Data Integration: A Logic-Based Perspective

Diego Calvanese
Faculty of Computer Science
Free University of Bolzano/Bozen
Piazza Domenicani 3, 39100 Bolzano, Italy
calvanese@inf.unibz.it

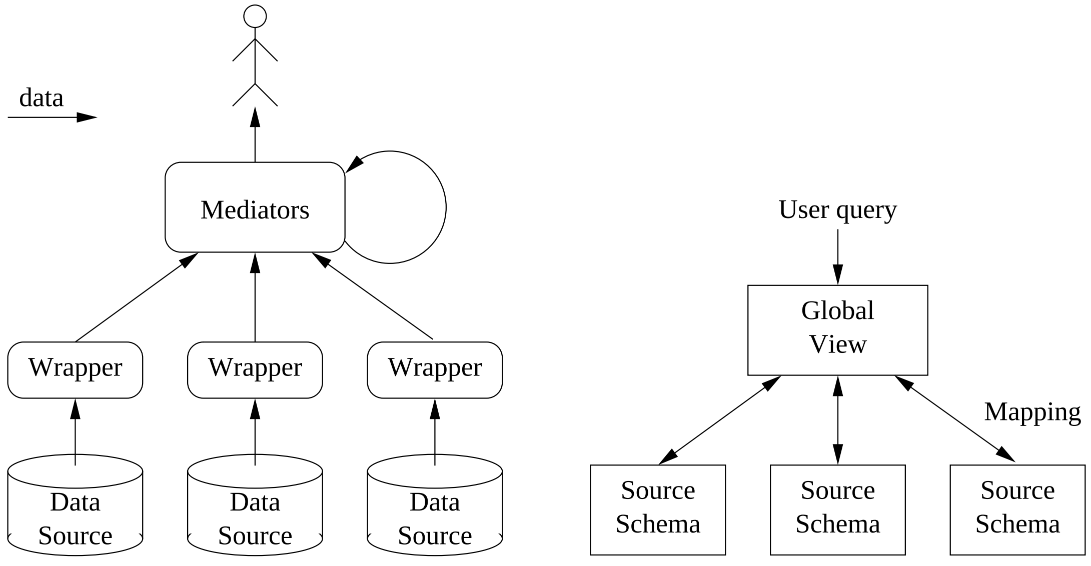
Giuseppe De Giacomo
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
degiacomo@dis.uniroma1.it

Abstract

Data integration is the problem of combining data residing at different autonomous, heterogeneous sources, and providing the client with a unified, reconciled global view of these data. We discuss data integration systems, taking the abstract viewpoint that the global view is an ontology expressed in a class-based formalism. We resort to an expressive Description Logic, $\mathcal{ALCQI}$, that fully captures class-based representation formalisms, and show that query answering in data integration, as well as all other relevant reasoning tasks, is decidable. However, the high computational complexity in the size of the data makes the use of a full-fledged expressive Description Logic infeasible in practice, when we have to deal with large amounts of data. This leads us to consider *DL-Lite*, a specifically tailored restriction of $\mathcal{ALCQI}$, that ensures tractability of query answering in data integration, while keeping enough expressive power to capture the most relevant features of class-based formalisms.

1 Introduction

Data integration is the problem of combining data residing at different autonomous, heterogeneous sources, and providing the client with a unified, reconciled view of these data. The typical architecture of a data integration system in depicted in Figure 1(a). In such a system, the actual data resides in a set of data sources. The user, however, does not access such data sources directly, but poses its queries to the integration system, and is thus freed from the necessity to know where the actual data reside and how to access the data



(a) Concrete architecture

(b) Abstract architecture

Figure 1: Architecture of a data integration system

sources to extract it. It is the task of the integration system to decide which sources are relevant for answering the user query, to distribute the query over such sources, to collect the returned answers, to combine and reconcile them, and to present the overall answer to the user. These tasks are typically accomplished by two types of software modules called *wrappers* and *mediators*. Wrappers are responsible for directly accessing the sources and returning the data therein in a unified form (e.g, as sets of tuples conforming to a relational schema). Mediators are responsible for combining the data coming from wrappers or other mediators and presenting them according to a specified structure (e.g., a relational schema with certain attributes). The problem of setting up data integration systems, and specifically wrappers and mediators, is becoming increasingly important, especially in enterprise applications, and is characterized by a number of issues that are interesting, both from a theoretical and from a practical point of view.

Most of the current work on data integration in databases [33, 39, 31, 35] takes a declarative approach to the problem. It assumes that a data integration system is characterized by giving explicitly to the client a global, virtual, reconciled and unified view of the data. The virtual concepts are mapped to the concrete data sources, where the actual data resides, through explicit mapping assertions. Thus, the user formulates its queries in terms of the global view, and the system decides how to exploit the mappings in order to reformulate the user query in terms of the data sources. The abstract architecture corresponding to such an approach is depicted in Figure 1(b). It maps to the concrete architecture in Figure 1(a) by considering that the mediators implement the query reformulation process and the actual execution of the reformulated query. Also, in this abstract view, we do not deal with the issues related to wrapping the sources, and assume that all sources are represented through their schema in a uniform data model, specifically, the relational model.

