from some join constraints (as asked for in the precondition set of the FINISH step) which is equivalent to the properties **Maximal** and **Feasible** from Definition 12. Property (2) from above imposes that a plan corresponds to a rule that is connected according to property **Connected**.

Example 15 *Consider the query "Retrieve the addresses and related authors for on-line collections that have the topics 'greenhouse effect' published in an IEEE journal", is expressed by:*

$$Q(U, A) : -\textbf{Address}(U), \textbf{BsoTerm}(\textit{'greenhouse effect'}), \textbf{AuthorName}(A), \textbf{Publication}(J), \textbf{publicationName}(J, \textit{'ieee'}) \quad (22)$$

Figure 4 shows a graphical representation of two plans generated by the planning algorithm for query 22. The dashed lines represent the outputs for the parameterized actions. The connected rules r_1 and r_2 are the rules corresponding to plan 1 and 2, respectively.

In the Algorithm *GenerateOrdering* we give the algorithm for obtaining a feasible ordering for a connected rule by constructing the solution graph $\bar{\mathcal{G}}_r$. The algorithm starts with (type or non-type) relations that don't need any binding, i.e., the relations having $[ff...f]$ among their adornment sets or the type relation E for which exists a bound variable X such that $E(X)$ is in the body of r. At any step the algorithm finds new bound variables that could be obtained from the relations that need as input already bound variables until

[17] If $p_1 < p_2$, then p is one of the two.

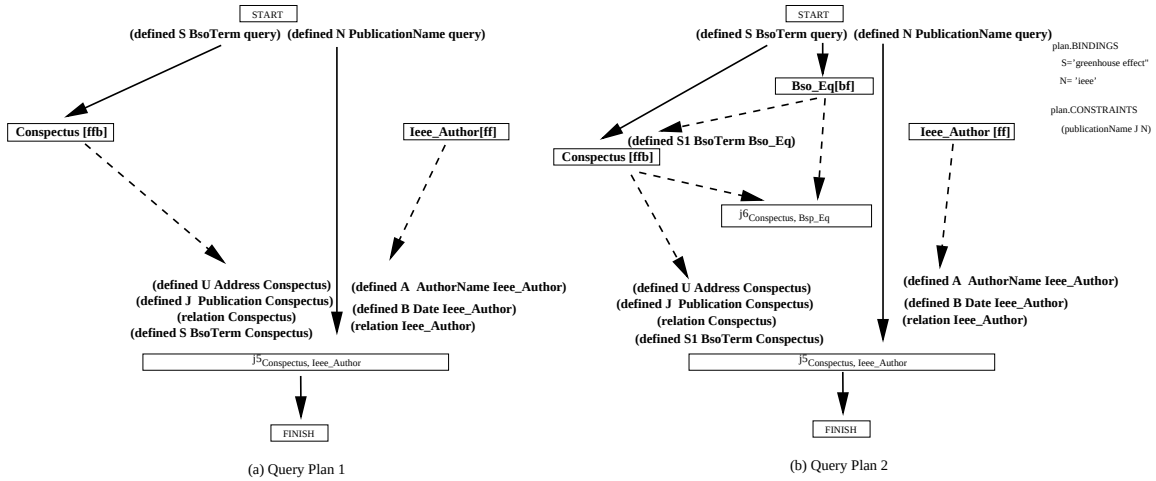

Figure 4: Two query plans for query (22) and their corresponding rules:
(a) $r1 : Q(U, A) : -S =$ *"greenhouse effect"*, **BsoTerm**(S), **Conspectus**(U, J, S), **Address**(U), **Publication**(J), **publicationName**(J, N), $N =' ieee'$, **AuthorName**(A), *Ieee_Authors*(A, B)
(b) $r2 : Q(U, A)$:- $S =$ *"greenhouse effect"*, **BsoTerm**(S), **Bso_Eq**$(S, S1)$, **Conspectus**$(U, J, S1)$, **Address**(U), **Publication**(J), **publicationName**(J, N), $N =$ *"ieee"*, **AuthorName**(A), *Ieee_Authors*(A, B)

no more variables could be bound. If all relevant variables are bound and all relations from the body of the rule r have attached a permissible binding pattern the algorithm ends successfully. As a last step, the join constraints used in the connected rule are added to the solution graph.

