State of the Art Survey

Deliverable D01

A. Calì[1], D. Calvanese[1], B. Cuenca Grau[3], G. De Giacomo[2], D. Lembo[2], M. Lenzerini[2], C. Lutz[4], D. Milano[2], R. Möller[5], A. Poggi[2], U. Sattler[3]

[1] Free University of Bozen-Bolzano
[2] Università di Roma "La Sapienza"
[3] The University of Manchester
[4] Technische Universität Dresden
[5] Technische Universität Hamburg-Harburg

Project:	FP6-7603 – Thinking ONtolgiES (TONES)
Workpackage:	WP1– Assessment of Fundamental Ontology Based Tasks
Lead Participant:	Free University of Bozen-Bolzano
Reviewer:	F. Baader
Document Type:	Deliverable
Classification:	Public
Distribution:	TONES Consortium
Status:	Final
Document file:	D01_StateOfArt.pdf
Version:	2.2
Date:	Dec. 29, 2005
Number of pages:	81

Abstract

Ontologies are formalism whose purpose is to support humans or machines to share some common knowledge in a structured way. They allow the concepts and terms relevant to a given domain to be identified and defined in an unambiguous way. As such, ontologies are seen as the key technology used to describe the semantics of information at various sites, overcoming the problem of implicit and hidden knowledge and thus enabling exchange of semantic contents. In this report we survey the work on ontologies that has been carried out in recent years. In particular, we first overview the languages that have been proposed for representing ontologies, and present the work on reasoning over ontologies. We then overview the work on ontologies from four different points of view: *(i)* We survey methodologies for designing and maintaining ontologies, presenting automated tools suitable for such tasks. *(ii)* We present languages and architectures for accessing, processing and in general making use of ontologies. *(iii)* We presents several approaches for integrating and merging ontologies by detecting correspondences among them. *(iv)* Finally, we present different approaches for making heterogeneous and autonomous ontologies interoperate, in the sense that the various ontologies are not modified as an effect of interoperating with the others.

Document Change Record		
Version	**Date**	**Reason for Change**
v.1.0	Oct 15, 2005	First draft distributed to partners
v.2.0	Nov 20, 2005	Integration of contributions by partners
v.2.1	Dec 9, 2005	Refinement of section on representation and reasoning
v.2.2	Dec 29, 2005	Final version

Contents

1 Introduction

An *ontology* is a formalism whose purpose is to support humans or machines to share some common knowledge in a structured way. Guarino [Gua98] distinguishes *Ontology*, the discipline that studies the nature of being, from *ontologies* (written with lowercase initial), the ontologies we deal with in the TONES project, that are systems of categories that account for a certain view or aspect of the world. Such ontologies act as standardized reference models to support knowledge sharing and integration, and with respect to this their role is twofold: *(i)* they support human understanding and communication, *(ii)* they facilitate content-based access, communication and integration across different information systems; to this aim, it is important that the language used to express ontologies is formal and machine-processable. To accomplish such tasks, an ontology must focus on the explication and formalization of the *semantics* of enterprise application information resources and of the relationships among them. According to Gruber [Gru93b, Gru95], an ontology is a formal, explicit specification of a shared conceptualization. A *conceptualization* is an abstract representation of some aspect of the world (or of a fictitious environment) which is of interest to the users of the ontology. The term *explicit* in the definition refers to the fact that constructs used in the specification must be explicitly defined and the users of the ontology, who share the information of interest and the ontology itself, must agree on them. *Formal* means that the specification is encoded in a precisely defined language whose properties are well known and understood; usally this means that the languages used for the specification of an ontology is logic-based, such as the languages used in the Knowledge Representation and Artificial Intelligence communities. *Shared* means that the ontology is meant to be shared across several people, applications, communities and organizations. According to the W3C Ontology Working Group[1], an ontology defines a set of representational *terms* used to describe and represent an area of knowledge. The ontology of can be described by giving the semantics such terms [Gru93b]. More specifically, such terms, also called *lexical references*, are associated with (i.e., mapped to) entities in the domain of interest; formal axioms are introduced to precisely state such mappings, which are in fact the statements of a logical theory. In other words, an ontology is an explicit representation of the semantics of the domain data [Mae03]. To sum up, though there is no precise common agreement on what an ontology is, there is a common core that underlies nearly all approaches [UG04]:

- a vocabulary of terms that refer to the things in the domain of interest;
- a specification of the meaning (semantics) of the terms, given (ideally) in some sort of formal logics.

Some simple ontologies consist only of a mere *taxonomy* of terms; however, usually ontologies are based on rigorous logical theories, equipped with reasoning algorithms and services. According to Gruber [Gru93b, Gru95], knowledge in ontologies is mainly formalized using five kinds of components:

1. *concepts (or classes)*, which represent sets of objects with common properties within the domain of interest;

[1] `http://www.w3c.org/2001/sw/WebOnt/`

2. *relations*, which represent relationships among concepts by means of the notion of mathematical relation;
3. *functions*, which are functional relations;
4. *axioms (or assertions)*, which are sentences that are always true and are used in general to enforce suitable properties of classes, relations, and individuals;
5. *individuals (or instances)*, which are individual objects in the domain of interest.

According to [Fen01], the goal of having an ontology is "a shared and common understanding that can be communicated between people and application systems". Ontologies allow the key concepts and terms relevant to a given domain to be identified and defined in an unambiguous way. Moreover, ontologies facilitate the integration of different perspectives, while capturing key distinctions in a given perspective; this improves the cooperations of people or services both within a single organization and across several organizations. We identify four main categories of application of ontologies [JU99]:

- *Neutral authoring:* a given company can develop its own neutral ontology for authoring, developing translators from the main ontology to the different terminologies required by the target systems. For example, an organization that needs to integrate multiple software applications can perform enterprise modeling by using an ontology that represents a uniform semantic core.
- *Common access to information:* when legacy software systems need to interoperate, the ontology provides a support for the *bidirectional* translation from different, autonomously evolved formats to the main (neutral) ontology and vice-versa. This scenario is very similar to neutral authoring, bidirectionality being the main difference between the two. The main advantage here is the reduction of maintainance cost, which is a major fraction of the cost of enterprise information integration [Pol02].
- *Ontology-based specification:* an ontology is used for the specification of the requirements of a software system. Having a common ontology as a basis for the development helps to improve the interoperability among systems that have relationships among them that would be implicit without the common ontology.
- *Ontology-based search:* an ontology is used as a semantic structure for a repository of information (web pages, documents, generic data, etc.), in order to achieve a high level of abstraction. This allows users to ask high-level queries to the repository, and possibly to get answers from multiple repositories in a uniform way. This scenario is captured on the one hand by Data Integration [Ull00, Len02, Noy04], where users gain access to a set of information sources, abstracting away from where the data is actually located and how it can be accessed and retrieved to satisfy a certain request. On the other hand, this scenario is at the basis of the Semantic Web [BLHL01], where a request from the user trigger the discovery and execution of services (in general not only queries to information sources, but also operations with side-effects) that are suitably composed to satisfy the request. To capture the dynamic aspect of the interaction among web services, sophisticated formalisms are needed.

The task of building and using ontologies raises a number of issues that embrace different fields of computer science and related disciplines. In this document we first provide, in Section 2, an overview on ontology representation languages and discuss reasoning over ontologies. Then, in Section 3, we survey methodologies for designing and maintaining ontologies, presenting automated tools suitable for such tasks. In Section 4, we present languages and architectures for accessing, processing and in general making use of ontologies. Section 5 presents several approaches for integrating and merging ontologies by detecting correspondences among them. Finally, in Section 6, we present different approaches for making heterogeneous and autonomous ontologies interoperate, in the sense that the various ontologies are not modified as an effect of interoperating with the others.

2 Ontologies: Representation and Reasoning

An ontology, as a conceptualization of a domain of interest, provides the mechanisms for modeling the domain and reasoning upon it, and has to be represented in terms of a well-defined language. Given such a representation, and ontology-based system should provide well-founded methods for reasoning upon it, i.e., for analyzing the representation, and drawing interesting conclusions about it. In this section we overview several languages proposed for representing ontologies, and discuss the reasoning methods associated to such languages. When talking about representation languages, it is useful distinguish between different abstraction levels used to structure the representation itself. We essentially refer to three levels, called extensional, intensional, and meta, respectively.

1. The *extensional level* is the level where the basic objects of the domain of interest are described, together with their relevant properties.

2. The *intensional level* is the level where objects with common properties are grouped together to form concepts, and relationships between concepts are established. At this level, the properties of concepts and relationships are specified.

3. The *meta-level* is the level where concepts singled out in the intensional level are abstracted, and new, higher level concepts are specified and described, in such a way that the concepts of the previous level are seen as instances of these new concepts.

We mainly concentrate on the extensional and the intensional level, since our primary goal is to provide an account for the basic mechanisms for representing ontologies and for reasoning about them. Categories used in the meta-level of ontologies are mainly related to specific applications where ontology languages are used.

2.1 Reasoning on Ontologies

By "reasoning over an ontology" we mean any mechanism or procedure that makes explicit facts that are represented implicitly in an ontology. Well-founded methods for reasoning about ontologies are required and of interest for several reasons. Here, we would concentrate on two important purposes of reasoning:

- *Validation.* Validating an ontology means ensuring that the ontology is a good representation of the domain of discourse that it is supposed to model. Reasoning is at the basis of validation done automatically (or at least supported by automated tools).

- *Analysis.* Based on the assumption that the ontology correctly represents the domain of interest, the analysis aims at inferring new facts about the domain that are implicitly represented. Again, reasoning plays a crucial role, since it constitutes the primary mechanism through which to exploit the ontology in order to make implicitly represented information explicit. The collected information can then be used in the same way as the one that originally was explicitly present in the ontology, and thus can, e.g., be used also for the purpose of validation.

The basic problem to address when talking about reasoning is to establish which is a "correct" way to single out what is implicit in an ontology. But what is implicit in an ontology strongly depends on the semantics of the language used to express the ontology itself. It follows that it makes no sense to talk about reasoning without referring to the formal semantics of the language used to express the ontology. We will discuss in Subsection 2.2 various ontology languages, classifying them according to different criteria. One criterion we will consider concerns the fact of whether the ontology language provides or not the possibility of dealing with incompleteness in the description of the domain of interest. The presence of incomplete information has as consequence that one has to deal with multiple models. In this context, *reasoning* is used in the proper sense of the term, namely as the task of deriving those facts that hold in all possible models of the ontology. On the other hand, when dealing with ontologies interpreted in a single model, deriving implicit information is a form of *computation* over that model.

We discuss in more detail below the relevant aspects of deriving implicit information. We concentrate first on logical reasoning, and discuss specifically deduction, concentrating on reasoning in Description Logics, which are the most relevant cases for ontologies. We conclude by some observations on the issue of computation.

2.1.1 Forms of logical reasoning

We address the issue of *logical reasoning*, i.e., the issue of deriving implicit information in the case where the ontology language has the expressive power to state incomplete information over the domain. Logical reasoning has been the subject of intensive research work in the last decades. Here, we are especially interested in automated reasoning, investigated primarily in Knowledge Representation in Artificial Intelligence. The basic kinds of reasoning that are relevant in the context of ontologies can be considered as forms of theorem proving in First-Order Logic. This amounts to checking whether a certain fact is true in every model of a First-Order theory. We can classify reasoning based on the type of desired conclusions. According to this classification, we distinguish among the following:

- *Deduction.* A fact is a *deductive conclusion* from an ontology if it holds in every situation (extensional level) coherent with the ontology. Deduction is the most

important and well-investigated form of reasoning considered in the context of ontologies, and we discuss it in more detail below.