Different approaches for specifying mappings in a data integration system have been proposed [31, 35]. In the *global-as-view* (GAV) approach, each concept of the global view is mapped to a query over the data sources. In other words, it is assumed that the data corresponding to a concept of the global view, which the user expects to obtain when she formulates her queries, can actually be retrieved from the data sources through a specific query, specified in a certain query language (e.g., select-project-join queries in SQL). In this way, query processing is conceptually easy, since it amounts to just replacing (or *unfolding*) each global concept in the user query with the associated query over the sources, and then executing the unfolded query over the sources[1]. However, the approach does not cope well with dynamicity and changes in the sources, since such changes potentially affect all mappings and require to restructure the global view. In contrast, in the *local-as-view* (LAV) approach, each concept in the data sources is defined in terms of a query over the global view. In other words, the information content of the sources is described in terms of the global view, i.e., in terms of those concepts that are familiar to the user and in terms of which the user accesses the system. This complicates query processing, since now the system is not told explicitly how to reformulate the concepts in the global view mentioned in the user query in terms of the data sources. On the other hand, changes in the sources require only to change the associated mappings and have no impact on the global view. A generalized approach, where a mapping assertion relates a query over the global view to a query over the sources, called GLAV has also been considered [28].

The loose coupling between data sources and global view by means of the mappings, results in having incomplete information on the extensions of the concepts of the global view. In other words, if we fix the actual data at the sources, there are in general many possible ways to get the extension of the concepts of the global view that are compatible with the data at the sources and with the mapping. Hence, when answering queries posed over the global view, such incompleteness must be taken into account. This results in being interested in computing the *certain answers* [31, 35], i.e., those answers that hold in all extensions of the global view that are compatible with the provided information.

More recently, the work on data integration in databases has started to consider also constraints expressed over the concepts of the global view, ranging from keys and foreign keys to more complex forms of assertions expressible in semantic data models, such as the Entity-Relationship model or UML class diagrams. Such constraints help to better capture the complex interrelationships in the domain of interest. However, they have a deep impact on how certain answers are computed, and hence they must be fully taken into account during query answering [9, 10, 13]. We observe that, once we allow for constraints on the global view, the differences between the various approaches for establishing mappings become blurred, since, with the help of constraints, one can mimic one approach in the other [12].

If we take an AI point of view, we can consider the whole integration system, constituted

[1] Of course, system level problems still remain to be addressed, such as how to distribute the query over the sources, how to collect and combine the answers, etc., but we are not concerned with these aspects here.

by the global view (with constraints), the data sources, and the mapping, as a knowledge base. In such a knowledge base, knowledge about specific data items (i.e., *extensional knowledge*), and knowledge about how the information of interest is organized (i.e., *intensional knowledge*) are clearly separated: extensional knowledge is constituted by the data sources, while intensional knowledge is formed by the global view and the mapping. Under this view, computing certain answers essentially corresponds to logical inference: the certain answers are those data for which the data present in the sources and the information on the global view and the mapping logically implies that they are in the answer to the query.

Building on the above considerations, the ultimate realization of a global view is an ontology which the client can access and which gives her a semantically rich framework in which to understand the information gathered by the system. The mappings relate such an ontology to the data sources used to retrieve the extensional information. In this paper we take this perspective, and consider an integration system as formed by a (global) ontology, a set of data sources, and mappings between the two. In particular, we discuss in a certain detail ontologies that are expressed in terms of classes and relationships between classes. Such an approach stems from representation formalisms developed in various areas, ranging from Entity-Relationship diagrams in databases [4], UML class diagrams in software engineering [2], and ontology languages for the Semantic Web, e.g., OWL-DL[3].

Specifically, we start by introducing a general formal framework for describing information integration systems based on an ontology for the global view (Section 2). Then (Section 3), we look at systems whose ontology is expressed in terms of an expressive Description Logics, namely $\mathcal{ALCQI}$ [2], which is a Description Logic that fully captures class-based representation formalism [26, 11] and that is at the base of the current proposals for standard ontology languages. Notably, although we are using a full-fledged class-based language, all reasoning tasks, including computing certain answers in integration systems, are decidable. However, in information integration systems, since we typically deal with large amounts of data, it is crucial not only to be decidable, but to remain tractable in the size of data. Unfortunately, this is not the case for $\mathcal{ALCQI}$ nor for any representation formalism that aims at fully capturing class based modeling. This leads us to consider (Section 4) a specifically tailored restriction of $\mathcal{ALCQI}$, which we call *DL-Lite*, that, on the one hand provides enough expressive power to capture the most relevant features of class-based formalisms, and on the other hand ensures tractability with respect to the size of the data. We conclude the paper by discussing further research directions (Section 5).

2 General Framework for Semantic Integration

In this section we present a general formal framework for semantic integration systems. Following the standard approach in information integration, we will refer to integration systems whose components are the following: (*i*) a set of data sources, containing the actual information users are interested in; (*ii*) a global ontology, which provides a reconciled, integrated,

[2] http://www.omg.org/uml/

[3] http://www.w3.org/2001/sw/WebOnt/

and virtual view of the underlying sources and in terms of which users access the system; and (*iii*) the mapping between the two, which is used to relate the information in the sources to the concepts in the global ontology.

In what follows, one of the main aspects is the definition of the semantics of both the Integration System, and of queries posed to the global ontology. For keeping things simple, we will use in the following a unique semantic domain Δ, constituted by a fixed, infinite set of symbols. We also assume to have a fixed set of constants, and we fix the interpretation of such constants so that: (*i*) each constant denotes an element in Δ; (*ii*) different constants denote different elements of Δ; (*iii*) each element in Δ is denoted by a constant.[4] In the following, with some abuse of notation, we will not distinguish between constants and the domain elements they denote.

Formally, an *Ontology-Based Integration System* (OIS) $\mathcal{O}$ is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}$ is the global ontology, $\mathcal{S}$ is the set of data sources, and $\mathcal{M}$ is the mapping between $\mathcal{G}$ and $\mathcal{S}$.

- We assume that the global ontology $\mathcal{G}$ of an OIS is expressed as a theory (named simply $\mathcal{G}$) in some logic (e.g., first-order logic) involving a set of predicates interpreted over Δ.