Using the Algorithm *GenerateOrdering*, we can prove that the connected relation of a loosely-specified query r (as defined in Definition 14) is equivalent to the relation defined by the set of rules corresponding to the plans obtained by the Algorithm *DynamicQueryPlanning*.

Theorem 6 *Let r be a query rule with $\bar{r}_{/\mathcal{E}}$ not empty. Let r_{plans} be the set of plans obtained by the Algorithm DynamicQueryPlanning and $\bar{r}_{plans}$ the set of rules corresponding one by one to the set r_{plans}. Then $\bar{r}_{/\mathcal{E}} \equiv \bar{r}_{plans}$.*

Algorithm *GenerateOrdering* $(r : Q(\bar{V}) := E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0)\ )$
/* Given r a connected query rule in $\mathcal{E}$*-normal form* and the adornment sets for all relations contained in the body of the rule, the algorithm finds a subgoal ordering for r */

Note that a rule r has no feasible ordering if there exists a nonrelevant variable (i.e., it appears only once in the body of the rule) such that X appears as i'th argument in a relation E and E has no adornment α such that $\alpha[i] = f$. It is clear that in any feasible ordering for r, E must have an adornment with f on i'th position. Thus for all E containing nonrelevant variables, we restrict their adornment sets to this type of adornments.

Construct a solution graph $\bar{\mathcal{G}}_r = (\bar{\mathcal{N}}_r, \bar{\mathcal{E}}_r)$ as following:

There are two types of nodes in $\bar{\mathcal{G}}_r$: (1) complete nodes of the form $(E, \bar{X}, \tilde{\alpha}, k)$ where $E(\bar{X})$ is an expression from the body of the rule r, $\tilde{\alpha} \in \tilde{A}^E$ is the adornment assigned to E and k is the sequence number for E, corresponding to the fact that all the needed bound arguments for $E(\bar{X})$ depend on the relations having the sequence numbers smaller than k; at any stage of the algorithm, there is at most one complete node for an expression $E(\bar{X})$; (2) an incomplete node $(E, \bar{X}, \tilde{\alpha}, \_)$ with $E(\bar{X})$ an expression from the body of the rule r, $\tilde{\alpha} \in \tilde{A}^E$, specifies that there is no complete node corresponding to the expression $E(\bar{X})$ and that not all the needed bound arguments for the expression $E(\bar{X})$ with the adornment $\tilde{\alpha}$ could be obtained from the existing complete nodes; at any stage of the algorithm there could be more than one incomplete node corresponding to an expression $E(\bar{X})$.

Initialize the graph

(1) $START_{nontype} = \{\ (E, \bar{X}, \tilde{\alpha}, 0) \mid E(\bar{X})$ is a non-type relation in the body of r and $\tilde{\alpha} \in \tilde{A}^E$ such that $\tilde{\alpha}$ has only f's $\}$

(2) $START_{type} = \{(E, V, [bf], 1) \mid E(V)$ is a type relation in the body of r and V is a bound variable in $C_{X_0}\ \}$

(3) $START_{var} = \{(V, V, [f], 0) \mid V$ is a bound variable in $C(\bar{X}_0)\ \}$

(4) $\bar{\mathcal{N}}_r = START_{nontype} \cup START_{type} \cup START_{var}$

(5) $\bar{\mathcal{E}}_r = \{(V, -, [f], 0) \xrightarrow{1,1} (E, V, [bf], 1) \mid (V, -, [f], 0) \in START_{var}, (E, V, [bf], 1) \in START_{type}\}$

(6) **while** $\exists (E, \bar{X}, \tilde{\alpha}, k)$ a complete node unmarked in $\bar{\mathcal{N}}_r$ **do**

Expend a unmarked complete node

(7) choose $(E, \bar{X}, \tilde{\alpha}, k) \in \bar{\mathcal{N}}_r$ and mark it

(8) **if** E is a type relation (i.e., $(E, \bar{X}, \tilde{\alpha}, k)$ must be of the form $(E, X_j, [bf], k)$) **then**