- *Induction.* Consider a set of observations at the extensional level done wrt an ontology. We say that an intensional level property is an *inductive conclusion* with respect to the observations and the ontology if *(i)* the ontology itself does not imply the observations, *(ii)* the inductive conclusion is consistent with the ontology and the observations, and *(iii)* the ontology together with the inductive conclusion implies the observations. Induction can be used, for example, to single out new important concepts in an ontology, or to come up with new axioms regarding such concepts, based on observations on individuals. In this sense, induction is the basic form of reasoning for learning [CH94, MS01, Mae03].

- *Abduction.* Consider again a set of observations at the extensional level done wrt an ontology. We say that an extensional level property is an *abductive conclusion* with respect to the observations and the ontology if *(i)* the ontology itself does not imply the observations, *(ii)* the abductive conclusion is consistent with the ontology and the observations, and *(iii)* the ontology together with the abductive conclusion implies the observations. Abduction can be used, for example, to derive explanations of certain facts at the extensional level.

2.1.2 Forms of deductive reasoning

We overview now the most prominent forms of classical deductive reasoning that are of importance both in analysis and in validation of ontologies. We point out that all these forms of reasoning can be applied for basically all specific ontology languages and settings in use, and that, in all cases, reasoning support by automated tools is highly desirable for these activities.

- *Consistency of the whole ontology.* An ontology is *consistent*, if it admits at least one model (possibly more, in the case where the ontology specifies some form of incompleteness), i.e., if its concepts can be populated without violating any of the requirements or constraints asserted in the ontology. When an ontology is inconsistent, its statements altogether are contradictory, and the usefulness of the ontology itself becomes dubious. In fact, an inconsistent theory cannot be a good representation of a domain of discourse. Hence, consistency checking is at the basis of the task of validating an ontology. Note that certain ontology languages are of limited expressive power, and do not even allow for expressing inconsistent ontologies.

- *Consistency of single concepts/relations.* A concept (similar considerations hold for a relation) is *consistent*, if the ontology admits at least one model in which the concept can be populated without violating the requirements imposed by the ontology. The inconsistency of a concept (resp., relation) may be due to a design error or due to over-constraining. In any case, the understandability of the ontology is weakened, since the inconsistent concept stands for the empty concept, and thus, at the very least, it is inappropriately named. Also, an inconsistent concept in an ontology cannot correctly represent any meaningful concept of the domain of

interest. To increase the quality of the ontology, one may remove the inconsistency by relaxing some constraints (possibly by correcting errors), or by deleting the concept, thus removing redundancy and increasing understandability.

- *Equivalence of concepts/relations.* Two concepts (similar considerations hold for two relations) are *equivalent* if they denote the same set of instances whenever the requirements imposed by the ontology are satisfied: in this case one of them is typically considered redundant. Determining equivalence of two concepts allows, e.g., for their merging, thus reducing the complexity of the ontology. Moreover, knowing about the equivalence of two concepts is essential to avoid misunderstanding among different users.

- *Subsumption between concepts/relations.* A concept C_1 *subsumes* a concept C_2 (similarly of relations), if in every model of the ontology, the extension of C_1 is a superset of the extension of C_2. Such a subsumption allows one to deduce that properties for C_1 hold also for C_2. This suggests the possible omission of an explicit generalization. Alternatively, if all instances of the more specific concept are not supposed to be instances of the more general one, then something is wrong with the ontology, since it is forcing an undesired conclusion. Concept subsumption is also the basis for a *classification* of all the concepts of an ontology.

- *Membership of individuals in concepts.* An individual is an instance of a concept, if it belongs to its extension in all the models of the ontology. Determining the (most specific) concept of which an individual is an instance allows for establishing the properties of that individual that logically follow from the knowledge in the ontology, in particular how the individual relates to other individuals in the ontology. Notice that this kind of inference is only of interest in the case where the ontology language, besides the intentional component, foresees also an extensional component allowing for stating properties of single individuals. Also, in the case where a meta-level is present, the concepts or relations of the ontology are considered as individuals wrt the meta-level, and meta-level properties of concepts and relations can be determined by establishing membership of such individuals in the meta-level concepts.

- *Implicit consequences.* A property is an *(implicit) consequence* of an ontology if it holds whenever all requirements imposed by the ontology are satisfied, i.e., the property holds in every model of the ontology. Determining implicit consequences is useful on the one hand to reduce the complexity of the ontology by removing those parts that implicitly follow from other ones, and on the other hand it can be used to make certain properties explicit, thus enhancing understandability.

2.1.3 Deductive reasoning in Description Logics

We discuss now in more detail the work done on deductive reasoning, and we concentrate mainly on what we consider to the most interesting one from the point of view of ontology representation and reasoning, namely the work carried out in the context of Description

Logics (DLs) [BCM+03]. DLs are logics specifically suited for the representation of structured knowledge, and they provide the formal foundation for the ontology languages that are now becoming standard. Indeed, OWL, and more specifically its less expressive variants OWL-DL and OWL-Lite [BvHH+04, PSHH04, SWM04] can be seen as syntactic variants respectively of $\mathcal{SHIF}$ and $\mathcal{SHOIN}$, two DLs of the $\mathcal{SH}$ family.

Description Logics have been devised as formalizations of Frame-based systems [FK85] and Semantic Networks [Qui68, Bra79, AF82], to overcome the problems of these formalisms related to the lack of a clear formal semantics. In DLs, the domain of interest is structured through concepts, denoting sets of objects, and roles, denoting binary relations over the domain. Complex concept and role expressions are formed using specific constructs, and it is the set of allowed constructs that characterize a certain DL. DL knowledge bases are typically constituted by an intensional component (called TBox), asserting inclusions between concepts (and possibly roles), and an extensional component (called ABox), asserting membership of individuals in concepts and of pairs of individuals in roles.

It would be impossible to overview here all the work done on extensions of DLs and reasoning in DLs in recent years. For an in depth treatment we refer to [BCM+03] and the extensive bibliography therein, and specifically to Chapters 2 to 6 [BN03, Don03, SCM03, CDG03, BKW03]. Here, we mention only the work that is fundamental for historical reasons, and the one that is most relevant to ontologies.

Structural subsumption. Techniques for reasoning in DLs have been developed starting with severe restrictions on expressiveness and have subsequently have evolved over time, from specialized, ad-hoc methods to fully general ones. The first approaches were developed under the assumption that one can embody the knowledge represented in the terminology directly into complex concept expressions, rather than assertions (i.e., axioms) in a knowledge base. Therefore, subsumption on concept expressions was regarded as the basic reasoning task. The first algorithms for subsumption between concept expressions were based on what is called *structural subsumption*, i.e., a comparison of the syntactic structure of concept expressions (put into a normalized forms), to detect whether one could be embedded in the other one [BPS94]. This method, while effective (i.e., sound, complete, and polynomial) for less expressive languages, becomes incomplete when additional constructs are present, e.g., disjunction or full negation.

Tableaux algorithms. The studies on the trade-off between the expressiveness of a representation language and the difficulty of reasoning on the representations built using that language [LB87] lead to the idea of carefully analyzing the various constructs of DLs, with the goal of characterizing the computational complexity of the reasoning tasks. This kind of research pointed out the need of a general approach to reasoning in DLs. [SSS91] propose the notion of *constraint system* as a general technique to meet this need. Subsequent investigations showed that constraint systems can be seen as specialized forms of tableaux. Many results on algorithms for reasoning on concept expressions, and their complexity were then derived using tableau-based techniques [DLNS96, DLNN97]. Such techniques, besides being intuitively appealing, provided a useful framework for modularizing the problem of designing reasoning algorithms

for languages formed by different sets on constructs. In fact, a tableau-based algorithm essentially amounts to providing an expansion rule for each of the constructs in the language, and then show the correctness of each rule and the termination of the expansion process. Tableaux-based algorithms were then extended by means of suitable termination strategies (essentially based on sophisticated forms of loop detection) to deal also with assertions in (cyclic) knowledge bases, while staying sound and complete wrt the semantics [Baa91, BDS93, HS99, HST99, BS01, HS05]. The algorithms for reasoning in DLs obtained in this way have also lead to actual implementations by application of clever control strategies and optimization techniques [BH91b, BFH$^+$94, HPS99, Hor03, MH03].

Techniques based on PDL based and on automata. For reasoning over DL knowledge bases and also for reasoning over concept expressions in expressive variants of DLs (in which the knowledge base can be internalized into a concept expression), tableau-based techniques, while effective in practice (see for example the first comparisons of implemented systems in [BHN$^+$92, HPS98]), in general do not provide optimal complexity bounds. Indeed, while reasoning in such DLs turns out to be EXPTIME-complete in most cases, devising tableaux algorithms that work in exponential time turned out to be surprisingly difficult [DGM00, DM00], and such computationally optimal tableaux algorithms have actually not found their way into implemented systems. Such difficulties have shifted the attention to other techniques for reasoning in expressive DLs. In particular, the correspondence between DLs and Propositional Dynamic Logics (PDLs) [FL79] has motivated the research on reasoning techniques for expressive DLs that are based on the translation into reasoning problems in Propositional Dynamic Logics [Sch91, DG95, CDGLN01, CDG03], and therefore rely on the associated automata-based methods [VW84, Var85, VW86]. Recently, in particular for expressive variants of DLs (in particular those including fixpoint constructs), automata-based decision procedures that are computationally optimal have been directly devised [SV01, KSV02, Tob01, CDGL02a]. Indeed, despite the apparent differences, there is a tight connection between automata and tableaux based reasoning algorithm for DLs [BHLW03].

Relationship to data models. The work on conceptual modeling formalisms and tools (see also Subsection 2.2) is relevant to ontologies, due to the many similarities shared by conceptual and semantic data models and ontologies representation formalism. Indeed, there are recent proposals to adopt methodologies for conceptual modeling for ontology building [JDM03]. The tight connection between DLs, conceptual data models used in databases (such as the Entity-Relationship Model [Che76]), and representation formalisms used in information systems (such as UML class diagrams [RJB98, UML05]), has been explored in [CLN99, BCDG05a]. In particular, UML class diagrams are widely adopted as the standard formalisms for modeling the static aspects of software applications. [BCDG05a] shows the surprising results that reasoning on such diagrams is EXPTIME-hard, i.e., as hard as reasoning in very expressive variants of DLs [CDG03], such as those underlying the current standard ontology languages [HST00, HS01].

Concrete domains. An extension of DLs that is highly significant for practical applications is the one with *concrete domains* [BH91a, Lut03], i.e., domains of pre-interpreted objects such as numbers or strings, for which specialized inference systems are available, which should be integrated with the DL reasoner. Reasoning with concrete domains has recently been addressed using automata-based approaches [Lut03], which however do not easily lead to implementations such as those based on tableaux [LM05]. Unfortunately, the extension with concrete domains proves to be rather fragile wrt the addition of other useful modeling features [LAHS05].

