Combining Artificial Intelligence and Databases for Data Integration

Alon Y. Levy
Department of Computer Science and Engineering
University of Washington
Seattle, Washington 98195, USA.
alon@cs.washington.edu

Abstract

Data integration is a problem at the intersection of the fields of Artificial Intelligence and Database Systems. The goal of a data integration system is to provide a uniform interface to a multitude of data sources, whether they are within one enterprise or on the World-Wide Web. The key challenges in data integration arise because the data sources being integrated have been designed independently for autonomous applications, and their contents are related in subtle ways. As a result, a data integration system requires rich formalisms for describing contents of data sources and relating between contents of different sources. This paper discusses works aimed at applying techniques from Artificial Intelligence to the problem of data integration. In addition to employing Knowledge Representation techniques for describing contents of information sources, projects have also made use of Machine Learning techniques for extracting data from sources and planning techniques for query optimization. The paper also outlines future opportunities for applying AI techniques in the context of data integration.

1 Introduction

The fields of Artificial Intelligence and Database Systems have traditionally explored opposite ends of the expressivity spectrum of representation languages [Rei88, LB87]. In database systems, where the major concern is scalability to large amounts of data, the relational model has been prevalent. Under this model, assertions about the domain of discourse are limited to ground atomic facts, and to restricted forms of integrity constraints on the contents of the database relations. In Artificial Intelligence, the focus has been on modeling more complex domains with smaller quantities of data. An essential element in modeling such domains is the ability to represent partial information, such as disjunctions and existential statements, which are not possible in the relational database model.

Data integration is a classical example of a problem that requires techniques developed in both fields [GMPQ+97, HKWY97, LRO96a, FRV96, FW97, DG97a, Coh98b, AAB+98, BEM+98, ACPS96, LR98, LK98, CCM+98]. In a nutshell, the goal of a data integration system is to provide a *uniform* interface to a multitude of data sources. As an example, consider the task of providing information about movies from data sources on the World-Wide Web (WWW). There are numerous sources on the WWW concerning movies, such as the Internet Movie Database (providing comprehensive listings of movies, their casts, directors, genres, etc.), MovieLink (providing playing times of movies in US cities), and several sites providing reviews of selected movies. Suppose we want to find which movies directed by Woody Allen are playing tonight in Seattle, and their respective reviews. None of these data sources *in isolation* can answer this query. However, by combining data from multiple sources, we can answer queries like this one, and even more complex ones. To answer our query, we would first search the Internet Movie Database for the list of movies directed by Woody Allen, and then feed the result into the MovieLink database to check which ones are playing in Seattle. Finally, we would find reviews for the relevant movies using any of the movie review sites.

M.J. Wooldridge and M. Veloso (Eds.): Artificial Intelligence Today, LNAI 1600, pp. 249-268, 1999.

The most important advantage of a data integration system is that it enables users to focus on specifying *what* they want, rather than thinking about *how* to obtain the answers. As a result, it frees the users from the tedious tasks of finding the relevant data sources, interacting with each source in isolation using a particular interface, and combining data from multiple sources.

The main characteristics distinguishing data integration systems from distributed and parallel database systems is that the data sources underlying the system are *autonomous*. In particular, a data integration system provides access to *pre-existing* sources, which were created independently. Unlike multidatabase systems (see [LMR90] for a survey) a data integration system must deal with a large and constantly changing set of data sources. These characteristics raise the need for richer mechanisms for describing our data, and hence the opportunity to apply techniques from Knowledge Representation. In particular, a data integration system requires a flexible mechanism for describing contents of sources that may have overlapping contents, whose contents are described by complex constraints, and sources that may be incomplete or only partially complete. Languages originating from Knowledge Representation formalisms have shown to be useful in capturing such complex relationships between data sources.

In this paper I survey the application of AI techniques and their combination with database techniques in the context of data integration. Although the focus is on the application of Knowledge Representation techniques, I also discuss the application of Machine Learning techniques to the problem of extracting data out of sources, and of techniques for interleaving planning and execution for the purpose of query optimization in the more dynamic environment of data integration. Section 2 is a brief introduction to database systems terminology. Section 3 describes the novel challenges encountered in data integration systems, and provides a reference architecture for such a system. Section 4 considers the problem of modeling the contents of data sources, and 5 discusses the modeling of source (in)completeness. Section 6 describes the issues concerning the construction of wrapper programs, whose task is to extract structured data from data sources, and the application of Machine Learning techniques to this task. Section 7 describes the novel issues that arise for query optimization in the context of data integration systems, and the need for interleaving of planning and execution. Section 8 describes the problem of web-site management which is currently one of the significant applications of data integration, and in itself presents several important opportunities for future AI research. Section 9 contains concluding remarks.

Let me state at the outset that this paper is not meant to be a comprehensive survey of the work on data integration in the AI community. My goal is simply to highlight the main issues that arise, and to provide a flavor of the solutions. I apologize in advance for any omissions, of which I am sure there are many.

2 Schemas and Queries

Our discussion will use the terminology of relational databases. A *schema* is a set of relations. Columns of relations are called attributes, and their names are part of the schema (traditionally, the type of each attribute is also part of the schema but we will ignore typing here).

Queries can be specified in a variety of languages. For simplicity, we consider the language of conjunctive queries, and several variants on it. A conjunctive query has the form:

$q(\bar{X}) :- e_1(\bar{X}_1), \ldots, e_n(\bar{X}_n),$

where $e_1, \ldots, e_n$ are database relations, and $\bar{X}_1, \ldots, \bar{X}_n$ are tuples of variables or constants. The atom $q(\bar{X})$ is the head of the query, and the result of the query is a set of tuples, each giving a binding for every variable in $\bar{X}$. Interpreted predicates such as $<, \leq, \neq$ are sometimes used in the query. Queries with unions are expressed by multiple rules with the same head. A *view* refers to a named query.

3 Challenges in Data Integration

As described in the introduction, the task of a data integration system is to provide a uniform interface to a collection of data sources. The data sources can either be full-fledged database systems (of various flavors: relational, object-oriented, etc.), legacy systems, or structured files hidden behind some interface program. For the purposes of our discussion we model data sources as containing relations. In this paper (as in most of the research) we only consider data integration systems whose goal is to query the data, and not to perform updates on the sources.

Throughout the discussion it is instructive to keep in mind the distinction between different classes of data integration applications. For example, integration of arbitrary sources on the WWW is quite a different task from that of integrating multiple sources within a single enterprise (though it is unclear which one is harder!). In the latter case, the sources are not as autonomous as they are on the WWW, but the requirements imposed on a data integration system may be more stringent.