- We assume to have a set $\mathcal{S}$ of n data sources $S_1, \ldots, S_n$, each one consisting of extensions of predicates over Δ. We assume that the (predicate) alphabets of the various data sources are mutually disjoint, and that each one is disjoint from the alphabet of the global ontology. For simplicity of exposition, without loss of generality, we assume that each source is constituted by the extension of a single predicate.

- The mapping $\mathcal{M}$ is the heart of the OIS, in that it specifies how the predicates in the global ontology $\mathcal{G}$ and in the data sources $\mathcal{S}$ are mapped to each other. In particular, such mappings are established by relating open formulas (i.e., queries) over the global ontology to open formulas over the data sources.

Notice that we have assumed that data sources are seen as databases. In turn, such sources may be complex ontologies, thus containing dependencies and interrelationships among their various concepts at the intensional level. However, we consider the setting where such data sources are completely autonomous, and hence may not conform to the global ontology that the clients of an OIS can access. Neither we want to integrate their intensional knowledge into the global ontology seen by the client. We want just to take into account how the data at the sources is used to feed the predicate extension of the global ontology. Intuitively, in specifying the semantics of an OIS, we have to start with an extension of the data sources, called the *source database*, and the crucial point is to determine which are the models of the global ontology that correspond to such a source database. In doing so, both the constraints specified in the global ontology and the mapping are taken into account. More precisely, the semantics of an OIS is defined as the set of all models of the global ontology that satisfy the mapping with respect to the source database. What it

[4] In other words, such constants act as *standard names* [36].

means to satisfy a mapping depends on the form of the mapping and will be discussed in Section 3.2 (see also [19]).

Queries posed to an OIS $\mathcal{O}$ are expressed in terms of a certain query language over the alphabet of the global ontology, and are intended to extract a set of tuples of elements of the semantic domain Δ. In accordance with what is typical in databases, we require that each query has an associated arity, and that it extracts only tuples of that arity. Given a source database for $\mathcal{O}$, the tuples we are interested in are those that are guaranteed to be in the answer of the query for every model for $\mathcal{O}$ with respect to the source database. In other words, we are interested in *certain answers*.

One of the most common ways to express knowledge on a domain of interest is to resort to class-based formalisms, in which knowledge is represented in terms of objects grouped into classes and relationships between classes. Examples are Entity-Relationship diagrams in Databases, UML class diagrams in Software Engineering, and ontology languages for the Semantic Web such as OWL-DL. All such formalisms can be captured in a fragment of first-order logic in which one can express inclusions and/or equivalences between classes, and possibly pose additional constraints on the relations between classes. Such fragments correspond to a class of logics, called Description Logics [2] (see Section 3).

On the other hand, for the mapping, which represents the heart of an OIS, it is in general not sufficient to limit the expressive power to direct correspondences between classes, since this does not allow one to capture the complex interrelations that may exist between the data in the sources and the (virtual) data in the global view. In a real-world setting, one needs a much more powerful mechanism for establishing mappings between the sources $\mathcal{S}$ and the global view $\mathcal{G}$. Specifically, one would like, on the one hand, to acquire the relevant information to be extracted from $\mathcal{S}$ by navigating and aggregating several concepts, and on the other hand, to characterize these data in terms of the elements of $\mathcal{G}$ as precisely as possible. To achieve this, it is necessary to resort to mappings that relate to each other a *query* Q_s over $\mathcal{S}$ and a *query* Q_g over $\mathcal{G}$, both expressed in an appropriate query language. As common in data integration, we assume mappings to be *sound*, i.e., the data extracted from the sources via Q_s are in general only a subset of those satisfying the corresponding query Q_g in the global models for $\mathcal{O}$ with respect to a source database.

3 Semantic Integration Using Description Logics

The considerations made in the previous section lead us to provide a formalization of an OIS which is based on the use of Description Logics to represent ontologies [20, 21]. *Description Logics* (DLs) [2] are knowledge representation formalisms that are able to capture the core features of virtually all class-based representation formalisms used in Artificial Intelligence, Software Engineering, and Databases [25, 26]. Recently, DLs are gaining an increased popularity as the formalisms that provide the theoretical foundation for the languages becoming standard for the Semantic Web, specifically OWL-DL. One of the distinguishing features of these logics is that they are equipped with optimal reasoning algorithms, and practical systems implementing such algorithms are now available [32, 29, 38].

In the following, we first introduce a specific DL and then we illustrate how we use such a logic to define an OIS.

3.1 The Description Logic $\mathcal{ALCQI}$

In DLs, the domain of interest is modeled by means of *concepts* and *roles*. Concepts are unary predicates, which denote classes of objects called *instances* of the concept. Roles instead are binary predicates, which denote binary relationships between objects. The simplest forms of concepts and roles are atomic concepts and roles, which are constituted just by a name. Each DL is then equipped with a set of specific constructs that allow one to obtain, starting from atomic concepts and roles, complex concept expressions (or simply concepts). Each construct has a precise set-theoretic semantics and therefore the meaning of complex concepts is determined on the basis of the meaning of its constituents and the constructs combining them. Similarly, a DL may be equipped with construct for obtaining complex role expressions (or simply roles).

We focus our attention on a specific DL, $\mathcal{ALCQI}$, which belongs to the well-studied family of $\mathcal{AL}$-languages [2]. $\mathcal{ALCQI}$ is a notable example for an expressive DL, featuring constructs that are typical of conceptual modeling formalisms, and that in fact allow $\mathcal{ALCQI}$ to capture the most important features of such formalisms [6, 5]. Here, we do not provide a formal presentation of $\mathcal{ALCQI}$; instead we introduce its constructs by means of examples. Also, instead of the abstract notation typical of the DL literature (cf. [2]), we make use of a more verbose, textual notation, that is however easier to understand for readers not familiar with the DL syntax.