Hybrid languages. Hybrid languages are constituted by two (or even more) components, equipped with different knowledge representation capabilities and different reasoning procedures, whose aim is to overcome limitations inherent in each of the two subsystems considered separately. DLs, while quite expressive in representing and structuring the domain of interest, are rather weak as query languages (e.g., the only possible form of join is chaining). On the other hand, query languages as those typically used in databases lack the possibility of fully exploiting a complex structured domain. Hence, different proposals have been made to combine the structuring ability of DLs with the querying ability of Datalog rules. $\mathcal{AL}$-Log [DLNS91] is the first such hybrid language that has been studied from a formal point of view. In $\mathcal{AL}$-Log the combination of the two components is achieved by allowing for the use of DL concepts to express constraints in Datalog clauses, and adopting a form of constrained resolution for inference. $\mathcal{AL}$-Log has further been extended in [Ros99] to more expressive DLs and to negation in the Datalog clauses. Also in CARIN [DLNS91, LR98, Ros99], more expressive variants of DLs are considered, and it is investigated how the presence of roles (i.e., binary predicates of the DL knowledge base) in the Datalog rules impacts decidability of inference. Specifically, it is shown that inference becomes undecidable already for rather weak DLs, and restrictions that ensure decidability are presented.

Access to large data repositories. The idea of using ontologies as a conceptual view over data repositories is becoming more and more popular, and recently efficient management of large amounts of data has become a primary concern in ontology reasoning systems [HLTB04, CHW05]. In this context, it becomes important to single out the contribution of the data to the overall complexity of reasoning, i.e., to determine *data complexity* of reasoning [Var82]. A second important requirement is the possibility to answer queries over an ontology that are more complex than the simple queries usually considered in Description Logics research allowing only for retrieving the instances of a concept [DLNS94, Sch93].

Traditionally, research carried out in DLs has paid only a limited attention to data complexity. Data complexity of instance checking was studied in [DLNS94, Sch93], where it was shown to be intractable (more precisely, coNP-hard) already for rather weak DLs. More recently, DLs have been proposed in which suitable restrictions ensure that conjunctive query answering is polynomial in data complexity [CDGL$^+$05b, CDGL$^+$05a]. In [CDGL$^+$05a], also more precise complexity bounds within the polynomial class are investigated, and the tractability boundary for conjunctive query answering under data complexity is determined.

In [LR98], a hybrid system integrating a DL knowledge base with Datalog rules is studied. Specifically, a tight coNP upper bound for conjunctive query answering under data complexity is shown in a rather expressive DL is shown. However, the considered DL still lacks the necessary constructs to capture semantic data models or UML class diagrams.

An EXPTIME upper bound under data complexity of CQ answering in the very expressive DL $\mathcal{DLR}$ directly follows from the results on containment of conjunctive queries and on view-based query answering in [CDGL98a, CDGL00]. These results are based on a reduction to reasoning in PDL, which prevents to single out the contribution to the complexity coming from the data. In [HMS05], a different technique, based on a reduction to Disjunctive Datalog, allows to single out the contribution of the data, and provides tight upper bounds under data complexity for instance checking (though not for answering conjunctive queries).

Non-monotonic reasoning. Finally, we observe that classical First-Order reasoning is not the only form of reasoning that could be of interest when dealing with ontologies. Indeed, non-monotonic reasoning (see, e.g., [Gin87, CS93]), which allows for the situation where new facts may invalidate old conclusions, goes beyond what can be accomplished in classical logic, which is monotone. Here we have concentrated mainly on research on classical reasoning, which constitutes the vast majority of work on reasoning that is relevant for ontologies. As for non-monotonic reasoning, we only mention [BH95a, BH95b], which investigate the use of defaults (a form of non-monotonic specification) in the context of Description Logics.

2.1.4 Computation

While logical reasoning aims at characterizing the conclusions that one can draw from an ontology in the case where the language used to express the ontology allows for multiple models, computation is the kind of reasoning we perform when the ontology is characterized by a single model.

We mention two types of computation that can be performed over an ontology.

In the first case, computation is performed at the intensional/extensional level of the representation. This kind of reasoning aims at capturing the kind of conclusions we want to draw by looking at a single model, and by eliciting properties that are implicit in this model. A typical setting where it is of relevance to compute extensional properties that hold in an ontology characterized by a single model, is the following: the ontology is expressed in the form of a database, and one wants to compute the result of a query over such a database. Notice that a database can also be considered as a logical theory having a single model, namely the database itself [Rei84]. Every query expressed over the database computes a set of tuples, according to the property expressed by the query. Such tuples can be seen as a kind of knowledge that is implicit in the database. However, their elicitation is not obtained by means of a process looking at several models of the ontology. Rather, the tuples are made explicit by means of a computation over the ontology, in particular the one that governs the evaluation of the query over the single model and derives the result.

In the second case, computation is performed at the meta-level. This is an interesting form of reasoning that aims at capturing the process of coming up with interesting conclusions by looking at the meta-level. Here, we talk about computation because the meta-level of an ontology is usually kept sufficiently simple to be characterized by a single model (although at the meta-level, instead of the object-level). As an example, consider the task of computing the so-called *Most Specific Concept* (MSC) of an individual in an ontology [BK98, KM01, BKW03]. A concept C is called a most specific concept for an individual if the individual is an instance of the concept, and moreover, for every other concept C' of which the individual is an instance, we have that, C is subsumed by C' in the ontology. Note that this notion can help in abstracting a given portion of the extensional level of the ontology, or in defining concepts by examples.

2.2 Ontology Languages

We present now a schematic comparison of the most important ontology representation languages that have been proposed in recent years. We concentrate our analysis on those languages that allow for the specification of the intensional and the extensional levels of a domain of interest. The comparison is done by analyzing and putting the ontology languages into relation according to three main classification criteria:

- **How to express.** This criterion takes into account the basic formal nature of the ontology languages. Under this criterion, we will consider the following classes of languages: languages based on Logic Programming, frame-based languages, conceptual and semantic data models, information system and software engineering formalisms, graph-based formalisms, logic-based languages, XML-related formalisms, temporal languages.

- **What to express.** This criterion takes into account that ontology is a generic term for denoting domain representation, but specific ontology languages may concentrate on representing certain aspects of the domain. In this paper, we concentrate our attention on classes of languages whose primary goal is to focus on:

  - *Class/relation.* We use this class for referring to languages aiming at representing objects/classes/relations.
  - *Action/process.* We use this class for referring to languages that provide specialized representation structures for describing dynamic characteristics of the domain, such as actions, processes, and workflows. These languages may also incorporate mechanisms for the representation of the static aspects of the domain (e.g., objects and classes), but they usually provide only elementary mechanisms for this purpose, whereas they are much more sophisticated in the representation of the dynamic aspects.
  - *Everything.* We use this class for referring to languages that do not make any specific choice in the aspects of the domain to represent, and, therefore, may be in principle used for any kind of contexts and applications.

- **How to interpret the expression.** This criterion takes into account the degree at which the various languages deal with the representation of incomplete information in the description of the domain. Under this criterion, we consider the following classes of languages:
  - *Single model.* Languages of this class represent a domain without the possibility of representing incomplete information. An ontology expressed in this kind of languages should be interpreted as an "exact" description of the domain, and not as a description of what we know about the domain. In terms of logic, this means that the ontology should be interpreted in such a way that only one model of the corresponding logical theory is a good interpretation of the formal description. This assumption is at the basis of simple ontology languages: for example, if we view the relational model as an ontology language (where concepts of the domain are represented as relations), then this language is surely a "single model"-ontology language according to our classification.
  - *Multiple models.* Languages of this class represent a domain allowing for the possibility of representing incomplete information. An ontology expressed in this kind of languages should be interpreted as specifying what we know about the domain, with the proviso that the amount of knowledge we have about the domain may be incomplete. This point of view is at the basis of sophisticated ontology languages: for example, languages based on First-Order Logic represent a domain as a logical theory. This theory may allow for different legal interpretations, and such interpretations correspond exactly to several models. Thus, languages based on First-Order Logic are classified as "multiple models"-ontology languages.

We discuss now various languages proposed in the literature that have been used for or are directly related to the representation of ontologies. We group the various languages according to the first classification criterion mentioned above, namely the one regarding "how to express". We refer to several classes of languages, and within a single class, we refer to various languages of that class.

2.2.1 Languages based on Logic Programming

Languages like Prolog and LISP have always been advocated as languages suited for knowledge representation, and recently they have also been deployed as ontology representation languages. In particular, a syntax based on a mix of first order logic and LISP is at the base of various ontology representation languages like Ontolingua [Gru93b] and OCML [Mot98]. The Ontolingua Framework has the ability to translate ontologies into Prolog syntax, to allow importing the contents of Ontolingua ontologies into Prolog-based representation systems. XSB inc.[2] offers a suite of ontology tools based on extensional CDF, a formalism in which ontologies can be defined using Prolog-style facts.

After the emerging of the object-oriented paradigm, several attempts have been made to bring this philosophy into functional and even logic-programming languages. An example is F-logic [KLW95], considered particularly suited to represent ontologies and used, for

[2] `http://www.xsb.com/`

instance, as an internal representation model in the OntoEdit tool (see Subsection 3.3). F-logic's syntactical framework is similar to that of First-Order Logics, since the most basic constructs of the language are terms (composed of function symbols, constants and variables), and formulas are constructed from terms with a variety of connectives. In general, elements of the language are intended to represent properties of objects and relations among objects. Semantics is given in terms of F-structures and satisfaction of F-formulas by F-structures, and based on a notion of logical entailment. In [KLW95], a resolution-based proof theory for F-logic is provided. Anyway, F-logics intended use is as a logic programming language, to be used both as a computational formalism and as a data specification language.

2.2.2 Frame-based languages

Frame-based languages are based on the notion of *frame*, introduced by Minsky [Min75] with the aim of explaining mental activities. In the context of knowledge representation, it assumed a more specific meaning, representing either an object or a class of object (a concept). When a frame represents a concept, associated attributes represent the properties shared by all instances of the class. An attribute is specified through the definition of a *slot*, which contains all information relevant to the attribute: restrictions on the number of possible values (called slot *fillers*), a default value, which is the one to take for the attribute if more specific information is missing, procedures for calculating the value when it is requested but not yet available, or procedures that are activated if the value is modified or deleted. Additionally, each slot has an associated domain for its fillers. Such domain can either be a concrete domain, such as strings or integers, or another frame specified through its identifier. Moreover it is possible to specify that a frame is a subframe of another one and therefore should inherit all of its properties, i.e., all of its slots. A frame can also represent a single object of the domain, in which case it has a special attribute that relates it to the frame representing the class of which it is an instance. The slots of a frame representing an object are inherited by the frame representing its class, and to each slot a concrete value taken from its definition domain is associated. If not explicitly overridden, the value taken is the default value for the slot. Reasoning in frame systems involves usually both the intensional and the extensional knowledge contained in the frame knowledge base. At the intensional level, the deduction process leads to the construction of a frame taxonomy, using both the explicit inclusion assertions in the knowledge base, and the structural information associated to the frames by virtue of their slots. The taxonomy induces a modification of properties at the extensional level, by assigning values to slots and by propagating the effects caused by the activation of the procedures associated to the slots.

Among the currently used frame-based languages/systems, we mention the following:

- Ontolingua [Gru93b, FFR96] denotes both a system for the management of portable ontology definitions and the language therein used. The Ontolingua tool, rather than a knowledge representation system, is a tool to translate ontologies from a common, shared language, based on of KIF (the Knowledge Interchange Format) to the languages or specification formalisms used in various knowledge representation systems.