(9) $\bar{\mathcal{E}}_r = \bar{\mathcal{E}}_r \cup \{\ (E, X_j, [bf], k) \xrightarrow{2,j} (R, \bar{X}, \tilde{\beta}, \_) \mid \nexists$ a complete node for $R(\bar{X})$ in $\bar{\mathcal{N}}_r$, $R(\bar{X})$ is a non-type relation in the body of the rule r such that the j'th argument is of type E, $\tilde{\beta} \in \tilde{A}^R, \tilde{\beta}[j] = b$ and $\bar{X}[j] = X_j\ \}$

$\bar{\mathcal{N}}_r = \bar{\mathcal{N}}_r \cup \{(R, \bar{X}, \tilde{\beta}, \_) \mid$ for all new incomplete nodes added above$\}$

(10) **else if** E is a non-type relation **then**

(11) $\bar{\mathcal{E}}_r = \bar{\mathcal{E}}_r \cup \{(E, \bar{X}, \tilde{\alpha}, k) \xrightarrow{i,1} (R, X_i, [bf], \_) \mid \nexists$ a complete node for $R(X_i)$ in $\bar{\mathcal{N}}_r$, $R(X_i)$ is a type relation in the body of r such that $\exists i$ where the i'th argument of the relation E is of type R and $\tilde{\alpha}[i] = f, \bar{X}[i] = X_i\}$

$\bar{\mathcal{N}}_r = \bar{\mathcal{N}}_r \cup \{(R, \bar{X}_i, [bf], \_) \mid$ for all new incomplete nodes added above$\}$

(12) **end**

Add new complete nodes

(13) **for all** incomplete nodes $(R, \bar{X}, \tilde{\alpha}, \_)$ in $\bar{\mathcal{N}}_r$ such that for all i with $\tilde{\alpha}[i] = b$ $\exists (E, \bar{Y}, \tilde{\beta}, l) \xrightarrow{j,i} (R, \bar{X}, \tilde{\alpha}, \_) \in \bar{\mathcal{E}}_r$ **do**

(14) $\bar{\mathcal{N}}_r = \bar{\mathcal{N}}_r \cup \{(R, \bar{X}, \tilde{\alpha}, k+1)\} \setminus \{(R, \bar{X}, \tilde{\gamma}, \_) \mid \forall \tilde{\gamma}\}$

(15) $\bar{\mathcal{E}}_r = \bar{\mathcal{E}}_r \cup \{(E, \bar{Y}, \tilde{\beta}, l) \xrightarrow{j,i} (R, \bar{X}, \tilde{\alpha}, k+1) \mid (E, \bar{Y}, \tilde{\beta}, l) \xrightarrow{j,i} (R, \bar{X}, \tilde{\alpha}, \_) \in \bar{\mathcal{E}}_r\}$
$\setminus \{(E, \bar{X}, \tilde{\beta}, m) \xrightarrow{j,i} (R, \bar{X}, \tilde{\gamma}, \_) \mid \forall \tilde{\gamma}, i, j$ and $(E, \bar{X}, \tilde{\beta}, m) \in \bar{\mathcal{N}}_r\}$

(16) **end**

(17) **end**

(18) **if** every $E(\bar{X})$ from the body of the rule r is marked in $\bar{\mathcal{N}}_r$, i.e., $\exists (E, \bar{X}, \tilde{\alpha}, k) \in \bar{\mathcal{N}}_r$ a complete node

(19) **then** the solution is given by $\bar{\mathcal{N}}_r$

(20) **else** r doesn't have a feasible solution

(21) **end**

If r is a connected query rule then by property **Connected** from Definition 12 $\exists J_1, J_2, \ldots, J_m$ DIIM join constraints where for any $i, 1 \leq i \leq m$, (J_i) is defined as $E_{J_i,1}(\bar{Y}_1), \phi_{J_i,1}(\bar{Y'}_1) \equiv_J E_{J_i,2}(\bar{Y}_2), \phi_{J_i,2}(\bar{Y'}_2)$ and there exists a unifier τ_i that unifies the join constraint J_i with the body of the rule r, i.e., $\exists k_i, l_i, 1 \leq k_i, l_i \leq n$ such that $\tau_i(E_{J_i,1}(\bar{Y}_1)) = E_{k_i}(\bar{X}_{k_i})$ and $\tau_i(E_{J_i,2}(\bar{Y}_2)) = E_{l_i}(\bar{X}_{l_i})$. The solution graph of r is modified by adding the nodes $(J_i, (\bar{X}_{k_i}, \bar{X}_{l_i}), [bb], s)$ to $\bar{\mathcal{N}}_r$, and the edges $(E_{k_i}, \bar{X}_{k_i}, \tilde{\alpha}, j) \xrightarrow{1,1} (J_i, (\bar{X}_{k_i}, \bar{X}_{l_i}), [bb], s)$ and $(E_{l_i}, \bar{X}_{l_i}, \tilde{\beta}, p) \xrightarrow{1,2} (J_i, (\bar{X}_{k_i}, \bar{X}_{l_i}), [bb], s)$, for all $1 \leq i \leq m$.