The $\mathcal{ALCQI}$ DL provides concept constructs for complement, intersection, union, existential restriction, universal quantification, and number restrictions. As for roles, it provides the construct for inverse roles. Recall that roles denote binary relations between objects; in the following we say that an object o_1 is *connected* to another object o_2 via a role R, meaning that the pair (o_1, o_2) is in the relation represented by R. We discuss now the various constructs in more detail. *Complement*, *intersection*, and *union* denote simply the corresponding set operations on the sets of instances of the involved concepts. Existential restriction and universal quantification represent restricted forms of existential and universal quantification, respectively. More precisely, through *existential restriction* on a role R one can denote all those objects connected through R to at least one instance of a concept C. For example, (Staff and (teaches some Course)) denotes those individuals that are staff members and that teach some course. The dual construct, *universal quantification* on a role R, denotes objects that are connected through R only to instances of a concept C. For example, (teaches only UGCourse) denotes those individuals that teach only undergraduate courses. Also, through *number restrictions* on a role R one can express restrictions on the minimum and maximum number of connections via R that an object may have to instances of a concept C. Thus, number restrictions represent a generalization of existential, functionality, and multiplicity constraints in data models. For example, (teaches at-most 3 Course) denotes those individuals that teach at most three courses. Finally, through an *inverse role* (inverse R) one can denote the inverse of the relationship denoted by a role R. For example, (Course and ((inverse teaches) some Postdoc) denotes all those courses that are taught by a postgraduate. This is done by referring to the role teaches, whose inverse is the taught-by relation.

In $\mathcal{ALCQI}$, a *knowledge base* is constituted by two components, a *TBox*, used to express intensional knowledge, and an *ABox*, used to express extensional knowledge. Specifically, a TBox is constituted by a set of *inclusion assertions*, each of the form (C_1 is-a C_2), where C_1 and C_2 are two arbitrary $\mathcal{ALCQI}$ concepts. Such an inclusion assertion states a subclass-superclass relationship in which C_1 is the subclass and C_2 is the superclass. For example, ((Staff and (teaches some Course)) is-a Busy) expresses that each staff member teaching a course is busy. There is no restriction on the set of assertions that may constitute a TBox, and, in particular, they may involve cycles.

The ABox of an $\mathcal{ALCQI}$ knowledge base is constituted by a set of *membership assertions* involving concepts or roles, of the form $C(z)$ and $R(z_1, z_2)$, stating respectively that the object z is an instance of the concept C and that the pair of objects (z_1, z_2) is an instance of the role R. For example, Staff(ann), Course(ai), teaches(ann, ai) express respectively that ann is a staff member, that ai is a course, and that ann teaches ai.

Being logics, DLs in general and $\mathcal{ALCQI}$ in particular are equipped with a formal semantics and with reasoning services defined in accordance with the semantics. The basic reasoning services over DL knowledge bases are *knowledge base satisfiability*, i.e., determining whether a knowledge base can be populated without violating any of the inclusion or membership assertions; *concept satisfiability* with respect to a knowledge base, i.e., determining whether it is possible to populate a knowledge base in such a way that a given concept is populated with at least one instance; and *logical implication*, i.e., determining whether a given TBox or ABox assertion necessarily holds whenever all assertions in a given knowledge base hold.

Finally, we introduce the notion of query in $\mathcal{ALCQI}$. We remind that the answer to a query, when the query is evaluated over a knowledge base, is a set of tuples of objects. The types of queries we consider are conjunctive queries, which correspond to SQL select-project-join queries, but have a notation that is more convenient for formal manipulations. A *conjunctive query* over an $\mathcal{ALCQI}$ knowledge base is a conjunction of atoms, where each atom involves a predicate applied to a variable or a constant. Each predicate is either an atomic concept (hence, a unary predicate) or an atomic role (hence, a binary predicate), which may also freely be used in the assertions of the knowledge base. When evaluating the query, the constants denote specific domain objects, while the variables are instantiated on the domain objects, in accordance with the predicates in which they appear. For example, the variable x in the atom UGCourse(x) may be instantiated only on undergraduate courses. Each variable may be either free or existentially quantified. The free variables (also called *distinguished variables*) denote the components of the tuples that are in the answer to the query. Existentially quantified variables, instead, are used to relate to each other the various atoms in the query, but they do not directly contribute to the answer to the query. For example, the conjunctive query $\{\, x, y \mid \mathsf{Staff}(x) \land \mathsf{Staff}(y) \land \mathsf{teaches}(x, z) \land \mathsf{teaches}(y, z) \land \mathsf{UGCourse}(z) \,\}$ denotes all pairs of staff members that have at least one undergraduate course they teach in common. The distinguished variables are x and y, while z is an existentially quantified variable that stands for the commonly taught undergraduate course (notice that the existential quantifier on z is not explicitly present in the query, but is implicit in its semantics).

The basic reasoning services that are of interest in the presence of queries are query-answering and query-containment. *Query answering* consists in determining all tuples of objects that are in the answer to the query, whenever all assertions of the knowledge base are satisfied. Observe that, as a special case of query answering, we have concept satisfiability and logical implication of ABox assertions. *Query containment* consists in determining, given a knowledge base and two queries of the same arity, whether the answer to one query is contained in the answer to the other one, whenever all assertions of the knowledge base are satisfied. As a special case of query containment we have logical implication of inclusion assertions involving atomic concepts on both sides. In fact, it can also be shown that query containment can be reformulated as query answering [1].