To understand the challenges involved in building data integration systems, we briefly compare the problems that arise in this context with those encountered in traditional database systems. Figure 1 illustrates the different stages in processing a query in a data integration system.

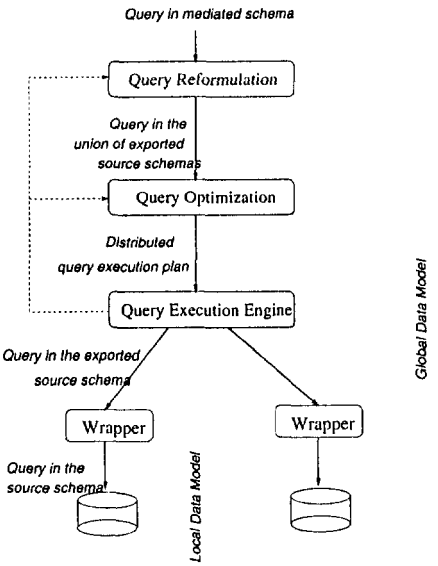

Figure 1: Prototypical architecture of a data integration system

Data modeling: in a traditional database application one begins by modeling the requirements of the application, and designing a database schema that appropriately supports the application. As noted earlier, a data integration application begins from a set of pre-existing data sources. Hence, the first step of the application designer is to develop a *mediated schema* that describes the data that exists in the sources, and exposes the aspects of this data that may be of interest to users. Note that the mediated schema does not necessarily contain all the relations and attributes modeled in each of the sources. Users pose queries in terms of the mediated schema, rather than directly in terms of the source schemas. As such, the mediated schema is a set of *virtual* relations, in the sense that they are not actually stored anywhere. For example, in the movie domain, the mediated schema may contain the relation MOVIEINFO(ID, TITLE, GENRE, COUNTRY, YEAR, DIRECTOR) describing the different properties of a movie, the relation MOVIEACTOR(ID, NAME), representing the cast of a movie, and MOVIEREVIEW(ID, REVIEW) representing reviews of movies.

Along with the mediated schema, the application designer needs to supply *descriptions* of the data sources. The descriptions specify the relationship between the relations in the mediated schema and those in the local schemas at the sources. The description of a data source specifies its contents (e.g., contains movies), attributes (e.g., genre, cast), constraints on its contents (e.g., contains only American movies), completeness and reliability, and finally, its query processing capabilities (e.g., can perform selections, or can answer arbitrary SQL queries).

The fact that data sources are pre-existing requires that we be able to handle the following characteristics in the language for describing the sources:

1. *Overlapping* and even *contradictory* data among different sources.

2. Semantic mismatches among sources: since each of the data sources has been designed by a different organization for different purposes, the data is modeled in different ways. For example, one source may store a relational database in which all the attributes of a particular movie are stored in one table, while another source may spread the attributes across several relations. Furthermore, the names of the attributes and of the tables will be different from one source to another, as will the choice of what should be a table and what should be an attribute.

3. Different naming conventions for data values: sources use different names or formats to refer to the same object. Simple examples include various conventions for specifying addresses or dates. Cases in which persons are named differently in the sources are harder to deal with (e.g., one source contains the full name, while another contains only the initials of the first name). This problem is not discussed further in this paper (see [Coh98b] for an elegant treatment of the problem).

Query reformulation: a user of a data integration system poses queries in terms of the mediated schema, rather than directly in the schema in which the data is stored. As a consequence, a data integration system must contain a module that uses the source descriptions in order to *reformulate* a user query into a query that refers directly to the schemas of the sources. Such a reformulation step does not exist in traditional database systems. Clearly, as the language for describing data sources becomes more expressive, the reformulation step becomes harder. Aside from wanting the reformulation to be semantically correct (i.e., the answers obtained from the sources will actually be correct answers to the query), an important goal of query reformulation is to ensure that we do

not access irrelevant sources (i.e., sources that cannot contribute any answer or partial answer to the query). Data source modeling and query reformulation are discussed in Section 4.

Wrappers: the other layer of a data integration system that does not exist in a traditional system is the wrapper layer. Unlike a traditional query execution engine that communicates with a local storage manager to fetch the data, the query execution plan in a data integration system must obtain data from remote sources. A wrapper is a program which is specific to a data source, whose task is to translate data from the source to a form that is usable by the query processor of the system. For example, if the data source is a web site, the task of the wrapper is to translate the query to the source's interface, and when the answer is returned as an HTML document, it needs to extract a set of tuples from that document. Wrapper construction is described in Section 6.

Query optimization and execution: a traditional relational database system accepts a *declarative* SQL query. The query is first parsed and then passed to the *query optimizer*. The role of the optimizer is to produce an efficient *query execution plan*, which is an imperative program that specifies exactly how to evaluate the query. In particular, the plan specifies the *order* in which to perform the different operations in the query (join, selection, projection), a specific algorithm to use for each operation (e.g., sort-merge join, hash-join), and the scheduling of the different operators (in cases where parallelism is possible). Typically, the optimizer selects a query execution plan by searching a space of possible plans, and comparing their estimated cost. To evaluate the cost of a query execution plan the optimizer relies on extensive statistics about the underlying data, such as sizes of relations, sizes of domains and the selectivity of predicates. Finally, the query execution plan is passed to the query execution engine which evaluates the query.

The main differences between the traditional database context and that of data integration are the following:

- Since the sources are autonomous, the optimizer may have no statistics about the sources, or unreliable ones. Hence, the optimizer cannot compare between different plans, because their costs cannot be estimated.
- Since the data sources are not necessarily database systems, the sources may appear to have different processing capabilities. For example, one data source may be a web interface to a legacy information system, while another may be a program that scans data stored in a structured file (e.g., bibliography entries). Hence, the query optimizer needs to consider the possibility of exploiting the query processing capabilities of a data source. Note that query optimizers in distributed database systems also evaluate where parts of the query should be executed, but in a context where the different processors have identical capabilities.
- Finally, in a traditional system, the optimizer can reliably estimate the time to transfer data from the disc to main memory. But in a data integration system, data is often transferred over a wide-area network, and hence delays may occur for a multitude of reasons. Therefore, even a plan that appears to be the best based on cost estimates may turn out to be inefficient if there are unexpected delays in transferring data from one of the sources accessed early on in the plan.