- OCML [Mot98] is a knowledge modeling language aimed at providing a framework to support development of knowledge-based systems over their whole life cycle. Its main purpose is to support modeling at the knowledge level, and thus it focuses more on logical than on implementation level primitives. Though the main modeling facilities of OCML closely resemble those of Ontolingua, differently from Ontolingua OCML is directly aimed at prototyping KB applications, and thus has operational semantics and provides interactive facilities for theorem proving, constraint checking, function evaluation, evaluation of forward and backward rules, and also non-logical facilities such as procedural attachments. Apart from its operational nature, another difference is that OCML is not only aimed at representing terminological knowledge, but also behavior, which is supported by primitives that allow to specify control structures.

- GFP (Generic Frame Protocol) [KMG95] and its successor OKBC (Open Knowledge Base Connectivity) [CFFK98, CFF+98] rather than knowledge representation languages, are APIs and reference implementations that allow to access and interact in a uniform way with knowledge bases stored in different knowledge representation systems. They allow to write tools that interact with knowledge representation systems (e.g., graphical browsers, frame editors, analysis tools, inference tools) in a system-independent (and thus interoperable) fashion. GFP was primarily aimed at systems that can be viewed as frame representation systems, while OKBC has been extended to general knowledge representation systems, providing a uniform knowledge model based on a common conceptualization of knowledge bases, classes, individuals, slots, facets, and inheritance. Such a knowledge model is an implicit representation formalism that underlies all the operations provided by OKBC. It serves as an implicit interlingua for knowledge that is being communicated using OKBC, and systems that use OKBC translate knowledge into and out of that interlingua as needed.

- XOL [KCT99] is a language inspired by Ontolingua, designed as an intermediate language for transferring ontologies among ontology-based tools. Its syntax is based on XML, and its semantics is defined as a subset of the OKBC knowledge model called OKBC-Lite. OKBC-Lite preserves most of the essential features of OKBC, while not including some of its complex aspects. The design of XOL uses what its authors call a "generic approach" to defining ontologies, meaning that a single set of XML tags (described by a single XML DTD) defined for XOL can describe any and every ontology. This approach contrasts with the approaches taken by other XML schema languages, in which typically a generic set of tags is used to define the schema portion of the ontology, and the schema itself is used to generate a second set of application-specific tags (and an application-specific DTD) that in turn are used to encode a separate XML file that contains the data portion of the ontology.

2.2.3 Conceptual and semantic data models

In parallel to the work done in knowledge representation, also in databases formalisms for the representation of complex data relationships have been studied. Together with

the relational model [Cod70], in the '70s so called *semantic data models* have been introduced [KKT76, HK87]. While the relational model allows the database designer to separate the logical design of a database from its implementation, conceptual and semantic data models were initially developed to be used in the design phase of database applications as a tool for conceptual modeling. Hence, they offer sophisticated structuring primitives, which allow the designer to represent the data in a form that is conceptually similar to the way in which this data is effectively used. The first semantic data model was proposed by Abrial [Abr74]. Successive research has led to the development of a great variety of semantic data models with different characteristics [HK87]. Among them, the most prominent are the Entity-Relationship (ER) model [Che76], the extended Entity Relationship model [BCN92], the Functional Data Model [Shi81], SDM [HM81], and IFO [AH87].

A major difficulty to the widespread use of semantic data models as a front end to existing database management systems has been their lack of a formal semantics and the ability to reason on conceptual schemas. This difficulty has been overcome only recently, by resorting to variants of expressive Description Logics [CLN99].

Conceptual data schemes and ontologies share many similarities, and there are proposals of using conceptual methodologies and tools for ontology modeling. For example, [JDM03] proposes a methodology for ontology building with semantic models and a markup language to exchange conceptual diagrams at run-time. The proposed methodology is applied to the ORM (Object-Role Modeling) semantic model, but it may be extended to other semantic models as well.

2.2.4 Information systems and software engineering formalisms

Starting from the mid '70s, several formalisms to be used in the design of software and information systems have been developed, and among these, the most influential became those based on the object-oriented paradigm, such as the Conceptual Modeling Language (CML) [Sta96], Telos [MBW80, MBJK90], and the Object Modeling Technique OMT. By the mid '90s, new versions of these methods began to incorporate each other's techniques, and a few clearly prominent methods emerged. This culminated when in 1995 Grady Booch, Jim Rumbaugh, and Ivar Jacobson merged the three most prominent approaches, leading to the development of the Unified Modeling Language (UML) [RJB98, UML05], which is now one of the most widely used formalisms for Information Systems design. UML comprises different parts for modeling different aspects, both static and dynamic ones, of an information system. The primary drawback of UML is definitely the lack of a formal semantics, especially regarding the modeling of dynamic information. For modeling the static aspects of an information system, accomplished primarily through UML class diagrams, recent proposals based on Description Logics [BCDG05a] have established a solid basis, providing also support by automated reasoning tools. The relationship between UML and ontologies based formalisms is further explored in [B$^+$01, GWGvS04].

2.2.5 Graph-based models

In knowledge representation, graph-based formalisms have historically played an important role due to their intuitive nature. Among the most influential ones, we find Semantic

Networks [Qui68, Bra79, AF82, Bra83, BL01a], the early precursors of Description Logics, and Conceptual Graphs [Sow84, CM92], which are still used nowadays and have had a direct impact on the Resource Description Framework [KC04, Hay04, BG04] (see also [BL01a]). We further mention the Conceptual Markup Language (CKML) [Ken99, Ken00], and its subset Ontology Markup Language (OML)[3], an XML-based markup language for knowledge and ontology representation that incorporates the principles of Conceptual Graphs. Finally, Topic Maps are a recent proposal, standardized by ISO [Top02], defining a knowledge interchange format based on XML. A strong drawback is their lack of formal semantics, although recent proposals aim at overcoming this limitation [AdMRV02].

2.2.6 Logic-based languages

The distinguishing feature of knowledge representation languages based on logic is that they are equipped with a well-defined formal semantics based on logic. The domain of interest is typically represented in terms of the classes of objects that are of interest and relationships among such objects. The logic-based semantics allows to derive the meaning of arbitrary sentences of the language from the interpretation of the symbols in the alphabet of the language.

Considering the tradeoff between expressive power and effectiveness/efficiency of reasoning, we distinguish between languages based on First-Order Logic, and those based on Description Logics, which are subsets of First-Order Logic that are well-behaved from the computational complexity point of view.

Languages based on First-Order Logic. The languages we present, namely KIF and CycL allow for the possibility, through reification, of stating properties of the terms of the language itself, i.e., they allow for meta-level statements. Notice that, since such languages are based on First-Order Logic, logical inference for them is undecidable.

Knowledge Interchange Format (KIF) [GF92] is a language for the interchange of knowledge among programs, intended to facilitate the independent development of knowledge-manipulation programs. When a program needs to communicate with another program, it does so by mapping its internal data structures, which can be arbitrary, into KIF.

Started as a research project in 1984, the purpose of Cyc[4] [LG90, Len95] was to specify the largest common-sense ontology aimed at providing Artificial Intelligence to computers. Such a knowledge base would contain general facts and heuristics, and a wide sample of specific facts and heuristics for reasoning by analogy, and would have to span human consensus reality knowledge. Far from having attained its initial goals, Cyc is now a working technology with applications to real-world business problems. Its vast knowledge base enables it to perform well at tasks that are beyond the capabilities of other software technologies. At the present time, the Cyc knowledge base contains nearly 200 000 terms, and for each of it several dozen hand-entered assertions. CycL is the language in which the

[3] http://www.ontologos.org/

[4] http://www.cyc.com/

Cyc knowledge base is encoded. It is a formal language whose syntax derives from First-Order predicate calculus and from Lisp. In order to express common sense knowledge, however, it extends First-Order Logic to handle equality, default reasoning, skolemization, and some second-order features. For example, quantification over predicates is allowed in some circumstances, and complete assertions can appear as intensional components of other assertions. Moreover, CycL functions and predicates are typed.

We further discuss action specification languages on the example of PSL. A fundamental problem in Knowledge Representation is the design of a logical language to express theories about actions and change [Rei01]. One of the most prominent proposals for such a language is John McCarthy's situation calculus [MH69], a formalism which views situations as branching towards the future. Several specification and programming languages have also been proposed, based on the situation calculus. GOLOG [LRL$^+$97] maintains an explicit representation of the dynamic world being modeled, on the basis of user supplied axioms about the preconditions and effects of actions and the initial state of the world. This allows programs to reason about the state of the world and consider the effects of various possible courses of action before committing to a particular behavior. Process Specification Language (PSL) [Grü03, GM03] is a First-Order Logic ontology explicitly designed for allowing correct interoperability among heterogeneous software applications, which exchange information as First-Order sentences. It has undergone years of development in the business process-modeling arena, by defining concepts specifically regarding manufacturing processes and business process. Recently, PSL has become an international standard (ISO 18629). PSL can model both "black box" processes, i.e., activities, and "white box" processes, i.e., complex activities, and allows formulae that explicitly quantify over and specify properties about complex activities. The latter aspect, not shared by several other formalisms, makes it possible to express in an explicit manner a broad variety of properties and constraints on composite activities.

Languages based on Description Logics. Description Logics (DLs) are a family of logic-based formalisms for the structured representation of knowledge about a certain domain of interest. For a presentation of Description Logics and a discussion of reasoning, both in pure DL based languages and in hybrid systems integrating DLs with rule-based formalisms, we refer to Subsection 2.1. Here we discuss only the OWL language [BvHH$^+$04, PSHH04, SWM04], which is considered the standard language for the representation of ontologies on the Semantic Web. Its less expressive variants, OWL-DL and OWL-Lite, in fact are syntactic variants of two DLs of the $\mathcal{SH}$ family.

OWL is a rather expressive language that was strongly influenced by Description Logics, but has its roots also in the frames paradigm and the Semantic Web vision of a stack of languages including XML and RDF. On the one hand, OWL semantics is formalized by means of a DL style model theory. In particular, OWL is based on the $\mathcal{SH}$ family of Description Logics [HST00, HS05], which besides the traditional boolean constructs and quantification, allows for enforcing roles (i.e., binary predicates) to be transitive, and for forming role hierarchies. Such a family of languages provides a reasonable tradeoff between expressiveness and computational complexity of inference. Moreover, practical, tableaux-based decision procedures for reasoning on them are available [HST00, HS05],

as well as implemented systems such as such as Fact[5], Racer[6], and Pellet[7]. The OWL formal specification is given by an abstract syntax that has been heavily influenced by frames and constitutes the surface structure of the language. Moreover, axioms can be directly translated into Description Logics axioms and they can be easily expressed by means of a set of RDF triples [KC04, Hay04]. This property is essential, since OWL was also required to have RDF/XML exchange syntax, because of its connections with the Semantic Web.

2.2.7 XML-related formalisms

XML [BPSM98] is a tag-based language for describing tree-shaped structures in textual form. It has been developed by the W3C XML Working Group[8] and has become the standard de-facto language for exchange of information in the Web. It is self-describing, since by introducing tags and attribute names on documents, it defines their structure while providing semantic information about their content. Given the popularity of XML in exchange of information, XML-related languages have been considered as suitable for ontology representation (and exchange). Similar to logic-based languages, a domain ontology expressed in XML-related languages describes the domain of interest in terms of classes of objects and relationships between them.