$\mathcal{ALCQI}$ is equipped with effective reasoning techniques that are sound and complete with respect to the semantics. In particular, all reasoning tasks involving a knowledge base only (and not queries) are EXPTIME-complete. Instead, checking query containment (and hence also query answering) is EXPTIME-hard and solvable in 2EXPTIME in the size of the knowledge base [17]. Note that such an exponential bound depends also on the size of the data (i.e., the ABox).

3.2 OIS Based on $\mathcal{ALCQI}$

We now set up a framework for ontology integration, which extends ideas developed for data integration over DL knowledge bases [20, 18]. In particular, we describe the main components of the ontology integration system, and we provide the semantics both of the system and of query answering.

In this setting, an OIS $O = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ is defined as follows:

- The *global ontology* $\mathcal{G}$ is an $\mathcal{ALCQI}$ knowledge base constituted by a TBox only. Notice that, in accordance with what discussed in Section 2, this means that only intensional knowledge (i.e, describing how the information is organized) and no extensional knowledge (i.e., about specific data items) can be maintained by such a global ontology.

- Each *data source* in $\mathcal{S}$ is constituted simply by a relational alphabet, and by the extensions of the relations in such an alphabet. For example, such extensions may be expressed as relational databases. Observe that we are assuming that no intensional relation between terms is present in the local ontologies.

- The *mapping* $\mathcal{M}$ between $\mathcal{G}$ and $\mathcal{S}$ is given by a set of correspondences of the form $Q_s \rightarrow Q_g$, where Q_s is a conjunctive query over one of the data sources in $\mathcal{S}$, and Q_g is a conjunctive query over the global ontology $\mathcal{G}$. As mentioned, the mapping correspondences are assumed to be sound. This means that the correspondence $Q_s \rightarrow$

Q_g is satisfied whenever the data extracted from the sources via Q_s is a subset of (not necessarily equal to) the global data satisfying Q_g.

The form of mapping we have considered here is quite general, and represents a generalization of the types of mappings that have been considered in the literature on data integration [33, 31, 35]. Indeed, two basic approaches for defining such a mapping have been proposed:

- the *local-as-view* (LAV) approach, where each relation of the data sources in $\mathcal{S}$ is mapped to a query over the global ontology $\mathcal{G}$;
- the *global-as-view* (GAV) approach, where each concept of the global ontology $\mathcal{G}$ is mapped to a query over the data sources in $\mathcal{S}$.

The GAV approach has been traditionally considered simpler, since, in order to answer a query over the global ontology, it is sufficient to unfold all concepts referenced in the query with their definition in terms of the data sources specified in the mapping. However, in the presence of intensional constraints in the global ontology this is in general not sufficient any more, and query answering becomes more involved [9, 10, 13].

Many authors point out that, despite its difficulty, the LAV approach better supports a dynamic environment, where data sources can be added to the system without the need for restructuring the global ontology. Hence, recent research work on data integration has followed this approach [39, 31, 37, 20, 24]. The major challenge in this case is that, in order to answer a query expressed over the global ontology, one must be able to reformulate the query in terms of queries to the sources. While in the GAV approach such a reformulation is guided by the correspondences in the mapping, in LAV the problem requires a reasoning step, so as to infer how to use the sources for answering the query.

The type of mapping we have considered here, called GLAV [28], combines the flexibility of the LAV and GAV approaches, by allowing one to establish directly mappings between two queries. We will see below that, also in our setting, the added expressive power provided by GLAV mappings does not add complexity to the techniques that have already to be adopted to handle LAV mappings.

Query answering in this setting requires quite sophisticated techniques. Indeed, in order to answer a query posed over the global ontology with the data contained in the local ontologies, one has to take into account the knowledge both in the global ontology and in the mapping. Such query answering techniques are studied in [18] for the case of LAV, and are essentially based on encoding the data extracted through the mappings into an ABox. The techniques can be applied to GLAV mappings as well, by observing that a GLAV mapping $Q_s \rightarrow Q_g$ can be rephrased by introducing a new source relation R of the same arity as the queries Q_s and Q_g in the mapping. The extension of the new source R is given by evaluating Q_s on the data sources, and R is then mapped to the global ontology through a LAV mapping $\{x_1, \ldots, x_n \mid R(x_1, \ldots, x_n)\} \rightarrow Q_g$.

Example 1 Consider for example the OIS $\mathcal{O}_d = \langle \mathcal{G}_d, \mathcal{S}_d, \mathcal{M}_d \rangle$ defined as follows:

- The global ontology $\mathcal{G}_d$ is the $\mathcal{ALCQI}$ knowledge base

$$\begin{array}{rcl} \mathsf{Postdoc} & \texttt{is-a} & \mathsf{Staff} \\ \mathsf{UGCourse} & \texttt{is-a} & \mathsf{Course} \\ \mathsf{Staff}\ \texttt{and}\ (\mathsf{teaches}\ \texttt{some}\ \mathsf{Course}) & \texttt{is-a} & \mathsf{Busy} \end{array}$$

  expressing that each postdoc is a staff member, that each undergraduate course is a course, and that each staff member who teaches a course is busy.

- The set $\mathcal{S}_d$ of data sources consists of two data sources, containing respectively the unary relations T_1 and T_2.

- The mapping $\mathcal{M}_d$ is

$$\begin{array}{lcl} \{\, x \mid \mathsf{T}_1(x) \,\} & \rightarrow & \{\, x \mid \mathsf{Staff}(x) \wedge \mathsf{teaches}(x,y) \wedge \mathsf{UGCourse}(y) \,\} \\ \{\, x \mid \mathsf{T}_2(x) \,\} & \rightarrow & \{\, x \mid \mathsf{Postdoc}(x) \,\} \end{array}$$

  i.e., it is constituted by two LAV correspondences, associating to each source relation a conjunctive query over the global ontology. The first correspondence expresses that source relation T_1 contains a set of staff members teaching an undergraduate course. Notice that, since the mapping is sound, there may be also other staff members teaching an undergraduate course besides those listed in T_1. Similarly, the second correspondence expresses that source relation T_2 contains postgraduate students.