Semistructured data: an issue that cuts across many layers of a data integration system is the problem of managing semistructured data. The term semistructured data has been used to refer

to data that does not necessarily fit into a rigidly predefined schema, as is required in traditional database systems. This may arise because the data is very irregular (e.g., objects of the same type may have varying sets of attributes, attribute names are used in an irregular fashion), and hence can be described only by a schema that is relatively large. In other cases, the schema may be rapidly evolving, or not even declared at all (i.e., it may be implicit in the data). The database community has developed several methods to model and query for semistructured data (see [Bun97, Abi97] for recent surveys). Semistructured data is important for data integration systems for two reasons: first, in many cases, the data in the sources is semistructured; second, when integrating data from many sources, each with differing data models, it is convenient to consider a data model that is the least common denominator of these models. Data models for semistructured data, based on labeled directed graphs, tend to have this property. It should also be noted that XML, the emerging standard for data exchange over the WWW, has many features in common with semistructured data. An application of Description Logics to the the problem of reasoning about semistructured data is described in [CGL98].

4 Modeling Data Sources and Query Reformulation

As described in the previous section, one of the main differences between a data integration system and a traditional database system is that users pose queries in terms of a mediated schema. The data, however, is stored in the data sources, organized under local schemas. Hence, in order for the data integration system to answer queries, there must be some description of the relationship between the source relations and the mediated schema. The query processor of the integration system must be able to reformulate a query posed on the mediated schema into a query against the source schemas.

In principle, one could use arbitrary formulas in first-order logic to describe the data sources. But in such a case, sound and complete reformulation would practically impossible. Hence, several approaches have been explored in which restricted forms of first-order formulas have been used in source descriptions, and effective accompanying reformulation algorithms have been presented. These approaches include: *Global as view* (GAV) [GMPQ+97, PAGM96, ACPS96, HKWY97, FRV96, TRV98], *Local as view* (LAV) [LRO96b, KW96, DG97a, DG97b, FW97], and the use of Description Logics [AKS96, CL93, LRO96a, LR98].[1] Description Logics are covered in a different paper in this volume, and we do not elaborate on it any further here.

Global As View: This approach has its origins in multidatabase systems (e.g., Multibase [LR82]). In the GAV approach, for each relation R in the mediated schema, we write a query over the source relations specifying how to obtain R's tuples from the sources.

For example, suppose we have two sources DB_1 and DB_2 containing titles, directors and years of movies. We can describe the relationship between the sources and the mediated schema relation MOVIEYEAR as follows:

$DB_1(id, title, director, year) \Rightarrow MovieYear(title, year)$
$DB_2(id, title, director, year) \Rightarrow MovieYear(title, year).$

If we have a third source that shares movie identifiers with DB_1 and provides movie reviews, the following sentence describes how to obtain tuples for the MOVIEREVIEW relation:

[1] It should be noted that although Description Logics were used in [AKS96] to describe the data sources, the reformulation problem is viewed as a planning problem, and hence solved by a specialized planner.

$DB_1(id, title, director, year) \wedge DB_3(id, review) \Rightarrow MovieReview(title, director, review)$

In general, GAV descriptions are Horn rules that have a relation in the mediated schema in the consequent, and a conjunction of atoms over the source relations in the antecedent.

Query reformulation in GAV is relatively straightforward. Since the relations in the mediated schema are defined in terms of the source relations, we need only unfold the definitions of the mediated schema relations. For example, suppose our query is to find reviews for 1997 movies:

$q(title, review) :- MovieYear(title, 1997), MovieReview(title, review).$

Unfolding the descriptions of MOVIEYEAR and MOVIEREVIEW will yield the following queries over the source relations: (the second of which will obviously be deemed redundant)

$q(title, review) :- DB_1(id, title, director, year), DB_3(id, review)$
$q(title, review) :- DB_1(id, title, director, year),$
$\qquad DB_2(title, director, year), DB_3(id, review)$

Local As View: The LAV approach is the opposite of GAV. Instead of writing rules whose consequents are relations in the mediated schema, the rules contain a conjunction of atoms over the mediated schema in the consequent, and an atom of the source relation in the antecedent. That is, for every data source S, we write a rule over the relations in the mediated schema that describes which tuples are found in S.

Suppose we have two sources: (1) V_1, containing titles, years and directors of American comedies produced after 1960, and (2) V_2 containing movie reviews. In LAV, we would describe these sources by the following sentences (variables that appear only on the right hand sides are assumed to be existentially quantified):

$S_1 : V_1(title, year, director) \Rightarrow Movie(title, year, director, genre) \wedge American(director) \wedge$
$\qquad year \geq 1960 \wedge genre = Comedy.$

$S_2 : V_2(title, review) \Rightarrow Movie(title, year, director, genre) \wedge year \geq 1990 \wedge Review(title, review).$

Query reformulation in LAV is more tricky than in GAV, because it is not possible to simply unfold the definitions of the relations in the mediated schema. In fact, the reformulation problem here leads to a new inference problem, which can be explained intuitively as follows. Because of the form of the LAV descriptions, each of the sources can be viewed as containing an answer to a query over the mediated schema (the one expressed by the right hand side of the source description). Hence, sources represent materialized answers to queries over the virtual mediated schema. A user query is also posed over the mediated schema. The problem is therefore to find a way of answering the user query using only the answers to the queries describing the sources.

For example, suppose our query asks for reviews for comedies produced after 1950:

$q(title, review) : -Movie(title, year, director, Comedy), year \geq 1950, Review(title, review).$

The reformulated query on the sources would be:
$q'(title, review) : -V_1(title, year, director), V_2(title, review).$

The LAV reformulation problem is very closely related to the problem of answering queries using views, studied in the database literature [YL87, TSI96, LMSS95, CKPS95, RSU95, DG97b]. This problem has received significant attention because of its relevance to other database problems,

such as query optimization [CKPS95], maintaining physical data independence [YL87, TSI96], and data warehouse design.

As it turns out, the query reformulation problem is in general NP-complete in the size of the source descriptions and user query even when the queries describing the sources and the user query are conjunctive and don't contain interpreted predicates [LMSS95]. However, in this and other important cases, reformulation is still polynomial in the number of data sources, and more importantly, answering queries is polynomial in the size of the data in the sources. Algorithms for query reformulation in LAV have been considered in [LRO96a, DG97a, FW97, LK98].

An interesting phenomenon in several variants of LAV descriptions is that the reformulated query may actually turn out to be a *recursive* query over the sources. The most interesting of these variants is the common case in which data sources can only be accessed with particular patterns.

Consider the following example, where the first source provides papers (for simplicity, identified by their title) published in AAAI, the second source records citations among papers, and the third source stores papers that have won significant awards. The superscripts in the source descriptions depict the access patterns that are available to the sources. The superscripts contain strings over the alphabet $\{b, f\}$. If a b appears in the i'th position, then the source requires a binding for the i'th attribute in order to produce provide answers. If an f appears in the i'th position, then the i'th attribute may be either bound or not. From the first source we can obtain all the AAAI papers (no bindings required); to obtain data from the second source the first we must provide a binding for a paper and then receive the set of papers that it cites; with the third source we can only query whether a *given* query won an award, but not ask for all the award winning papers.