Proof

(1) $\bar{r}_{/\mathcal{E}} \sqsubseteq \bar{r}_{plans}$.

We now show that for every connected rule r^c from $\bar{r}_{/\mathcal{E}}$ there exists a plan such that its corresponding rule is in $\bar{r}_{plans}$. Let r^c be defined as follows:

$$r^c : \underbrace{Q(\bar{V}) : -E_1(\bar{X}_1), E_2(\bar{X}_2), \ldots, E_n(\bar{X}_n), C(\bar{X}_0)}_{r}, R_1(\bar{Y}_1), R_2(\bar{Y}_2), \ldots, R_k(\bar{Y}_k) \tag{23}$$

Let $\bar{\mathcal{G}}_{r^c} = (\bar{\mathcal{N}}_{r^c}, \bar{\mathcal{E}}_{r^c})$ be its solution graph constructed by the Algorithm *GenerateOrdering*. Then we construct a plan corresponding to $\bar{\mathcal{G}}_{r^c}$ as follows:

(1) *MakeInitialPlan*(*r*, *plan*)

(2) **for all** nodes $(R, \bar{X}, \tilde{\alpha}, k) \in \bar{\mathcal{N}}_{r^c}$
we construct an instantiated parameterized action $S_{(R,\bar{X},\tilde{\alpha},k)}$
from the parameterized action $R^{\tilde{\alpha}^R}$ by unifying $\bar{X} = (X_1, X_2, \ldots, X_n)$ with $?\bar{t} = (?t_1, ?t_2, \ldots, ?t_n)$.
plan.STEPS = *plan*.STEPS $\cup\{S_{(R,\bar{X},\tilde{\alpha},k)}\}$
plan.CONSTRAINTS = *plan*.CONSTRAINTS $\cup\{g \mid g \in S_{(R,\bar{X},\tilde{\alpha},k)}, g$ is a built-in subgoal in EFFECT or PRECOND$\}$
end

(3) **for all** edges $(E, \bar{X}, \tilde{\beta}, m) \stackrel{j,i}{\rightarrow} (R, \bar{Y}, \tilde{\gamma}, n)$
in $\bar{\mathcal{E}}_{r^c}$ with S_1 and S_2 their corresponding steps added at (2) **do**
plan.LINKS = *plan*.LINKS $\cup\{S_1 \stackrel{(defined\ Y_i\ E_i\ E)}{\longrightarrow} S_2\}$
plan.ORDERINGS = *plan*.ORDERINGS $\cup\{S_1 \prec S_2\}$

(4) **for all** nodes $(J_i, (\bar{X}_{k_i}, \bar{X}_{l_i}), [bb], s) \in \bar{\mathcal{N}}_{r^c}$
we construct an instantiated parameterized action $S_{(J_i,(\bar{X}_{k_i},\bar{X}_{l_i}),[bb],s)}$
from the parameterized action J_i by unifying $(\bar{X}_{k_i}, \bar{X}_{l_i}) = (X_1, X_2, \ldots, X_{n_0}, Y_1, \ldots, Y_{m_0})$ with $?\bar{t} = (?x_1, ?x_2, \ldots, ?x_{n_0}, ?y_1, \ldots, y_{m_0})$.
plan.STEPS = *plan*.STEPS $\cup\{\ S_{(J_i,(\bar{X}_{k_i},\bar{X}_{l_i}),[bb],s)}\ \}$
end

We claim that the plan constructed above is obtained by the Algorithm *DynamicQueryPlanning* less a symbolic mapping.

(2) $\bar{r}_{plans} \sqsubseteq \bar{r}_{/\mathcal{E}}$

Any plan obtained by the Algorithm *DynamicQueryPlanning* has a connected rule representation. This is true because function SOLUTION in the algorithm is checking exactly this property.