Consider the conjunctive query $\mathsf{Q}_w = \{x \mid \mathsf{Postdoc}(x) \wedge \mathsf{Busy}(x)\}$ over $\mathcal{G}_d$, asking for the postdocs who are busy. Given the source database $\mathcal{D}$ in which T_1 has extension $\{\mathsf{ann}, \mathsf{bill}\}$, and T_2 has extension $\{\mathsf{ann}, \mathsf{dan}\}$, we have that the certain answer is constituted by the single tuple ann. Observe that, to obtain such an answer, we have to reason using the mappings and the inclusions in the global ontology. ■

4 Simplifying reasoning tasks

One of the most important lines of research in DLs is concerned with the trade-off between expressive power and computational complexity of sound and complete reasoning. Research on this topic has shown that DLs with efficient, i.e., worst-case polynomial time, reasoning algorithms lack modeling power required in capturing conceptual models and basic ontology languages, while DLs with sufficient modeling power, such as $\mathcal{ALCQI}$, suffer from inherently worst-case exponential time behavior of reasoning [25, 26, 8]. This is reflected also when addressing ontology-based integration, where the inherently high computational complexity of the underlying DL has a bad impact on the computational complexity of query answering, and makes it infeasible in practice.

In this section we introduce a DL called *DL-Lite* [23, 16] to be used as the formalism underlying an ontology-based integration system. Such a DL provides a very good trade-off between expressive power and complexity of reasoning, both over a knowledge base and over queries. On the one hand it has sufficient expressive power, being specifically tailored to capture the fundamental aspects of conceptual data models (e.g., Entity-Relationship diagrams) [4], object-oriented formalisms (e.g., basic UML class diagrams), and basic ontology languages such as OWL-DL. On the other hand, it admits advanced forms of sound and complete reasoning, which take into account both a knowledge base (constituted by TBox and ABox), and queries, and which are polynomial time in the size of the knowledge base, including the data.

4.1 *DL-Lite*

DL-Lite is a DL that is quite simple from the language point of view. The constructs it provides are complement and intersection of concepts (but no union), simplified forms of existential restriction and number restrictions, and inverse roles (recall that roles denote binary relations). Moreover, the concept constructs may not be combined freely, but need to respect certain syntactic conditions. Namely, starting again from atomic concepts and atomic roles, we define *basic concepts* as either an atomic concept or an *unqualified existential restriction* [3]. Such a construct denotes all objects that are connected through a role R to some other object o. In an existential restriction in $\mathcal{ALCQI}$, we specify the concept that this object o must be an instance of. In *DL-Lite*, we just say that such an object exists, but do not further qualify it. For example, the concept (`teaches something`) denotes those objects that teach something, without further qualifying what is taught. General concepts in *DL-Lite* are then conjunctions of basic concepts and their complements. Note that, in *DL-Lite*, the use of complement is restricted to basic concepts only, and that we cannot express union of concepts. As for roles, *DL-Lite* has a construct for *inverse roles*, similarly to $\mathcal{ALCQI}$.

Using this simple language, in a *DL-Lite* knowledge base we allow to make assertions of specific forms only. Specifically, in a *DL-Lite* TBox, we allow for *inclusion assertions* of the form (B `is-a` C) where on the left-hand-side we must have a basic concept, while on the right-hand-side we may have an arbitrary *DL-Lite* concept. Observe that, as in $\mathcal{ALCQI}$, we do allow for cyclic assertions. Indeed, we can enforce the cyclic propagation through the role P of the property of belonging to concept A using the two *DL-Lite* inclusion assertions (A `is-a` (P `something`)) and (((`inverse` P) `something`) `is-a` A). The first assertion states that all objects in A are connected through role P to some object. The second assertion states that all objects to which role P connects are in A. Hence, if we start by considering an object o_1 in A, then o_1 must be connected through P to some object o_2, which itself must be in A, and hence connected trough P to some object o_3, etc. This could go on forever, or we could close the “chain” by connecting a certain o_i to one of the previous objects, e.g., o_1.

Also, in addition to the above inclusion assertions, in *DL-Lite* we have a specific form of inclusion assertions that make use of number restrictions. Such assertions, called *functionality assertions* have the form (`functional` P) and (`functional` (`inverse` P)), and express, respectively, the functionality of an atomic role P and of the inverse (`inverse` P) of an atomic role. Functionality of P means that each object may be connected via P to at most one object, similarly for functionality of (`inverse` P). Note that, in contrast to $\mathcal{ALCQI}$, in *DL-Lite* functionality of a certain role can be expressed only as a global property, and not locally, i.e., for the instances of a certain concept. Thus, we are not allowed to assert, for instance, that postgraduates teach at most one course, while professors can teach an arbitrary number of courses. This would require to state that teaches is functional for the instances of Postdoc, while it is not so for the instances of Professor, and this is not possible in *DL-Lite*.

Finally, the ABox of a *DL-Lite* knowledge base has the same form as that of an $\mathcal{ALCQI}$ knowledge base, and is thus constituted by a set of membership assertions involving concepts and roles. Recall that the former have the form $C(z)$, where C is a concept and z is an individual, while the latter have the form $R(z_1, z_2)$, where R is a role and z_1, z_2 are individuals.