$AAAIdb^f(X) \Rightarrow AAAIPapers(X)$
$CitationDB^{bf}(X,Y) \Rightarrow Cites(X,Y)$
$AwardDB^b(X) \Rightarrow AwardPaper(X)$

Suppose our query is to find all the award winning papers:

$Q(X) : -AwardPaper(X)$

As the following queries show, there is no finite number of conjunctive queries over the sources that is guaranteed to provide *all* the answers to the query. In each query, we can start from the AAAI database, follow citation chains of length n, and feed the results into the award database. Since we cannot limit the length of a citation chain we need to follow apriori (without examining the data), we cannot put a bound on the size of the reformulated query.

$Q'(X) : -AAAIdb(X), AwardDB(X)$
$Q'(X) : -AAAIdb(V), CitationDB(V, X_1), \ldots, CitationDB(X_n, X), AwardDB(X).$

However, if we consider recursive queries over the sources, we can obtain a finite concise query that provides all the answers, as follows (note that the newly invented relation *papers* is meant to represent the set of all papers reachable from the AAAI database):

$papers(X) : -AAAIdb(X)$
$papers(X) : -papers(Y), CitationDB(Y, X)$
$Q'(X) : -papers(X), AwardDB(X).$

Other cases in which recursion may be necessary are in the presence of functional dependencies on the mediated schema [DL97], when the user query is recursive [DG97a], and when the descriptions of the sources are enriched by description logics [BLR97].

Finally, it turns out that slight changes to the form of source descriptions in LAV can cause the problem of answering queries to become NP-hard in the size of the data in the sources [AD98]. Most of these cases arise when we discuss completeness information (Section 5), but even in our case, it may happen, for example in the case where the query contains the predicate $\neq$. This phenomenon hints at the fact that in some sense, LAV has a greater expressive power than GAV.

A Comparison of the Approaches: The main advantage of the GAV approach is that query reformulation is very simple, because it reduces to rule unfolding. However, adding sources to the data integration system is non-trivial. In particular, given a new source, we need to figure out all the ways in which it can be used in order obtain tuples for each of the relations in the mediated schema. Therefore, we need to consider the possible interaction of the new source with each of the existing sources, and this limits the ability of the GAV approach to scale to a large collection of sources.

In contrast, in the LAV approach each source is described in isolation. It is the system's task to figure out (at query time) how the sources interact and how their data can be combined to answer the query. The downside, however, is that query reformulation is harder, and sometimes requires recursive queries over the sources. An additional advantage of the LAV approach is that it is easier to specify rich constraints on the contents of a source (simply by specifying more conditions in the source descriptions). Specifying complex constraints on sources is essential if the data integration system is to distinguish between sources with closely related and overlapping data.

5 Modeling Source Completeness

The source descriptions we considered in the previous section only allowed us to express necessary conditions for finding a certain piece of data in a source but not sufficient conditions. For example, we were able to express that V_1 contains American comedies produced after 1960, but we could not state that the source contains *all* such movies.

Such source descriptions are adequate to capture incomplete sources, which at least in the case of sources on the WWW, covers many of the cases. However, there are situations in which we want to state that a source is complete, or at least partially complete. For example, the DB&LP Database[2] contains the complete set of papers published in most major database conferences; the Library of Congress web site contains the complete list of books published in the U.S. When modeling data sources within an enterprise, completeness assertions are even more common.

Knowledge of a data source's completeness can help a data integration system in several ways. Most importantly, since a *negative* answer from a complete source is meaningful, the data integration system can prune access to other sources. For example, if our query asks whether a certain person authored a paper in SIGMOD-98, we need only consult the DB&LP bibliography, and not other sources that may contain overlapping data.

Local completeness assertions were first considered in the context of the Internet Softbot project [EGW94]. In that work, the authors considered statements of the form:[3]

$$LCW(V_1(title, director, year), American(director) \wedge year \geq 1960)$$

[2] `http://www.acm.org/sigmod/dblp/db/index.html`

[3] LCW stands for Local Closed World [EGW94].

meaning that the movie source is complete with respect to films produced by American directors after 1960. These statements were shown to be a restricted form of more general non-monotonic reasoning formalisms.

The inference question that arises in the presence of such completeness statements is the *answer completeness problem:* given a user query Q, a reformulation Q' of Q over the data sources, and a set of source completeness statements, does Q' produce the complete answer for Q? For example, if the user asks for titles of recent American comedies, the source above would provide a complete answer, but if the user also asks for reviews of such movies then the answer may be incomplete if our review sources are incomplete.

In [EGW94] the authors describe several rules for inferring completeness statements for queries. In [Lev96] is it shown that this inference problem is closely related to the problem of deciding whether a database query is independent of an insertion or deletion update to the database. This relationship enables the transfer of several algorithms and complexity results to the answer completeness problem. Several other works have considered more expressive forms of local completeness statements ([Dus97, FW97]). Finally, Abiteboul and Duschka [AD98] show that in the presence of completeness assertions in LAV descriptions, the complexity of answering a query becomes NP-hard in the size of the data in the sources.

Using probabilistic information for data integration: a different approach to handling incompleteness and source overlap based on using probabilistic information is proposed in [FKL97]. To motivate the use of probabilistic information, consider a data integration system providing access to multiple bibliography data sources available online.[4] Figure 2 shows a set of classes (i.e., unary relations) and a set of attributes we may associate with papers in our domain. The schema contains a class CS-Paper denoting the set of publications in Computer Science. One partition of the CS-paper class is by publication type, i.e., the classes Journal, Conference and Book, which are assumed to be pairwise disjoint. A second partition is by a topic hierarchy. Note that whereas in the first case, the classes in the partition were pairwise disjoint, these classes in the second partition are obviously overlapping.

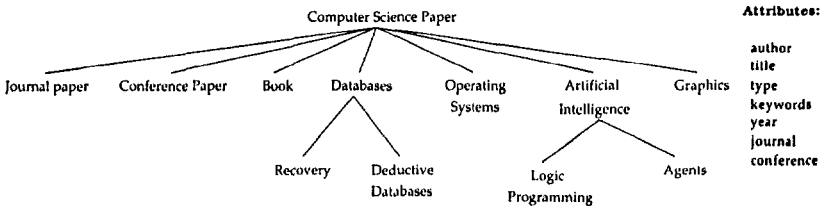

Figure 2: Mediated schema for publication domain.