Languages for XML document validation. Document Type Definitions (DTDs), which are defined as part of the XML language standard [BPSM98], provide a means to enforce structure on XML documents, and as such can be considered a form of ontology specification mechanisms. However, being designed with the primary aim of enforcing structure on textual documents, they are not well suited for knowledge representation purposes. For instance, they lack primitive data types and typing or inheritance mechanisms, order of elements is relevant. XML-Schema [FW04] tries to overcome some of these limitations by introducing data types, inheritance mechanisms, more flexible nesting rules, and more expressive forms of constraints. For a detailed study of the relationship between ontologies and XML-Schema, we refer to [KBF+03, KBF+03].

As the Web is huge and is growing at a healthy pace, there is the need to describe Web resources, by means of metadata that applications may exchange and process. The Resource Description Framework (RDF) and its ontology-style specialization, the Resource Description Framework Schema (RDFS), have both the purpose to provide a foundation for representing and processing metadata about Web documents. This would lead to a characterization of the information in the Web that would let reason on top of the largest body of information accessible to any individual in the history of the humanity.

Languages for representing information in the Web. Due to the fast growth of the Web, there is a clear need to describe Web resources, by means of metadata that applications may exchange and process. The Resource Description Framework

[5] http://www.cs.man.ac.uk/~horrocks/FaCT/
[6] http://www.sts.tu-harburg.de/~r.f.moeller/racer/
[7] http://www.mindswap.org/2003/pellet/
[8] http://www.w3.org/XML/

(RDF) [KC04, Hay04] and its ontology-style specialization, the Resource Description Framework Schema (RDFS) [BG04], have both the purpose to provide a foundation for representing and processing metadata about Web documents.

Specifically, RDF allows one to add semantics to a document without making any assumptions about the structure of the document. It is particularly intended for representing metadata about Web resources, such as the title, author, etc.. Thus, it enables the encoding, exchange and reuse of structured metadata, by providing a common framework for expressing this information so it can be exchanged between applications without loss of meaning. RDF represents all information in terms of subject-predicate-object triples, and is equipped with a formal model-theoretic semantics [Hay04], which provides a dependable basis for reasoning about the meaning of an RDF expression. However, RDF does not impose any interpretation on the kinds of resources involved in a statement beyond the roles of subject, predicate, and object. It has no way of imposing some sort of agreed meaning on the roles, or the relationships between them. Therefore, in order to use RDF as a means of representing knowledge it is necessary to enrich the language in ways that fixes the interpretation of parts of the language.

RDF Schema (RDFS) [BG04] enriches the basic RDF model, by providing a preinterpreted vocabulary for RDF. Predefined properties can be used to model *instance-of* and *subclass-of* relationships as well as domain and range restrictions of attributes. Indeed, the RDF schema provides modeling primitives that can be used to capture basic semantics in a domain neutral way. That is, RDFS specifies metadata that is applicable to the entities and their properties in all domains. The metadata then serves as a standard model by which RDF tools can operate on specific domain models, since the RDFS metamodel elements will have a fixed semantics in all domain models. However, due to its limited expressive power, RDFS has been extended to more powerful ontology modeling languages, such as OWL.

Summary

We now schematically represent, in the form of a table, the classes to which the languages presented above belong, according to the three classification criteria mentioned above. Moreover, for each language we provide in the table also the main literature references where the language has been proposed and/or its specific characteristics and features have been studied and discussed.

Language	**References**	**What to express**	**Single/multiple models**
Languages based on Logic Programming			
F-logic	[KLW95]	Everything	Single model
Frame-based languages			
Ontolingua	[Gru93b, FFR96]	Class/relation	Multiple models
OCML	[Mot98]	Everything	Single model

GFP, OKBC	[KMG95, CFFK98, CFF+98]	Class/relation	Single model
XOL	[KCT99]	Class/relation	Single model
Conceptual and semantic data models			
Entity-Relationship model	[Che76, HK87]	Class/relation	Single model
ORM	[JDM03]	Class/relation	Single model
Information systems and software engineering formalisms			
UML	[RJB98, UML05]	Everything	Single model
Graph based models			
Semantic Networks	[Qui68, Bra79, AF82, Bra83, BL01a]	Class/relation	Single model
Conceptual Graphs	[Sow84, CM92]	Class/relation	Single model
OML, CKML	[Ken99, Ken00]	Everything	Single model
Topic Maps	[Top02, AdMRV02]	Class/relation	N.A. (no formal semantics)
Logic-based languages – based on First-Order Logic			
KIF	[GF92]	Class/relation	Multiple models
CycL	[LG90, Len95]	Class/relation	Multiple models
Process Specification Language (PSL)	[Grü03, GM03]	Process/Action	Multiple models
Logic-based languages – based on Description Logics			
Pure DL languages ($\mathcal{AL}$ and $\mathcal{SH}$ families)	[SSS91, BN03, BS01]	Class/relation	Multiple models
Hybrid languages ($\mathcal{AL}$-Log, CARIN)	[DLNS91, LR98, Ros99]	Class/relation	Multiple models
OWL	[HPSvH03, PSHH04]	Class/relation	Multiple models
XML-related formalisms			
Languages for XML document validation (DTDs, XML-Schema)	[BPSM98, FW04, SW03]	Everything	Single model
RDF, RDFS	[KC04, Hay04, BG04]	Class/relation	Multiple models
Temporal languages			
Temporal ER model	[KC04, Hay04, BG04]	Class/relation	Single model

3 Ontology Design and Maintenance

We present an overview of the methodologies and tools that have been proposed in the literature for supporting the design and maintenance of ontologies. These methodologies mainly aim at providing guiding principles for ontology design that, if applied in a systematic and disciplined way, facilitate the construction of structured, understandable, and therefore also maintainable ontologies. The available methodologies are concerned with the following issues:

- how to build an ontology from scratch;
- how to support the creation of large ontologies as a collaborative effort of a group of communicating ontology designers;
- how to extract a special purpose ontology from a general purpose one, and how to design a general ontology from multiple special purpose knowledge bases;
- how to support the evolution of an ontology;
- how to debug an ontology.

In our overview, we focus on general methodological aspects instead of delving into the details of concrete ontology languages (which are discussed in Section 2) and their impact on ontology design. We mention support by automated reasoning techniques whenever available, but refer to Section 2 for details.

3.1 Design of Concrete Ontologies

The developers of large and influential ontologies have often published the principles on which their design was based and the methodology that they have followed in order to obtain a structured ontology of high quality. In this section, we survey such individual experience reports, concentrating on well-known ontologies that had a visible impact on subsequent research.

Lenat and Guha published the general methodologies followed during the construction of the massive-scale CYC ontology [LG90]. Their methodology consists of two phases. In the first phase, common sense knowledge that is implicit in the various sources underlying the ontology construction is manually extracted. In the second phase, based on the already constructed ontology new knowledge is acquired using a plethora of natural language processing and machine learning techniqes. The use of machine learning techniques is discussed in a more general setting in the following section.

Uschold and King [UK95] published the main steps followed in the development of the Enterprise Ontology. The method proposes some general steps to develop ontologies, which are *(i)* to identify the purpose; *(ii)* to capture the concepts, the relationships among these concepts, and the terms used to denote both of them; and *(iii)* to codify the ontology. This approach has later been refined into a more subtle and general methodology which is discussed in the subsequent section.

SNOMED CT (Systematized Nomenclature of Medicine, Clinical Terms) is a large-scale ontology for medical terms [SCC97]. SNOMED's main application is to define a medical reference terminology that is used widely in the health system of the united states. SNOMED was originally constructed as a traditional multi-axial coding system. Due to its sheer size (nowadays ∼400.000 concepts), it was later found difficult to continue with such a largely informal approach. Since then, SNOMED is constructed in a lightweight description logic using the Apelon TDE environment and methodology. Of particular importance is the reasoning-based automated ontology classification offered by this system (see "tools" section later on).

Gruninger and Fox reported the methodology used for building the TOVE (TOronto Virtual Enterprise) ontology in the domain of enterprise modelling [GF95]. As the previous ones, their approach consists of several steps. First, the designer has to determine the queries that the ontology has to be able to answer. Second, the objects, attributes, and relations (called terminology in TOVE) that are relevant for answering these queries can be determined. Third, constraints on the elements of the terminology are identified. The resulting specification is represented in first-order logic and implemented in Prolog.

Other methodologies for ontology design can, for example, be found in [Gru93a, GPJP95] and the survey [NH97].

3.2 General Methodologies

We review general methodologies that have been proposed for the design and maintenance of ontologies. Since design and maintenace phases are often not cleanly separated in the literature, we will make an explicit distinction only when appropriate. The methodologies proposed in the literature focus on different aspects of working with ontologies. For example, some approaches propose a general schema to be followed when constructing ontologies, some have an emphasis on the cooperative ontology construction by a group of knowledge engineers, and others concentrate on the automated learning of ontologies. In the following, we group the various approaches in different subsections according to their focus.

3.2.1 General Schemas

We describe some general schemas for ontology construction that have been proposed in the literature. Such a general schema usually consists of a sequence of steps that are to be followed during ontology construction. If necessary, some of the steps have to be repeated until a satisfactory result is achieved. Sometimes, the individual steps are supported by automated reasoning techniques.

Uschold and Gruninger [UG96] propose some methodological approaches for building ontologies. First, they propose to identify the main scenarios in which the ontology will be used. Later, a set of natural language questions, called competency questions, are used to determine the scope of the ontology, that is, the questions that have to be answered using the ontology. These questions are used to extract the main concepts, their properties, relationships and axioms, which are formally defined in Prolog. The main purpuse of this approach is to guide the transformation of an informal scenario into formal models.

Observe that there is a close connection to the TOVE methodology described above.

Methontology [FGPJ97] appeared at the same time and was extended a few years later. It is a methodology that aims at enabling ontology building by people that are non-experts in ontologies and ontology languages. The main feature of methontology is a set of intermediate representations that are half-way beween how people think about a domain and the languages in which ontologies are formalized. The ontology designer first acquires knowledge about the domain in an informal way, and then organizes and structures this knowledge by phrasing it in the intermediate representations. Then, the real ontology is automatically generated using translators. The whole process is guided by a structured ontology development process and an ontology life cycle model.

The OntoClean methodology was developed by Guarino and Welty since 2001 [GW04]. Its aim is to provide guidance for ontology design based on highly general ontological notions drawn from philosophical ontology. These notions are used to define a set of metaproperties that characterize relevant properties of the classes and relations in an ontology. The idea is to assign the metaproperties to the entities of an ontology in a principled way. Then, the backbone taxonomy of the ontology can be identified and it is possible to discover inconsistent and inappropriate uses of the subsumption relationship. OntoClean is independent of ontology languages and application domains and has been used in several large scale ontology projects.