8 Related Work

This research is conducted in the general context of a large, interdisciplinary project of constructing a distributed agent-based architecture for UMDL, the University of Michigan Digital Library System [ABD+96]. UMDL assumes a large, decentralized environment of independent agents, including task planners, mediators, and thousands of information sources. The development of a model such as DIIM to describe the content and capabilities of the information sources, and the efficient processing of user queries across this large space using planning and other such techniques, as done in this paper, is one of the key issues that must be tackled as part of this effort.

In the work of Levy et. al [LSK95], a global information system is designed using the world-view approach where the external information sources are described relative with the unified world-view relations. Put differently, a global unified schema as common for tightly integrated multidatabase systems is constructed. Having external sources described relative with the unified world-views is somehow similar with our approach of imposing join and type constraints for external relations. However, in such tightly-integrated model, the definition of the world-view relations and descriptions of the external information sources are subject to "schema evolution" if new sites join the system and they cannot be described relative to the existing world-view relations. The queries in the global information system are expressed in terms of world-view relations, i.e., the global multidatabase schema and external sources are assumed to support SPJ queries, i.e., all sources are relational having the same query capabilities. This approach requires the system users to know the definitions of the world-view relations.

Papakonstantinou et al. [PGMW95, PGMU95] are pursuing a similar goal of information gathering across multiple sources. However, their proposed language OEM assumes queries that explicitly list the source identifiers of the database from which the data is to be taken. Thus loosely-specified queries (e.g., that don't specify from where the data is to be retrieved) are not handled. However, the query reformulation process is similar to our planning algorithm both considering the query templates of the external information sources.

[CKP94] relates to our DIIM approach in as much as our information source descriptions, in particular, the query templates along with the source descriptions, can be regarded as view definitions. Our planning algorithm thus effectively constructs and optimizes queries across these 'views' - as done in [CKP94]. However, our query planning is more sophisticated because we also incorporate join and integrity constraints. In addition, we establish what constitutes a fully-specified refined query and give necessary conditions for query construction.

The notion of using query templates as part of an information source description is based on [Ull89, RSU95]. In DIIM, we utilize these query templates as one of the basic building blocks for constructing the parameterized action set and then to generate 'good' executable plans given our semantic of query refinement, i.e., connected query rule concept.

Our techniques of query planning are based on AI planning techniques [RN95]. However, we make several important extensions to this previous work in order to handle the DIIM model, e.g., we take care of join constraints for defining what constitutes an acceptable solution plan and find all the query plans which represent connected query rules.

9 Conclusions

In this paper, we have presented a novel solution approach towards addressing the problem of integrating diverse information sources in a dynamically changing information space such as a Digital Library environment. Our solution incorporates an interactive query processing strategy that dynamically adapts its behavior to the system resources at hand, rather than generating plans for a fixed a-priori constructed, unified global schema across all sources. One contribution of our work is the Dynamic Information Integration Model (DIIM) which is utilized to model the content and capability descriptions of individual information sources using query-templates, bindings, and constraints as well as possible interrelationships between two or more information sources expressed as join constraints. Our approach does not require a unified global data model nor a uniform interface to all information sources in the environment. Instead, the system will permit the user to enter loosely-specified queries expressed using the known concepts of the application domain (ontology). Our system will then do the work of refining these queries into well-defined fully-specified query plans that can be executed against the current configuration of available sources.

Contributions of this work include (1) the DIIM model and query language semantics, (2) the introduction of the notion of fully specified queries that are semantically equivalent to a loosely-specified query permitted in our system (we show that our concept is a consistent and natural extension to the concept of full disjunction); (2) an algorithm for reducing the search space into a *relevant subspace* of a query and checking the necessary condition to be met by the current system configuration in order for the given query to be answerable by the system; (3) an algorithm based on CPOP that translates a loosely-specified high-level query into a set of semantically equivalent query plans that can be executed against the current configuration of available sources; (4) the steps of the resulting query plans are proven to be consistent with the binding patterns of query templates of the individual sources (capability descriptions in DIIM) and with possible interrelationships between two or more information sources (expressed as join constraints in DIIM) by defining the parameterized action library using DIIM descriptions found in the Capability Knowledgebase, and (5) the proof of correctness of our algorithms shows that the plans obtained by the query planning process correspond to semantically equivalent query plans.