The first kind of probabilistic information we would like to consider is a generalization of the completeness statements described above. Instead of specifying that a source is complete with respect to its content, we would want to express the probability of finding a certain data item in the source. For example, we would like to specify that the probability of finding an arbitrary paper on Deductive Databases in source S_1 is 0.8, which we can denote by $P(S_1 \mid DDB) = 0.8$. As a result, we are able to distinguish between sources that are relatively complete and those that are rather sparse.

[4] See `http://glimpse.cs.arizona.edu/bib/` for a large class of such sources.

The second kind of information concerns overlap between classes in the mediated schema. With purely logical formalisms, we can only express three kinds of relationships between classes: (1) one class is a superset of the other; (2) two classes are disjoint; (3) there is *some* overlap between a pair of classes. In our example, since no pair of topics in Computer Science are completely disjoint, if we asked for papers about Database Systems, the data integration system cannot prune *any* source containing Computer Science papers. However, we would like to be able at least to order the access to the sources, depending on their potential relevance to the query. Hence, we need a mechanism for specifying the *degree* of overlap between the classes. For example, we would like to specify that $P(AI \mid DB) = 0.05$, denoting that the probability that a Database paper is also about AI is 0.05, and $P(DB \mid OS) = 0.2$, showing that the overlap between the fields of Databases and Operating Systems is much larger.[5]

Finally, we may want to represent information on the overlap between information sources. While it is possible to assume that the overlap between sources can be automatically derived from the first two kinds of information, in some particular cases we may have more specific overlap information. For example, one source may be known to be a subset of another.

The work in [FKL97] describes a method for describing these three kinds of probabilistic information, and algorithms that exploit this information to order the access to data sources. The key challenge in designing a formalism for specifying probabilistic information is that the size of the specification is exponential in the size of number of classes considered in the schema. For example, suppose we want to specify a probability distribution on the set of topics of Computer Science. That is, for every set of topics $\mathcal{A}$ we want to know the probability that a paper belongs to all the topics in $\mathcal{A}$. To do so, we need to specify 2^n numbers, where n is the number of topics. This presents two problems. First, from a modeling viewpoint, we do not want to specify such a large number of probabilities. Second, performing computations with such a large set of probabilities will be prohibitively expensive. Instead, we would like to specify only a small number of probabilities (e.g., intersections only between pairs or subset of the pairs of topics in Computer Science), and to efficiently compute the other probabilities that may be needed in the process of source selection. It is interesting to note that Bayesian Networks are inadequate for describing the overlap between classes in the mediated schema because the set of classes are highly related to each other (leading to a highly connected network). Instead, [FKL97] describes a tree-based representation of the probability distribution.

6 Extracting Data from Sources: Wrapper Construction

In a traditional database system tuples of the database relations are obtained by the query execution engine via an interface to the local storage manager. However, a data integration system must communicate with external sources in order to obtain data. External sources typically do not provide the data integration system with a stream of tuples in a standard format, but rather in a format native to the source. Hence, the integration system communicates with each source through a wrapper program, whose main task is to pose queries to the data source and convert the answer into a format that can be manipulated by the execution engine.

When the data source is a web-site, the answer to a query is usually an HTML document. The wrapper must then extract a set of answer tuples out of the resulting HTML page. The main difficulty in building wrappers is that the HTML page is usually designed for human viewing, rather than for programmatic manipulation data by programs. Hence, the data is often embedded

[5] The probabilities in this example are pure fiction.

in natural language text or hidden within graphical presentation primitives. Moreover, the form of the HTML pages changes frequently (even within one web-site), making it hard to write rules for extracting the data from the HTML file and to maintain these rules over time.

Several works have considered the problem of building tools for rapid creation of wrappers. One class of tools (e.g., [HGMN+98, GRVB98]) is based on developing specialized grammars for specifying how the data is laid out in an HTML page, and therefore how to extract the required data.

A second class of techniques for building wrappers is based on developing inductive learning techniques for automatically learning a wrapper. Using such algorithms, we provide the system with a set of HTML pages where the data in each page is labeled. The algorithm uses the labeled examples to automatically learn a grammar by which the data can be extracted from subsequent pages. Naturally, the more examples we give the system, the more accurate the resulting grammar can be, and the challenge is to discover wrapper languages that can be learned with a small number of examples.

The first formulation of wrapper construction as inductive learning was presented in [KDW97]. In that work, the authors identified HLRT, a class of wrappers which is efficiently learnable, yet expressive enough to handle numerous actual Internet information resources. HLRT is designed for resources that display their content in a tabular layout. HLRT wrappers scan their input for substrings that delimit the information to be extracted. For example, in the context of Internet sources, these delimiters can be HTML tags. HLRT corresponds essentially to a class of finite-state automata, so wrapper induction is similar to FSA induction (e.g., [Ang82]). Since FSAs run in linear time, HLRT satisfies the desire that wrappers be fast. However, since wrappers are used for parsing (rather than just classification), the learned FSA must have a specific state topology. Existing FSA induction algorithms do not make such guarantees, hence specialized algorithms for HLRT induction had to be developed.

Other works that have considered this approach to wrapper construction include [AK97] that exploits heuristics specific to the common uses of HTML in order to obtain faster learning, and [Coh98a] who describes an algorithm for learning wrappers that do not output their answers in tabular form, but rather in the form of nested tables. Finally, we note that the emergence of XML may lead web-site builders to export the data underlying their sites in a machine readable form, thereby greatly simplifying the construction of wrappers. However, it should be emphasized that XML does not solve the problems of semantic integration.

The work described [CDF+98] is a first step in bridging the gap between the approaches of Machine Learning and of Natural Language Processing to the problem of wrapper construction. Another important use of Machine Learning in the context of data integration has been to learn the mapping between the source schemas and the mediated schemas [PE95, DEW97].

7 Query Optimization and Execution

In a traditional database system, query optimization refers to the process of translating a declarative query (e.g., in SQL) into a query execution plan, i.e., a specific sequence of steps that the query execution engine should follow to evaluate the query. As we have discussed earlier, in the data integration context, some optimizations are done already at the query reformulation step: the query is reformulated as to not access any irrelevant or redundant data sources, and to pose the most specific query possible to each of the sources it does access.

By and large, the query optimization problem for data integration has received relatively little attention, compared to the problems of query reformulation and wrapper construction. As stated earlier, the challenges for query optimization arise because the sources have different processing capabilities, there are few statistics about the data sources, and data transfer rates over the network cannot always be anticipated in advance. The issue of describing the query processing capabilities of a data source and creating query execution plans that conform to these capabilities is considered in [LRO96b, PGGMU95, VP97, HKWY97]. Adapting query execution plan in the presence of network delays is considered in [UFA98].