3.2.2 Construction by Abstraction

In the work by Baader et al. [BK98, BKM99], a bottom-up methodology for constructing ontologies formulated in description logics is described. There are two main ingredients to this methodology that both assist the ontology designer in the formulation of new concepts: first, the designer can describe a set of protoypical instances of a new concept. Then, automated reasoning techniques are used to compute the most specific concept description that captures all these instances. Second, the designer can specify a set of existing concepts for which he wants to introduce a common super-concept. Such a super-concept is then automatically generated and presented to the designer for inspection and modification. Note that the introduction of additional such super-concepts can be used to add more structure to ontologies that lack structure, i.e., whose subsumption hierarchy is broad and shallow. Additional reasoning techniques to support ontology maintenance are also available. For example, it is possible to use the reasoning services matching [BKBM99, BK00, BBK01] and unification [BN01, BK01] to identify concepts that intuitively have the same meaning (i.e., should be unified), but are not logically equivalent.

As part of the Esprit KACTUS project, Bernaras at al. [BLC96] present a method for constructing general ontologies out of special purpose knowledge bases. Based on existing knowledge bases that have been constructed for and are tailored towards concrete applications, an abstraction process is used to generate a general purpose ontology of the domain under consideration. Applying this method allows to capture in the ontology the consensual knowledge needed by all the applications. This methodology has been used in the domain of electrical networks.

3.2.3 Construction by Specialization

Swartout et al. [SPKR97] propose a methodology for building special purpose ontologies based on the Sensus ontology. The proposed method is a top-down approach for deriving domain specific ontologies from huge ones. The authors propose to identify a set of terms that are central to a particular domain. Such terms are linked manually to a broad-coverage ontology (in that case, the Sensus ontology, which contains more than 50,000 concepts). Then the relevant terms for describing the domain are selected automatically, and the Sensus ontology is pruned accordingly. The algorithm delivers the hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a special purpose ontology.

3.2.4 Cooperative Ontology Design

Comprehensive ontologies for broad domains are usually constructed by a collaborating group of designers. Often, these designers are geographically distributed. To guarantee the construction of high-quality ontologies in such an environment, methodologies for supporting cooperative design are required. In [Euz95, Euz96], the following problems of cooperative ontology design are identified: management of the interaction and communication among people, data access control, recognition of a moral right about the knowledge (attribution), error detection and management, concurrency management.

Euzenat [Euz96] proposes a protocol that can be used to reach consensus among distributed ontology designers. It is based on the idea that the ontology is split into parts organized in a tree. The designers can discuss and commit about the parts of the ontology separately. The leaves of the tree are called user knowledge bases, and the intermediate nodes are called group knowledge bases. The user KBs are maintained by single designers and there is no need to achieve consensus for them. On the other hand, there must be consensus about the group KBs. Intuitively, a group KB represents the consensual knowledge of its sons.

In the knowledge annotation initiative of the knowledge acquisition community described by Decker et al. [DEFS99], ontologies are developed in a joint effort by a group of designers at different locations. To make the process of ontology building easier, templates for ontology construction are generated and distributed to all ontology designers. It is also assumed that all assigners use the same language. The filled-in templates are sent to coordinating agents, which are experts in the different topics represented in the ontology. Once the ontology coordinating agents have all the portions of the ontology, they integrate them in a principled way. This integration rests on the common pattern enforced by the template.

3.2.5 Automated Learning of Ontologies

Automated learning provides a fundamentally different approach to ontology creation than manual construction by a designer. Though ontology learning is not in the primary focus of the TONES project, we will discuss some relevant references. The majority of papers in this area propose methods to extend an existing ontology with new concepts, using natural language processing, statistical, and machine learning techniques. Concepts

are often understood in a syntactic sense (i.e., as words found in some corpus), and semantic disambiguation is a major research issue. For example, [AAHM00, AM02] use learning to extend the WordNet ontology with new words. Berland and Charniak [BC99] show how to learn part-whole relations in ontologies and Cimiano et al. [CHS05] address the learning of taxonomic relations. An integrated ontology management and learning architecture is proposed by Missikoff et al. [MNV02]. The architecture consists of a software environment centered around the OntoLearn tool that can build and assess a domain ontology for intelligent information integration within a virtual community. Maedche and Staab [MS01, MS00] present an architecture to help ontology engineers in creating an ontology. In this approach, machine learning and manual construction of the ontology are used in an integrated way.

3.2.6 Ontology Evolution

In most applications, ontologies are not static. Instead, they have to be adapted to changing application domains, extensions of their scope, and evolving applications using them. Therefore, ontology evolution is one of the main aspects of ontology maintenance. Noy and Klein [NK04] argue that ontology evoluation is closely related to schema evolution in databases, but that ontology evolution has certain peculiarities. Most notably, these are a different semantics and different usage paradigms. Klein et al. [KFKO02] distinguish conceptual changes (the way a domain is understood) from explication changes (the way how concepts are specified).

In [KN03], changes to an ontology are seen as sequences of individual update operations like a log file of a database system. They discuss minimal transformations between two given ontology states, i.e., how to go from one state to the other with the smallest set of individual updates and how to construct complex update operators from sequences of individual updates (represented as minimal transformations). These update operations can themselves be organized as an ontology and offered to the user in a menu.

3.2.7 Ontology Debugging

During the design and maintenance phase of the ontology lifecycle, it frequently happens that inconsistencies and unwanted (non)-subsumptions are inadvertedly introduced by the ontology engineer. Therefore, debugging of ontologies is a central issue in both phases. As observed for example in [SC03], debugging consists primarily of two subtasks: explaining the problem and correcting it. Since correction is a strongly domain-dependant task, research has focussed on explanation.

Most work for explaining logical reasoning that is relevant for ontology engineering has been carried out in the context of description logics. Early approaches concentrate on explaining reasoning with isolated concepts (i.e., classes) rather than with whole ontologies. For example, [McG96, BFH$^+$99] are concerned with explaining subsumption in inexpressive and expressive description logics, respectively. Later, Schlobach and Cornet [SC03] address the explanation of inconsistencies in ontologies. The basic idea is to identify minimal inconsistent fragments of an ontology (so-called MUPS) to single out the source of an inconsistency. The techniques used for this purpose can be viewed as a re-invention of similar techniques from [BH92]. A further development of the techniques

of Schlobach and an integration with the ontology editor Swoop [KPS+05] can be found in [KPSH05].

Meyer et al. [MLB05] propose a related but more proactive approach: instead of only pinpointing the inconsistency and leaving the corrections to the knowledge engineer, a proposal for weakening the ontology is made such that consistency is achieved. The approach is based on computing maximal consistent fragments of an ontology and borrows techniques from inconsistency management in propositional logic and from non-monotonic reasoning.

3.3 Tools for Ontology Design and Maintenance

We provide an overview of the available tools and software environments that can be used for building and maintaining ontologies. Since the recent interest in ontologies has resulted in a proliferation of such tools, the overview presented here cannot be a complete one. Instead, we attempt to select those tools that dominate in the practice of ontology design or that support ontology design and maintenance in a particularly original and constructive way. The discussed tools usually provide a frame-like graphical user interface for creating ontologies without using a formal specification language. Many of them additionally offer graphical display modes for visualizing the structure of the ontology. Concerning reasoning, we can distinguish between three kinds of software environments: *(i)* Tools that are special purpose w.r.t. the supported language and/or the addressed design methodology sometimes include reasoning capabilities. Frequently, the reasoning capabilities offered by such tools is rather ad hoc, i.e., not based on a formal semantics and on provably sound and complete algorithms. *(ii)* General-purpose tools often allow to plug in external reasoning backends to support the ontology construction process. In such a case, the quality of reasoning depends both on the backend that is plugged in and on the interplay between the ontology tool and the reasoner. *(iii)* Finally, many of the available tools simply do not support reasoning at all.

1. **Apelon TDE**. The Terminology Development Environment (TDE) of Apelon Corp. is a commercial suite of software components for the creation, maintenance, and deployment of large ontologies. It offers a frame-like interface and is tailored towards the construction of massive-scale ontologies that are constructed by a large group of designers which are geographically distributed. TDE supports the detection of conflicts originating from contradictory modelling decisions among developers. Based on a very simply description logic, an automated classification of the ontology (i.e., a computation of the subconcept-superconcept hierarchy) is possible. The empolyed DL is not powerful enough to describe or detect inconsistent classes. Notably, the Convergent Medical Terminology (CMT) and SNOMED CT have been developed using the Apelon TDE.

2. **Apollo** is a language independent ontology editor that was developed at open university, UK, and offers a frame-like user interface. It allows to export the constructed ontology in the RDF, XML, Meta, and OCML formats. A basic compatibility check of slot types, values, and cardinalities is done, but no real reasoning is offered. This

editor is a prototypical example for one of the many available editors that offer a frame-based interface, but provide no user support in the form reasoning.

3. **ICOM** is a tool supporting the conceptual design phase of information systems [FN00]. The conceptual models that can be designed using ICOM are closely related to description logic ontologies, and therefore ICOM can be conceived (and has been used) as a tool for ontology design. ICOM is an evolution of part of the conceptual modelling demonstrators suite [JQC+00] developed within the European ESPRIT Long Term Research Data Warehouse Quality (DWQ) project [JLVV99]. It adopts an extended Entity-Relationship (EER) conceptual data model, enriched with multidimensional aggregations and interschema constraints. ICOM is integrated with reasoners for expressive description logics such as FaCT and RACER. This integration allows to explicate ramifications of the modelling and present them to the designer for inspection. Theoretical results from the DWQ project guarantee the correctness and the completeness of the reasoning process: the system uses the SHIQ description logic, as a mean to provide the core expressivity of the $\mathcal{DLR}$ Description Logic [CDGL98a]. In contrast to ontology editors, ICOM uses a graphical layout of the conceptual model as its main interface rather than working with a frame-like representation.

4. **OntoEdit** [SEA+02] is an ontology engineering environment developed at the Knowledge Management Group (AIFB) of Karlsruhe University. It is a stand-alone application that provides a graphical ontology editing environment (which enables inspecting, browsing, codifying, and modifying ontologies, supporting in this way ontology development and maintenance) and an extensible architecture for adding plug-ins. The conceptual model of an ontology is internally stored using a flexible ontology model that can be mapped onto different concrete representation languages such as XML, F-logic, RDF(S), and OWL. An interface to an F-logic inference engine is provided, and F-logic reasoning [KLW95] is used to assist the modelling process. An interface to description logic reasoners such as FaCT is planned with the goal of supporting OWL reasoning.

5. **Protégé-2000** is a powerful graphical and interactive ontology-design and knowledge-acquisition environment that is being developed by the Stanford Medical Informatics group (SMI) at Stanford University [GMF+03]. It is an open source, highly configurable application that provides a frame-like interface, a graphical ontology editing environment, and an extensible architecture for the creation of customized knowledge-based tools. Its component-based architecture allows to add new functionalities by creating appropriate plug-ins. For example, the Protégé plug-in library contains plug-ins for graphical visualisation of knowledge bases, semi-automatic ontology merging, and for the interaction with automated reasoning systems. Concerning the latter, Protégé can interact with inference engines for the verification of first-order logic constraints and with reasoners for expressive description logics such as FaCT and RACER. Protégé also provides translators to F-logic, OWL, Ontolingua, and RDF(S).

6. **OILEd** is a graphical ontology editor that has been developed at the University of Manchester [BHGS01]. This editor, which is based on a frame-like user interface, focusses on the OWL ontology language and related description logics. A main emphasis is on providing an interface to reasoners for expressive description logics such as FaCT and RACER. The main aim of this coupling is to automatically compute a classification of the ontology and to detect inconsistent concepts that indicate modelling mistakes. The feedback returned by the reasoner is presented in graphical form to the user of the ontology editor. The user can then review this feedback and modify his modelling in case that undesired consequences have been computed.