We consider queries in *DL-Lite* that have the same form as those in $\mathcal{ALCQI}$, and hence are conjunctive queries over a (*DL-Lite*) knowledge base.

4.2 Why *DL-Lite* is a "rich" DL

Although equipped with advanced reasoning services, at first sight *DL-Lite* seems to be rather weak in modeling intensional knowledge, and hence of limited use in practice. In fact this is not the case. Despite the simplicity of its language and the specific form of inclusion assertions allowed, *DL-Lite* is able to capture the main notions (though not all, obviously) of conceptual modeling formalism used in databases and software engineering, such as Entity-Relationship and UML class diagrams. In particular, *DL-Lite* assertions allow us to specify the following important constructs (relying on database terminology, we use the terms class and relation to denote respectively an atomic concept and an atomic role):

- *ISA*, or *subclass-superclass relations*, using assertions of the form (A_1 `is-a` A_2), stating that the class A_1 is a subclass of the class A_2. For example, (UGCourse `is-a` Course) states that each undergraduate course is a course.

- *Class disjointness*, using assertions of the form (A_1 `is-a not` A_2), stating disjointness between the two classes A_1 and A_2. For example (Course `is-a not` Staff) states that courses and staff members are disjoint.

- *Role-typing*, using assertions of the form ((P `something`) `is-a` A_1) (resp., (((`inverse` P) `something`) `is-a` A_2), stating that the first (resp., second) component of the relation P is of type A_1 (resp., A_2)[5]. Notice that these kinds of assertion correspond to domain (resp., range) assertions. For example, ((teaches `something`) `is-a` Staff) types the domain of teaches to be a staff member, while (((`inverse` teaches) `something`) `is-a` Course) types the range of teaches to be a course.

[5] Observe that this has nothing to do with the qualified restriction (P `some` A) (resp., (P `only` A)), which are not used to type the role P but are used to select those objects that are the first component of P and that are related (through P) to some object (resp., only to objects) belonging to A.

- *Participation constraints*, using assertions of the form (A is-a (P something)) (resp., (A is-a ((inverse P) something))), stating that instances of class A participate to the relation P as the first (resp., second) component. For example, (Postdoc is-a (teaches something)) states that each postdoc has to teach something, while (UGCourse is-a ((inverse teaches) something)) states that undergraduate courses need to be taught by someone.

- *Non-participation constraints*, using assertions of the form (A is-a not (P something)) (resp., (A is-a not ((inverse P) something)), stating that instances of class A do not participate to the relation P as the first (resp., second) component. For example, (Student is-a not (teaches something)) states that a student cannot teach anything.

- *Functionality restrictions*, using assertions of the form (functional P) (resp., (functional (inverse P))), stating that an object can be the first (resp., second) component of the relation P at most once. For example, (functional (inverse teaches)) states that a course may be taught by at most one individual.

Notably two important modeling features of class-based formalisms, which can be captured by $\mathcal{ALCQI}$, are missing in *DL-Lite*:

- the ability of stating *covering constraints*, i.e., stating that each instance of a class must be an instance of (at least) one of its subclasses;

- the ability of stating subset constraints between relations.

Note that these features are present in full-fledged Entity-Relationship diagrams and UML class diagrams. They are missing in *DL-Lite* exactly to get the nice computational characteristics that we are after. Instead, observe that the limitation to binary roles only is not crucial. Indeed, it is possible to extend the reasoning techniques discussed below to n-ary relations without losing most nice computational properties.

Example 2 We re-express Example 1 in *DL-Lite*. The OIS $\mathcal{O}_d = \langle \mathcal{G}_d, \mathcal{S}_d, \mathcal{M}_d \rangle$ is defined as follows:

- The global ontology $\mathcal{G}_d$ is the *DL-Lite* knowledge base

$$\begin{array}{rcl}
\mathsf{Postdoc} & \mathtt{is\text{-}a} & \mathsf{Staff} \\
\mathsf{UGCourse} & \mathtt{is\text{-}a} & \mathsf{Course} \\
(\mathsf{teaches}\ \mathtt{something}) & \mathtt{is\text{-}a} & \mathsf{Staff} \\
((\mathtt{inverse}\ \mathsf{teaches})\ \mathtt{something}) & \mathtt{is\text{-}a} & \mathsf{Course} \\
(\mathsf{teaches}\ \mathtt{something}) & \mathtt{is\text{-}a} & \mathsf{Busy}
\end{array}$$

  As in Example 1, we have that each postdoc is a staff member, and that each undergraduate course is a course. Here we also have that teaching is always performed by a staff member and involves a course. Moreover, who teaches is busy. Observe that the *DL-Lite* typing assertions on teaches, together with the last assertion, imply the $\mathcal{ALCQI}$ assertion (Staff and (teaches some Course) is-a Busy) of Example 1.

- The set $\mathcal{S}_d$ of data sources consists of the same two data sources.
- The mapping $\mathcal{M}_d$ is

  $$\begin{array}{lcl} \{\, x \mid \mathsf{T}_1(x) \,\} & \rightarrow & \{\, x \mid \mathsf{teaches}(x,y) \wedge \mathsf{UGCourse}(y) \,\} \\ \{\, x \mid \mathsf{T}_2(x) \,\} & \rightarrow & \{\, x \mid \mathsf{Postdoc}(x) \,\} \end{array}$$

  Note that, with respect to Example 1, we have removed the $\mathsf{Staff}(x)$ atom from the first assertion, since implied by the ontology.