The above characteristics of the data integration problem force us to reconsider the architecture of a query processor for data integration. Whereas a traditional system first decides on a query execution plan and then executes it, the absence of statistics and the need to adapt to network delays renders such a two-phase approach infeasible. Hence, an approach followed in recent work [Kno95, IFF+98] is to interleave optimization and execution of the query. Instead of creating a complete plan for the query in advance, the query optimizer may create a plan fragment, which answers only part of the query. Once this fragment has been executed, the optimizer may have additional information to use for planning for the subsequent parts of the query. The main challenge in this architecture is to find the effective points for interleaving the optimization and execution.

8 Web-Site Management

One of the prime forces driving many data integration applications is web-site construction and management. The first step that an enterprise needs to tackle when constructing a web-site is to build a coherent view of all the data that is available across the enterprise. Providing a uniform interface to all this data significantly simplifies the task of constructing the site. Furthermore, results of data integration queries in this context are not considered in isolation, but rather as entry points into complex webs of data (e.g., hyperlinks to information related to the answer).

The problem of building web-sites with complex structures that serve data from multiple sources has very recently received significant attention in the Database community. The theme underlying many of the works in this area is to represent the *content* and the *structure* (i.e., the set of pages, the data at each page and the links between the pages) of a web-site in a declarative fashion. The main advantage of this approach is the ability to easily *restructure* a web-site and to construct multiple versions of a web-site from the same underlying content (e.g., consider a company that creates an internal web-site for its employees and several external ones for its customers, suppliers, or other affiliate companies).

The realization that web-sites can be constructed (or at least, modeled) as richly connected pieces of data/knowledge, presents an important opportunity for future contributions of AI techniques. Before discussing such opportunities, I briefly describe the STRUDEL system [FFK+98], which was the first to introduce the idea of constructing web-sites using declarative representations.

The architecture of STRUDEL is shown in Figure 3. At the bottom level, STRUDEL uses a data integration system to accessed a set of data sources containing the data that will be served on the web-site. The data may be stored in databases, in structured files, or in existing web-sites. The data is represented throughout the system as a labeled directed graph, in the spirit of models proposed for semistructured data [Bun97, Abi97].

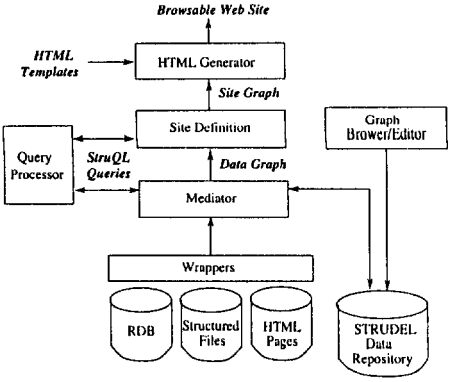

Figure 3: Architecture for Web-Site Management Systems

The main step in building a web-site is to write a declarative expression that represents the structure of the site. The expression is written in the STRUQL language, which is a language for querying and restructuring graphs. That is, STRUQL accepts a set of graphs as input, and outputs a graph. The result of applying this query to the underlying data, called the *site-graph,* is the logical representation of the web-site. The logical representation specifies the pages of the site, the data at each page, and the links between the pages, but *not* how to graphically render each page. To create HTML pages the system applies a set of HTML templates to the nodes in the site graph. An HTML template is an HTML file where some constants are replaced by variable names that can be bound to different values.

STRUDEL is best illustrated by an example. Consider the construction of a simplified version of a researcher's homepage. The source of raw data is a Bibtex bibliography that contains the researcher's publications. In the data graph, we represent this data by a class PUBLICATIONS, as seen in Figure 4 (note that the data can easily be represented by a labeled graph).

```
object pub1 in Publications {
  title "Web-Sites With Common Sense"
  author "John McCarthy"
  author "Tim Berners-Lee"
  year   1998
  booktitle "AAAI 98"
  pub-type  "inproceedings"
  abs-file  "abstracts/bm98"
  ps-file   "proceedings/aaai98.ps"
  category  "Philosophical Foundations"
  category  "Knowledge Representation"
}
```

Figure 4: Fragment of data graph for homepage site

The structure of the homepage site is defined by the STRUQL expression in Figure 5. The site has four types of pages: a root page containing general information, an "All Titles" page containing the list of titles of the researcher's papers, a "category" page containing summaries of papers in a particular category, and a "Paper Presentation" page for each paper.

The first clause creates the RootPage and AllTitles pages and links them. Lines 7–9 create a page for each publication, and links the publication page to each of its attributes. Note that we copy all the attributes of a given publication using the arc variable L, which gets bound to attribute names, rather than objects. Lines 12–16 consider the category attribute of each publication and create the appropriate category pages with links to the appropriate publication pages. Finally, lines 19–21 link the "All Titles" page to the titles of all the papers and the papers' individual pages.

```
1   INPUT BIBTEX
2   // Create root page and abstracts page and link them
3   CREATE RootPage(), AllTitlesPage()
4   LINK   RootPage() -> "All Titles" -> AllTitlesPage()
5
6   // Create a presentation for every publication x
7   WHERE  Publications(X), X -> L -> V
8   CREATE PaperPresentation(X)
9   LINK   PaperPresentation(X) -> L -> V,
10
11  // Create a page  for every category
12    { WHERE  L = "category"
13      CREATE CategoryPage(V)
14      LINK   CategoryPage(V)  -> "Paper" -> PaperPresentation(X),
15             CategoryPage(V)  -> "Name" ->  V
16
17      // Link root page to each category page
18             RootPage()  -> "CategoryPage"  -> CategoryPage(V) }
19    { WHERE L = "title"
20      LINK  AllTitlesPage() -> "title" -> V,
21            AllTitlesPage -> "More Details" -> PaperPresentation(X) }
22  OUTPUT HomePage
```

Figure 5: Site definition query for example homepage site

The main point to note about STRUDEL and its sibling systems (e.g., [AMM98, AM98, CDSS98, PF98, JB97, TN98]) is that the structure of the site is specified declaratively, and hence restructuring the site or creating multiple versions of the site amount to writing different site definition queries. Given that web-sites can be specified declaratively, one can start tackling higher-level issues in managing web-sites, as the following.

Reasoning about integrity constraints: as builders of web-sites, we would like to enforce constraints on the structure of our site (e.g., no dangling pointers, an employee's homepage should point to their department's homepage). Clearly, once we have created the web-site, we can go through it and check whether the constraints are satisfied, but in that case, we would have to repeat the check every time the web-site is updated. A more interesting approach is to look at the intensional definition of the structure of the web-site (i.e., the STRUQL query), and verify that a certain integrity constraint will hold for every web-site generated by the definition. This leads to a problem which is similar in spirit to that of knowledge base verification [LR96, SS97]. A first attempt to address this problem is described in [FFLS98].