7. **SONIC** is a reasoning backend for ontology editors such as OILEd and Protégé [TK04]. Its aim is to support the bottom-up construction of ontologies formulated in description logics as described in Subsection 3.2.2. In particular, SONIC allows to automatically generate a super-concept for a set of concepts selected by the user. This concept is then rewritten into a short form and presented to the user for inspection through the used ontology editor, i.e., OILEd or Protégé. SONIC delegates subtasks to reasoners for expressive description logics such as FaCT and RACER.

8. **SWOOP** is an ontology editor for OWL ontologies developed at Maryland University [KPS+05, KPH05]. It aims at providing a web browser like look & feel instead of offering the standard frame-like interface. For example, SWOOP includes hyperlink based navigation across ontological entities, history buttons (back, next), etc. The choice for this editing paradigm is based on the widely-held opinion that ontologies will play a central role in the upcoming Semantic Web, which is a second-generation web in which web pages are annotated with a machine-understandable description of their content. From the Semantic Web perspective, the marriage of web browsers and ontology editors is obviously rather natural. SWOOP also provides an interface to description logic reasoners. The feedback provided by such reasoners is used in a similar way as in OILEd. To support the collaborative design of ontologies, SWOOP allows the annotation of the entities contained in an ontology.

9. **SymOntoX** [MT03] is an ontology management system that makes use of a web-based interface and targets specifically the management of ontologies for the eletronic business domain. SymOntoX allows the management for multipole ontologies, supports access rights for different user profiles, and supports multiple languages. It provides for a set of feature types to decribe concepts: similar concepts, narrower concepts, part-of concepts, and attributes of concepts. By means of access rights, certain roles can be defined for the creation of ontologies. For example, only a super user can propose new terms while ordinary users are restricted to browsing.

For convenience, we provide the webpages of the various reasoners and tools presented above:

Apelon TDE	http://www.apelon.com/products/tde.htm
Apollo	http://appollo.open.ac.uk/
ICOM	http://www.inf.unibz.it/~franconi/icom/
OntoEdit	http://www.ontoknowledge.org/tools/ontoedit.shtml
Protégé	http://protege.stanford.edu/
OilEd	http://oiled.man.ac.uk/
SONIC	http://wwwtcs.inf.tu-dresden.de/~sonic/
SWOOP	http://www.mindswap.org/2004/SWOOP/

Apart from a convenient user interface and mechanisms to support and systematize the communication between collaborating designers, the main aid for ontology design and maintenance provided by current tools is an interface to reasoning systems. These reasoning systems are most often OWL/description logic reasoners, but sometimes also first-order theorem provers and reasoners for RDF. With the notable exception of SONIC, the main purpose of using reasoners is to automatically classify the concept of the ontology into a subclass-superclass relationship, and to detect logical errors that indicate modelling flaws. Although powerful reasoning systems are around, currently the actual use of reasoning to assist the design and maintenance of ontologies is thus relatively limited, and will be subject of investigation within the TONES project.

4 Ontology Access, Processing, and Usage

While in the previous section we have discussed ontology design and management tools, we now consider how ontologies are *accessed*, *processed*, and *used* by humans or machines (i.e., agents, programs, applications, etc.). In the following, we use the term "reasoning services" in a broad sense, i.e., with reasoning services all kinds of services are addressed, even if the services just return information that is apparent from the syntactic specification of an ontology. Since query answering with reference to ontologies involves reasoning, with "reasoning services" we also refer to query processing. We first give explanations for the terms ontology access, processing, and usage.

4.1 Explanation of Nomenclature

Ontology access. With *ontology access* we mean the way a human user or an application program invokes reasoning facilities provided by an ontology inference system. Different architectures are possible. Usually, ontology reasoning services are provided by a server whereas application programs or graphical interfaces are client systems. Ontology management and reasoning systems are usually built as server systems because, at the current state of the art, even in the average case, reasoning about reasonably large ontologies requires quite substantial computational resources in terms of time and space. Thus, it is advantageous to exploit computational results from previous service invocations for many client processes. For smaller ontologies, it is also possible to include the reasoner in the application. Ontology accesss also refers to how access rights are granted to query or reasoning services, and, maybe in the near future, how payment for accessing ontology reasoning services might be organized. Ontologies are seen as resources that are

either stored in files, are managed in databases (e.g., in the form of RDF triples), or are retrieved from remote web servers. In the latter case, access is usually implemented using Internet technology (e.g., HTTP). APIs and languages for ontology access are discussed below.

Ontology processing. The term *ontology processing* describes the way an ontology is handled internally by a reasoning engine in order to implement reasoning services. In this context, facilities are required that can be used to declare which classes of services are required by a client such that heuristics and index structures can be adapted to the needs of the usage scenario. Thus, it would be advantageous if human users or application programs could control ontology processing. Ontology processing also refers to automatic syntactic conversion and transformation steps (preprocessing) that are required if ontologies are accessed that are available in various languages with differing concrete syntaxes. Preprocessing involves ontology translation tasks in order to achieve syntactic interoperability. In order to provide fast access at usage time, preprocessing and setup times comparable to databases (maybe larger by one order of magnitude) have to be taken into consideration. In addition, concurrent service invocations, transactional processing, and load balancing are concerned by the notion of ontology processing. At the current state of the art, research has just begun to explore the latter notions w.r.t. ontologies.

Ontology usage. Ontologies can be used by humans or machines. For instance, humans can inspect ontologies via graphical interfaces (or portal systems) to get an understanding of a specific problem domain (ontology-based learning). Application programs use ontologies as part of the implementation of the program specification. For instance, an enterprise information system might use ontologies for answering queries with respect to different kinds of vocabularies. With *ontology usage* we refer to the purpose of ontology either for a human or for an application. Note that ontology use by humans usally also involves reasoning services whose invocations are hidden behind the interface. Ontology usage patterns are very important for optimizing the performance of handling multiple service invocations, and, in addition, usage patterns allow for the adequate design of reasoning services at the right level of abstraction such that optimized processing is possible (see above).

4.2 Usage Scenarios

For what purposes are ontologies used? We see two main kinds of scenarios: On the one hand, ontologies are used for conceptual domain modeling for data being stored, for instance, in databases. On the other hand, with the expressivity of current ontology languages, it is very well possible to also represent data (and not only conceptual data models). Ontologies provide for indefinite descriptions, i.e., in constrast to data representations in databases one can also use an ontology language to describe data in a detailed way in some respects while leaving open certain alternatives in others (i.e., one only restricts possible values of, for instance, an associated object, but does not directly state a particular associated object). In the logical view, a database instance corresponds to a (logical) model. With indefinite data descriptions, multiple models are possible (and,

hence, with ontologies, usually, multiple database instances are described). However, all models share a common part. For instance, in classical description logics, the conceptual modeling part is done with a so-called T-box (terminological box) whereas the data description part is done with an A-box (assertional box). However, in newer description logic languages with increased expressivity, the distinction is blurred, i.e., restrictions for single domain objects can also be part of the T-box. For instance, the W3C standard OWL (Web Ontology Language [SWM04, PSHH04, BvHH+04]) includes so-called nominals as part of the ontology language (for defining the T-box). If the ontology contains restrictions about domain objects, descriptions for data have an influence on the answer to ontology queries (e.g., w.r.t. subsumption), and, hence, processing ontological information can hardly be used for different sets of data descriptions. Nevertheless there still exists a W3C language standard for data descriptions (RDF, Resource Description Format [KC04, FW04]) that corresponds to an A-box if considered from a logical perspective [PH03]. In practice, reusing an ontology w.r.t. different sets of data restrictions is a very common usage scenario in many application contexts. From a practical perspective, a useful restriction would be that data descriptions (i.e., A-boxes, possibly defined in RDF) cannot be given for nominals used at the ontology level (see also the subsection on ontology processing below).

Ontologies are used for mutiple purposes at design-time and at runtime of an application. Common usage scenarios are:

- evaluation of conceptual data models w.r.t. "upper ontologies" (checking satisfiability of named concepts, finding unwanted subsumption relations etc.);
- optimization of database queries (exploit implicit subsumption relations to rewrite queries);
- matching of service descriptions (checking what terms must be added or removed to achieve subsumption);
- query answering w.r.t. data descriptions and w.r.t. to conceptual data models with expressive axioms in contrast to query answering in databases where axioms from the conceptual data model are not taken into consideration;
- query answering w.r.t. to "foreign" data models (query rewriting) in an information retrieval context;
- situation recognition (concept-based: classification, instance-based: direct types inference service);
- extracting commonnalities of data w.r.t. ontologies (e.g., least-common subsumer w.r.t. T-boxes);
- providing descriptions of data w.r.t. an ontology (e.g., computing the most-specific concept, rewriting of concepts w.r.t. a T-box).

Almost all high-level reasoning services are reduced to standard reasoning services (low-level services) such as satisfiability checking, subsumption checking etc. however, it

is a non-trivial task to implement high-level services in a layered, server-based architecture. Usually, imlementations for high-level services require a vast amount of low-level service calls, and if high-level services are not implemented in the server itself, communication costs are tremendous.

Recent research efforts investigate "ambient" ontology use (see research on the Semantic Web). The idea is to seemlessly integrate a reference ontology of the environment with the user's ontology, for adapting to the situational context, the location of ontology use, and the interface device. This involves ontology integration and ontology-based query rewriting and is discussed in the respective sections.

4.3 Access Languages and Protocols

For an ontology interface, various different access languages corresponding to different views on how to specify ontologies have been proposed. Predominant views are the frame-based view (GFP [KMG95], OKBC [CFF+98]), the logic-based view (e.g., DIG [BMC03], Common Logic [CLS05]), and the agent-communication-based view (KQML [FFMM94]). At the current state of the art, there is no commonly agreed on definition of explicitly given information (told information) and information that is entailed due to the semantics of the representation formalism (inferred information). For some applications, however, in particular for user interfaces, the distinction definitely makes sense.

At the architectural level, various protocols for client-server-based ontology architectures have been investigated in the literature. From a technical point of view, there is nothing particular to accessing ontologies, i.e., standard Internet technology is employed, and many concrete syntaxes are based on XML and are defined with XML schemata. In most cases, nowadays, ontologies are available as XML-based web resources. A lurking danger for ontology languages, however, is the inclusion of representation constructs that have been developed for programming languages (e.g., short unsigned integer) rather than semantically well-founded representation constructs that are based on mathematical theory.

In the following we distinguish between functional interfaces and query interfaces.

Functional access. With functional interfaces one can incrementally specify ontology statements (tell interface) and one can invoke functions that implement standard reasoning services for ontologies (ask interface for told and inferred information). In contrast to other modeling approaches, ontology interfaces do not offer access to internal data structures (e.g., a graph structure of related domain objects) but separate a client application from the reasoning systems using a functional interface (see [BPGL85] for one of the first accounts on this topic).

Query-based access. For some tasks, retrieving information using a functional interface results in too many (possibly naive) function invocations, which is particularly problematic in a client-server-based setting (see the discussion above). Declarative query languages are used to overcome the problems in this case. In the context of ontologies, query languages are not only used for retrieving data objects. Rather, query languages can also used to find certain elements of the ontology itself (e.g., names for concepts,

relations, nominals etc.). Examples for query languages that allow for the retrieval not only of data but also of ontological structures are OWL-QL [FHH04] or nRQL [WM05], both with slightly different expressivity. Both languages provide for incremental access to the result set of submitted queries (set-at-a-time vs. tuple-at-a-time retrieval mode). Query-based access is usually characterized by a small number of rather complex queries.