We are in the process of the design and implementation of the first DIIM prototype over the metadata information space in UMDL [ABD+96] having as information sources a conspectus database using Sybase DBMS, a number of controlled vocabulary resources, e.g., Broad System of Ordering, Nasa thesaurus and name authority sources, and some other metadata sources, e.g., author indexes. The planning algorithm has already been tested in isolation with promising performance. More extensive experimental evaluations are planned.

References

[ABD+96] D.E. Atkins, W.P. Birmingham, E.H. Durfee, E. Glover, T. Mullen, E.A. Rundensteiner, E. Soloway, J. Vidal, R. Wallace, and M. Wellman. Toward Inquiry-Based Education Through Interacting Software Agents. *IEEE Computer*, May 1996.

[BCea95] A. Barrett, D. Christianson, and M. Friedman et al. UCPOP User's Manual. *Computer Science and Engineering, University of Washington, Tech. Rep.*, 1995.

[BDMS94] C.M. Bowman, P.B. Danzig, U. Manber, and M.F. Schwartz. Scalable Internet Resource Discovery: Research Problems and Approaches. *Communications of the ACM*, 37(8), August 1994.

[BS93] C. Bäckström and E. Sandewall, editors. *Current Trends in AI Planning- EWSO'93.* IOS Press, 1993.

[Cat94] R.G.G. Cattell. *Object Data Management - Object-Oriented and Extended Relational Database Systems.* 1994.

[CKP94] S. Chaudhuri, R. Krishnamurthy, and S. Potamianos. Optimizing Query with Materialized Views. *HPL-DTD-94-16, Hewlett Packard Research Laboratories, Palo Alto, CA*, 1994.

[Cor95a] Infoseek Corporation. InfoSeek. *Technical Report, http://www.infoseek.com:80/Home*, April 1995.

[Cor95b] Netscape Communications Corporation. Internet Search. *Technical Report, http://home.mcom.com/home/internet-search.html*, April 1995.

[EN94] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* The Benjamin/Cummings Publishing Company, Inc., 1994.

[GL94] C. Galindo-Legaria. Outerjoins as disjunctions . *SIGMOD*, 1994.

[GMHI+95] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. Integrating and Accessing Heterogeneous Information Sources in TSIMMIS . *Proc. of the AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environments, Stanford, California*, March 1995.

[Gol91] C.F. Goldfarb. *The SGML Handbook.* Clarendon Press, Oxford, 1991.

[KLSS95] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. *Proceedings of the AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environments, Stanford, California*, March 1995.

[KS91] H. Korth and A. Silberschatz. *Database System Concepts.* McGraw-Hill, Inc., 1991.

[LS93] G. F. Luger and W. A. Stubblefield. *AI: Structures and Strategies for Complex Problem Solving* . Benjamin/Cummings Publishing Co., Inc., 1993.

[LSK95] A. Y. Levy, D. Srivastava, and T. Kirk. Data Model and Query Evaluation in Global Information Systems. *Journal of Intelligent Information Systems. Special Issue on Networked Information Discovery and Retrieval*, 1995.

[NR95] A. Nica and E. A. Rundensteiner. Uniform Structured Document Handling Using a Constraint-based Object Approach. *Advances in Digital Libraries (ADL'95), A Forum on Research and Technology Advances in Digital Libraries, Virginia, book chapter in ADL'95, Springer Verlag*, May 1995.

[PGMU95] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. Medmaker: A Mediation System Based on Declarative Specifications. *Available by ftp at db.stanford.edu as the file pub/papakonstantinou/1995/medmaker.ps*, 1995.

[PGMW95] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object Exchange Across Heterogeneous Information Sources. *IEEE International Conference on Data Engineering*, pages 251–260, March 1995.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[RSU95] A. Rajaraman, Y. Sagiv, and J.D. Ullman. Answering Queries Using Templates With Binding Patterns. *Principles of Database Systems*, pages 105–112, May 1995.

[RU96] A. Rajaraman and J.D. Ullman. Integrating Information by Outerjoins and Full Disjunctions. *Principles of Database Systems*, 1996.

[Ull89] J.D. Ullman. *Principle of Database and Knowledge-Base Systems*. Computer Science Press, 1989.