Rule-based specification of web-sites: in the above approach, we specify the structure of the web-site, and then check whether the given integrity constraints are guaranteed to hold for every instantiation of the structure. A different approach is to specify the structure of the site at a higher level: to specify only the integrity constraints for the site. The system would then consider the constraints and would propose a structure for the web-site. This approach can be useful in cases where most of the structure is driven by constraints. For example, [LS98] considers the design of online stores and proposes a set of rules that, if followed, would improve the site design. The challenge we face is to build a system that receives the constraints as input, and outputs the *best* possible site structure.

Automatically restructuring web-sites: the short experience in building web-sites has already shown that it is a highly iterative process. Even after the web-site is up, designers will frequently want to restructure it after understanding the patterns with which users browse the site. Perkowitz and Etzioni [PE97] have proposed the notion of *adaptive* web-sites that restructure themselves automatically. Declarative representations of web-sites provide a basis on which to build adaptive web-sites. In particular, once we have a model of a web-site, we can analyze the user browsing patterns and propose ways to restructure the web-site.

9 Conclusions

As shown in this paper, AI techniques have had a significant impact on building data integration systems. Of particular note is the use of various Knowledge Representation techniques for modeling contents of data sources. Machine Learning algorithms offer the most promising methods for rapid construction of wrappers.

Furthermore, current problems being addressed by the data integration community can also benefit from AI techniques: the need for techniques for interleaving query optimization and execution, use of probabilistic reasoning for modeling domains of data integration, and the use of Machine Learning algorithms for automatically deriving source descriptions. The area of web-site management poses several important challenges for AI researchers.

As we embark on these and future challenges it is important to keep in mind a couple of principles. First, an AI technique applied in isolation to a data integration problem, but rather in the context of a larger system. Hence, for an AI technique to have significant impact on the practice of data integration, one must understand well the rest of the system and how to best tailor the technique to this setting. This principle applies especially in the consideration of Knowledge Representation techniques and in future applications of planning methods. Second, data integration research has matured to the point where significant progress must be validated empirically. Hence I believe that future contributions must be accompanied by significant performance studies.

Acknowledgements

I would like to thank Dana Florescu, Zack Ives, Nick Kushmerick, Werner Nutt, Rachel Pottinger and Dan Weld for their input in writing this paper.

References

[AAB+98] Jos Luis Ambite, Naveen Ashish, Greg Barish, Craig A. Knoblock, Steven Minton, Pragnesh J. Modi, Ion Muslea, Andrew Philpot, and Sheila Tejada. ARIADNE: A system for constructing mediators for internet sources (system demonstration). In *Proc. of ACM SIGMOD Conf. on Management of Data*, Seattle, WA, 1998.

[Abi97] Serge Abiteboul. Querying semi-structured data. In *Proc. of the Int. Conf. on Database Theory (ICDT)*, Delphi, Greece, 1997.

[ACPS96] S. Adali, K. Candan, Y. Papakonstantinou, and V.S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Montreal, Canada, 1996.

[AD98] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Seattle, WA, 1998.

[AK97] Naveen Ashish and Craig A. Knoblock. Wrapper generation for semi-structured internet sources. *SIGMOD Record*, 26(4):8-15, 1997.

[AKS96] Yigal Arens, Craig A. Knoblock, and Wei-Min Shen. Query reformulation for dynamic information integration. *International Journal on Intelligent and Cooperative Information Systems*, (6) 2/3:99-130, June 1996.

[AM98] Gustavo Arocena and Alberto Mendelzon. WebOQL: Restructuring documents, databases and webs. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, Orlando, Florida, 1998.

[AMM98] Paolo Atzeni, Giansalvatore Mecca, and Paolo Merialdo. Design and maintenance of data-intensive web sites. In *Proc. of the Conf. on Extending Database Technology (EDBT)*, Valencia, Spain, 1998.

[Ang82] D. Angluin. Inference of reversible languages. *Journal of the ACM*, 29(3):741-65, 1982.

[BEM+98] C. Beeri, G. Elber, T. Milo, Y. Sagiv, O.Shmueli, N.Tishby, Y.Kogan, D.Konopnicki, P. Mogilevski, and N.Slonim. Websuite-a tool suite for harnessing web data. In *Proceedings of the International Workshop on the Web and Databases*, Valencia, Spain, 1998.

[BLR97] Catriel Beeri, Alon Y. Levy, and Marie-Christine Rousset. Rewriting queries using views in description logics. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Tucson, Arizona., 1997.

[Bun97] Peter Buneman. Semistructured data. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 117–121, Tucson, Arizona, 1997.

[CCM+98] T. Catarci, S.K. Chang, M.Lenzerini, D. Nardi, and G. Santucci. Turning the web into a database: the WAG approach. In *Proceedings of HICSS*, 1998.

[CDF+98] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Sean Slattery. Learning to extract symbolic knowledge from the world-wide web. In *Proceedings of the AAAI Fifteenth National Conference on Artificial Intelligence*, 1998.

[CDSS98] Sophie Cluet, Claude Delobel, Jerome Simeon, and Katarzyna Smaga. Your mediators need data conversion. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Seattle, WA, 1998.

[CGL98] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. What can knowledge representation do for semi-structured data? In *Proceedings of the National Conference on Artificial Intelligence*, 1998.

[CKPS95] Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, and Kyuseok Shim. Optimizing queries with materialized views. In *Proc. of Int. Conf. on Data Engineering (ICDE)*, Taipei, Taiwan, 1995.

[CL93] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, 1993.

[Coh98a] W. Cohen. A web-based information system that reasons with structured collections of text. In *Proc. Second Intl. Conf. Autonomous Agents*, pages 400–407, 1998.

[Coh98b] William Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Seattle, WA, 1998.

[DEW97] B. Doorenbos, O. Etzioni, and D. Weld. Scalable comparison-shopping agent for the world-wide web. In *Proceedings of the International Conference on Autonomous Agents*, February 1997.

[DG97a] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, Tucson, Arizona., 1997.

[DG97b] Oliver M. Duschka and Michael R. Genesereth. Query planning in infomaster. In *Proceedings of the ACM Symposium on Applied Computing*, San Jose, CA, 1997.

[DL97] Oliver M. Duschka and Alon Y. Levy. Recursive plans for information gathering. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.

[Dus97] Oliver Duschka. Query optimization using local completeness. In *Proceedings of the AAAI Fourteenth National Conference on Artificial Intelligence*, 1997.