Considering again the conjunctive query $\mathsf{Q}_w = \{x \mid \mathsf{Busy}(x)\}$ over $\mathcal{G}_d$, and the same source database, we get the same answers as in Example 1. ■

4.3 Reasoning

The techniques that have been developed for reasoning in *DL-Lite* are quite different from those of traditional DLs, since they are based on a series of results developed in databases for query containment and query answering under constraints [34, 14, 15]. Indeed, differently from more complex DLs, all reasoning tasks in *DL-Lite*, both involving the knowledge base and involving queries, can be done in polynomial time in the size of the knowledge base.

Hence, by resorting to *DL-Lite* instead of $\mathcal{ALCQI}$ as the formalism for representing the global ontology of an OIS, and exploiting the results presented in Section 3.2, we obtain that all tasks related to query answering in an OIS, in particular computing certain answers to queries, can be done in polynomial time in the size of the knowledge base, including the data, and in exponential time in the size of the query. Interestingly, this continues to hold even if we consider *DL-Lite* extended with relations of arbitrary arity.

On the other hand, the results reported in [16] imply that the introduction of inclusion assertions on roles (i.e., role inclusion assertions) makes the polynomial techniques at the base of reasoning in *DL-Lite* inapplicable.

5 Conclusions

We have discussed information integration under a logical perspective in which the global view is seen as an ontology expressed in class-based formalisms. Data sources have been considered simply as systems that provide data, but no further contribution to the query answering process.

The next step is to consider data sources as ontology based systems themselves, equipped with both intensional and extensional information, and with query answering capabilities. This leads us to a form of information integration that is based on autonomous peers that collaborate in making available to clients the information distributed in the system. This form of information integration is referred to as *peer-to-peer* [7, 30, 27]. Its formalization typically requires to go one step further and make a distinction between what is part of the extension, and what is *known* to be part of the extension [22, 16].

Acknowledgments

We would like to thank Natasha Noy for her careful reading of the manuscript and her very useful comments, which helped to substantially improve the readability of the paper.

References

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 254–265, 1998.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2003.

[3] F. Baader and W. Nutt. Basic description logics. In Baader et al. [2], chapter 2, pages 43–95.

[4] C. Batini, S. Ceri, and S. B. Navathe. *Conceptual Database Design, an Entity-Relationship Approach.* Benjamin and Cummings Publ. Co., Menlo Park, California, 1992.

[5] D. Berardi, A. Calì, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. Technical Report 11-03, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 2003.

[6] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams using description logic based systems. In *Proc. of the KI'2001 Workshop on Applications of Description Logics.* CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-44/`, 2001.

[7] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. In *Proc. of the 5th Int. Workshop on the Web and Databases (WebDB 2002)*, 2002.

[8] A. Borgida and R. J. Brachman. Conceptual modeling with description logics. In Baader et al. [2], chapter 10, pages 349–372.

[9] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Accessing data integration systems through conceptual schemas. In *Proc. of the 20th Int. Conf. on Conceptual Modeling (ER 2001)*, pages 270–284, 2001.

[10] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data integration under integrity constraints. In *Proc. of the 14th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2002)*, volume 2348 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2002.

[11] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. A formal framework for reasoning on UML class diagrams. In *Proc. of the 13th Int. Symp. on Methodologies for Intelligent Systems (ISMIS 2002)*, volume 2366 of *Lecture Notes in Computer Science*, pages 503–513. Springer, 2002.

[12] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. On the expressive power of data integration systems. In *Proc. of the 21th Int. Conf. on Conceptual Modeling (ER 2002)*, 2002.

[13] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data integration under integrity constraints. *Information Systems*, 29:147–163, 2004.

[14] A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*, pages 260–271, 2003.

[15] A. Calì, D. Lembo, and R. Rosati. Query rewriting and answering under constraints in data integration systems. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 16–21, 2003.

[16] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. What to ask to a peer: Ontology-based query reformulation. In *Proc. of the 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 469–478, 2004.

[17] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[18] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 386–391, 2000.

[19] D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In I. Cruz, S. Decker, J. Euzenat, and D. McGuinness, editors, *The Emerging Semantic Web — Selected Papers from the First Semantic Web Working Symposium*, pages 201–214. IOS Press, 2002.

[20] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13, 1998.

[21] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of the 6th Int. Conf. on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.

[22] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical foundations of peer-to-peer data integration. In *Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004)*, pages 241–251, 2004.

[23] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, and G. Vetere. DL-Lite: Practical reasoning for rich DLs. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-104/`, 2004.

[24] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing and constraint satisfaction. In *Proc. of the 15th IEEE Symp. on Logic in Computer Science (LICS 2000)*, pages 361–371, 2000.

[25] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 229–264. Kluwer Academic Publisher, 1998.

[26] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.

[27] E. Franconi, G. Kuper, A. Lopatenko, and L. Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In *Proc. of the VLDB International Workshop On Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2003)*, 2003.

[28] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*, pages 67–73. AAAI Press/The MIT Press, 1999.

[29] V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 701–705. Springer, 2001.

[30] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proc. of the 19th IEEE Int. Conf. on Data Engineering (ICDE 2003)*, pages 505–516, 2003.

[31] A. Y. Halevy. Answering queries using views: A survey. *Very Large Database J.*, 10(4):270–294, 2001.

[32] I. Horrocks. The FaCT system. In H. de Swart, editor, *Proc. of the 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 307–312. Springer, 1998.

[33] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, pages 51–61, 1997.

[34] D. S. Johnson and A. C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. of Computer and System Sciences*, 28(1):167–189, 1984.

[35] M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

[36] H. J. Levesque and G. Lakemeyer. *The Logic of Knowledge Bases*. The MIT Press, 2001.

[37] A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *J. of Intelligent Information Systems*, 5:121–143, 1995.

[38] R. Möller and V. Haarslev. Description logic systems. In Baader et al. [2], chapter 8, pages 282–305.

[39] J. D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997.