[EGW94] Oren Etzioni, Keith Golden, and Daniel Weld. Tractable closed world reasoning with updates. In *Proceedings of the Conference on Principles of Knowledge Representation and Reasoning, KR-94.*, 1994. Extended version to appear in *Artificial Intelligence.*

[FFK+98] Mary Fernandez, Daniela Florescu, Jaewoo Kang, Alon Levy, and Dan Suciu. Catching the boat with Strudel: Experiences with a web-site management system. In *Proc. of ACM SIGMOD Conf. on Management of Data*, Seattle, WA, 1998.

[FFLS98] Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. Reasoning about web-sites. In *Working notes of the AAAI-98 Workshop on Artificial Intelligence and Data Integration. American Association of Artificial Intelligence.*, 1998.

[FKL97] Daniela Florescu, Daphne Koller, and Alon Levy. Using probabilistic information in data integration. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, pages 216–225, Athens, Greece, 1997.

[FRV96] Daniela Florescu, Louiqa Raschid, and Patrick Valduriez. A methodology for query reformulation in cis using semantic knowledge. *Int. Journal of Intelligent & Cooperative Information Systems, special issue on Formal Methods in Cooperative Information Systems*, 5(4), 1996.

[FW97] M. Friedman and D. Weld. Efficient execution of information gathering plans. In *Proceedings of the International Joint Conference on Artificial Intelligence, Nagoya, Japan*, 1997.

[GMPQ+97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. *Journal of Intelligent Information Systems*, 8(2):117–132, March 1997.

[GRVB98] Jean-Robert Gruser, Louiqa Raschid, María Esther Vidal, and Laura Bright. Wrapper generation for web accessible data sources. In *Proceedings of the CoopIS*, 1998.

[HGMN+98] Joachim Hammer, Hector Garcia-Molina, Svetlozar Nestorov, Ramana Yerneni, Markus M. Breunig, and Vasilis Vassalos. Template-based wrappers in the TSIMMIS system (system demonstration). In *Proc. of ACM SIGMOD Conf. on Management of Data*, Tucson, Arizona, 1998.

[HKWY97] Laura Haas, Donald Kossmann, Edward Wimmers, and Jun Yang. Optimizing queries across diverse data sources. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Athens, Greece, 1997.

[IFF+98] Zachary Ives, Daniela Florescu, Marc Friedman, Alon Levy, and Dan Weld. An adaptive query execution engine for data integration. submitted for publication, 1998.

[JB97] R. Jakobovits and J. F. Brinkley. Managing medical research data with a web-interfacing repository manager. In *American Medical Informatics Association Fall Symposium*, pages 454–458, Nashville, Oct 1997.

[KDW97] N. Kushmerick, R. Doorenbos, and D. Weld. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.

[Kno95] Craig A. Knoblock. Planning executing, sensing and replanning for information gathering. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1995.

[KW96] Chung T. Kwok and Daniel S. Weld. Planning to gather information. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, 1996.

[LB87] Hector J. Levesque and Ronald J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[Lev96] Alon Y. Levy. Obtaining complete answers from incomplete databases. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Bombay, India, 1996.

[LK98] E. Lambrecht and S. Kambhampati. Optimization strategies for information gathering plans. TR-98-018, Arizona State University Department of Computer Science, 1998.

[LMR90] Witold Litwin, Leo Mark, and Nick Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22 (3):267–293, 1990.

[LMSS95] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, San Jose, CA, 1995.

[LR82] T. Landers and R. Rosenberg. An overview of multibase. In *Proceedings of the Second International Symoposium on Distributed Databases*, pages 153–183. North Holland, Amsterdam, 1982.

[LR96] Alon Y. Levy and Marie-Christine Rousset. Verification of knowledge bases using containment checking. In *In Proceedings of AAAI*, 1996.

[LR98] Veronique Lattes and Marie-Christine Rousset. The use of the CARIN language and algorithms for information integration: the PICSEL project. In *Proceedings of the ECAI-98 Workshop on Intelligent Information Integration*, 1998.

[LRO96a] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Query answering algorithms for information agents. In *Proceedings of AAAI*, 1996.

[LRO96b] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Bombay, India, 1996.

[LS98] Gerald Lohse and Peter Spiller. Electronic shopping. *Comm. of the ACM*, 41(7), July 1998.

[PAGM96] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Bombay, India, 1996.

[PE95] Mike Perkowitz and Oren Etzioni. Category translation: Learning to understand information on the internet. In *Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments*. American Association for Artificial Intelligence., 1995.

[PE97] Mike Perkowitz and Oren Etzioni. Adaptive web sites: an AI challenge. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.

[PF98] P. Paolini and P. Fraternali. A conceptual model and a tool environment for developing more scalable, dynamic, and customizable web applications. In *Proc. of the Conf. on Extending Database Technology (EDBT)*, Valencia, Spain, 1998.

[PGGMU95] Yannis Papakonstantinou, Ashish Gupta, Hector Garcia-Molina, and Jeffrey Ullman. A query translation scheme for rapid implementation of wrappers. In *Proc. of the Int. Conf. on Deductive and Object-Oriented Databases (DOOD)*, 1995.

[Rei88] Raymond Reiter. Towards a logical reconstruction of relational database theory. In John Mylopoulos and Michael Brodie, editors, *Readings in Artificial Intelligence and Databases*, pages 301-326. Morgan Kaufmann, Los Altos, CA, 1988.

[RSU95] Anand Rajaraman, Yehoshua Sagiv, and Jeffrey D. Ullman. Answering queries using templates with binding patterns. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, San Jose, CA, 1995.

[SS97] James Schmolze and Wayne Snyder. Detecting redundant production rules. In *Proceedings of the National Conference on Artificial Intelligence*, 1997.

[TN98] Motomichi Toyama and T. Nagafuji. Dynamic and structured presentation of database contents on the web. In *Proc. of the Conf. on Extending Database Technology (EDBT)*, Valencia, Spain, 1998.

[TRV98] A. Tomasic, L. Raschid, and P. Valduriez. Scaling access to distributed heterogeneous data sources with Disco. IEEE Transactions On Knowledge and Data Engineering *(to appear)*, 1998.

[TSI96] Odysseas G. Tsatalos, Marvin H. Solomon, and Yannis E. Ioannidis. The GMAP: A versatile tool for physical data independence. *VLDB Journal*, 5(2):101-118, 1996.

[UFA98] Tolga Urhan, Michael J. Franklin, and Laurent Amsaleg. Cost based query scrambling for initial delays. In *Proc. of ACM SIGMOD Conf. on Management of Data*, pages 130-141, Seattle, WA, 1998.

[VP97] Vasilis Vassalos and Yannis Papakonstantinou. Describing and using the query capabilities of heterogeneous sources. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, Athens, Greece, 1997.

[YL87] H. Z. Yang and P. A. Larson. Query transformation for PSJ-queries. In *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, pages 245-254, Brighton, England, 1987.