Subscription-based access to ontology servers and data descriptions. The access to ontologies can also be organized by managing a set of subscriptions. In this context, the access pattern is usally characterized by a very large number of clients with rather simple queries. Subscription-based access to ontologies is particularly important if incremental changes are handled by an ontology server. In this case a subscription causes notifications about changes of query result sets to be sent to the subscribers in an asynchronous way.

Ontology portals. Portals for ontology repositories have been investigated recently. In particular, a query approach for finding ontologies is based on (sub)string search and link-based reference counting (Swoogle [FJDP05]). Some authors also propose portals that support manual ontology reviews and public discussions about ontology quality. At the current state of the art, there is no reward system for making ontologies available that are useful for many application scenarios. However, one can imagine systems that provide a ranking for ontologies (and ontology authors) based on "citations", i.e., references in articles and system implementations (see what the system CiteSeer does for standard publications). Such an "OntoSeer" system might provide the basis for application-oriented scientific work in many scientific fields (e.g., medical ontologies, ontologies for e-commerce, ontologies for natural language processing or image understanding, etc.).

4.4 Archictectures for Efficient Ontology Processing

Ontology languages for different purposes have different expressivity. In many cases exponential worst-case algorithms are required. Some prefer even undecidable languages (e.g., first-order logic or higher-order-logic) and are willing to sacrifice completeness of reasoning services. Usually, however, decidable languages are used and ontology processing relies on average-case efficient algorithms (e.g., this holds for description logics, (disjunctive) datalog, rule languages with answer-set semantics, etc.).

4.4.1 Control strategies, configuration of reasoning services

According to usage and access patterns (see above) current ontology processing systems can be instructed to employ appropriate control strategies for inference. In addition, the computation of index structures can be done in a preparation phase. However, there are no standards available for languages for controlling the behavior of reasoning systems, and, in fact, a standard will be rather hard to define due to different processing strategies employed in practical systems. Todays ontology systems use a mixture of, e.g., tableau-based, resolution-based, or datalog-based techniques, and use top-down or bottom-up

inference algorithms. Developing a standard for specifying control strategies seems a long-term goal that can hardly be reached in the near future.

4.4.2 Optimization techniques

For different inference algorithms various kinds of optimization techniques have been developed (see, e.,g., [HM01, Hor03, MH03, HM04]. Explaining all of them is impossible in this overview. Major systems use a combination of techniques: rewriting (e.g., normal forms), separation of unrelated ontology parts, caching, binary constraint propagation (e.g., forward checking), dependency-directed backtracking (e.g., backjumping), semantic branching, optimized blocking, model-merging, taxonomy traversal optimization, average-case-efficient encoding techniques (e.g., with specific kinds of binary decision diagrams), indexing, subgoal ordering, topological sorting of classification steps, automata-based techniques for regular expressions to name just a few. At the current state of the art, new optimization techniques for huge amounts of data descriptions as well as for new representation constructs in expressive ontology languages have to be developed. Furthermore, not only huge amounts of data descriptions have to be handled (e.g., in the A-box part in term of description logics), but also very large amounts of terminology statements (e.g., in T-boxes) are required for practical applications. In addition, ontology-based access to (possibly existing) database instances (single-model reasoning) is a fruitful research topic. High-level reasoning problems as discussed above also require sophisticated optimization techniques. In general, it is a long way from a decision procedure developed to investigate formal properties such as decidability and complexity to a system implementation that is stable w.r.t. real-world input sizes. Even management of told information and answering of queries w.r.t. told information is a non-trivial task, and some framework provide optimizations for this (e.g., Jena).

4.4.3 Incremental processing of queries (or ask statements)

A common problem is that, except for subscription-based access, an ontology system sees only one query at a time. For instance, in some usage scenarios, queries will be refined, i.e., the next query will return a subset of the instance of a previous query, but this is not known to the ontology system. Optimized processing would be possible, if this knowledge were exploited. However, determining query subsumption and managing required caches pays only back if the resources are actually exploited. At the current state of the art, strategies for dynamic adaption to automatically detected query patterns have not been investigated. This might be interesting, however, since standards for control languages for ontology processing are hard to achieve (see the arguments discussed above).

In some usage scenarios, not all results of a query will probably be exloited (e.g., in an information retrieval scenario). Ontology processing servers must provide for means to incrementally compute results that are transferred to the client on request (computation on demand if there is a high load on the ontology server or eager compuation if no further queries are to be answered). In addition to client-side caching, ontology processing involves managing server-side caches and index structure computation. In the incremental result set computation mode, it is advantageous if the ontology processing engine can give

a hint to the application if, for computing further elements, there is a jump in computational complexity expected. In other words, ontology processing strategies should provide for means to retrieve "easy answers" first. Newer ontology systems provide these facilities, however, the separation between "easy" and "hard" answers is technology-driven (opportunistic) and not semantically well-founded. It would be better, if processing behavior were determined by semantics- and complexity-theoretic insights.

4.4.4 Incremental processing of tell statements

Incremental changes and updates to ontologies have been supported by some system implementations, implemented reasoning algorithms are known to be incomplete. Although theoretical investigations show that, even for languages with limited expressive power, coping with new axioms added to ontologies is of the same worst-case complexity than computing results from scratch [Neb90, 158ff.], in the average-case, quite substantial speedups might be expected by new optimizations for complete techniques for incrementally handling additional axioms by exploiting previously computed results. Note that it is not only the set of data descriptions (A-box in description logic ontologies) but also the part of the ontology that corresponds to the terminological part (T-box) that must support for incremental updates for incorporating new representational primitives in order to support practical requirements.

4.4.5 Multi-client access to multi-processor ontology servers

As has been argued above, ontology servers can be used by multiple clients. In a standard scenario, one can assume rather few clients which specify complex queries. At the current state of the art, ontology servers support locking and support for multiple contexts that can be referred to in ask expressions as well as queries (e.g., multiple T-boxes and A-boxes if ontology servers based on description logic are considered). In the near future, ontology servers will support multi-processor computer architectures and will include proxies supporting load balancing with replicated ontologies. Load balancing with replicated services for a particular ontology is also required if a single application uses multiple threads. Depending on the scenario, one can envision client-side proxies with local caching in order to reduce network resource consumption for synchronous query answering via Internet technology (i.e., via TCP, HTTP, and the like). In addition, access to distributed ontologies has been investigated recently [BS03].

4.4.6 Stability of terminological reasoning results

From a practical perspective, it is advantageous to achieve the decoupling of reasoning about the conceptual level from reasoning about data descriptions (see the discussion above). It is a valid assumption that ontologies change rather rarely on the conceptual level where for the data description level, in the near future, transactional processing systems must be developed for multi-client scenarios. With restrictions about data objects at the conceptual level, for instance, new subsumption relations might emerge if new data descripton are added. In this case, for instance, the taxonomy might change due to new information about data objects. In this case, terminological reasoning results are

not stable. New optimization strategies must be developed to cope with these effects efficiently. In addition, the effects must be carefully considered if ontologies are used in applications.

4.4.7 Benchmarking

Benchmarking ontology processing is a research field that will become more and more important as ontologies are processed as part of more and more applications. Although, some progress has been made about methodological generation strategies for artificial benchmarks, due to the nature of ontology processing w.r.t. average-case behavior, additional research is still necessary to allow for the assessment of different techniques and architectural compromises that underly new-generation ontology processing systems with multi-processor architectures as well as multi-client asynchronous access as usually employed in loosely coupled systems of the future.

5 Ontology Integration and Merging

The acceptance of the Web Ontology Language (OWL) [PSHH04, SWM04] as a standard will facilitate the proliferation of independently developed ontologies. In this scenario, different ontologies need to be confronted and related to each other, either to produce a single integrated and reconciled ontology that deals with a larger domain of interest or to establish a *connection*, with a precise semantics, between the different ontologies, which remain distinct. Often, ontologies to be integrated have been developed by different groups of experts and may present *mismatches* that need to be reconciled. Thus, their integration may require a previous reconciliation of the terms they contain.

In the last few years a rapidly growing body of work has been developed under the names of *Ontology Mapping and Alignment*, *Ontology Merging* and *Ontology Integration* [KS03b, Noy04]. This work is very diverse and has been originated from different communities. Given this diversity, it is difficult to identify and formalize the problems to be solved and to comprehensively integrate the various approaches into a common framework.

However, a bold divide into the different approaches available in the literature could be presented as follows:

1. methods for (semi-automatically) *detecting* correspondences between terms in the signatures of the ontologies to be integrated;

2. frameworks and formalisms for *representing* connections and correspondences between ontologies;

3. methods for assessing the *consequences* of the integration.

We briefly survey these different categories

5.1 Methods and Tools for Detecting Correspondences between Ontologies

Many researchers in the Semantic Web and Knowledge Engineering communities agree that discovering correspondences between terms in different ontologies is a crucial problem. Sometimes two ontologies refer to similar or related topics but do not have a common vocabulary, although many terms they contain are related. Often, ontologies are too large and complex for humans to manually establish these correspondences.

Current practices for discovering correspondences (or mappings) between ontologies comprise a large number of techniques developed by various communities, such as Machine Learning and Linguistics. These methods are based on different heuristics and highly overlap with the techniques traditionally used by the database community for integrating database schemas [RB01, DH05].

In general, the existing methodologies for (semi)automatic mapping discovery accept as input a set of ontologies to be integrated and return a set of correspondences between the entities in their respective signatures. Most of the existing approaches proposed in the literature rely on some notion of *similarity* and consist of roughly 4 stages, as identified by Ehrig and Staab [ES04]:

1. Transformation of the initial representation of the input into an adequate format.
2. Identification of the search space.
3. Computation of the similarity values between candidate mappings.
4. If several similarity values are obtained (typically using different heuristics) for a candidate mapping, aggregation of the different values into a single similarity value.
5. Computation of the mappings from the obtained similarity values.

Some algorithms perform several iterations of the steps 1)–5) in order to refine the results.

In what follows, we survey the different techniques proposed in the literature. For more details, we refer the interested reader to [KS03b, Noy04].

1. The **PROMPT** and **ANCHORPROMPT** algorithms [NM03] were originally designed for assisting knowledge engineers in the process of merging and aligning ontologies. The system provides different heuristics for suggesting mappings to the users and identifying the concepts and roles to be merged. The heuristic techniques used to provide the suggestions range from string matching methods to algorithms that try to find structural similarities in the definitions of concepts. The system also takes into consideration the manual correspondences potentially identified by the users. The main limitation of both PROMPT and ANCHORPROMPT is that the heuristics used do not take into consideration most of the expressive power provided by modern ontology languages, such as OWL-DL.

2. The **FCA-Merge** [SM01] method for ontology merging is based on Formal Concept Analysis techniques. The approach taken by the authors is "extensional", in the sense that it is based on objects/individuals which appear in both ontologies. Concepts having the same individuals are then supposed to be merged. Thus, the method heavily relies on the availability of instances in the ontologies to be merged. The Formal Concept Analysis techniques employed by the system allow to derive conceptual hierarchies from data tables. The ontology concepts are then clustered and the concept lattice is pruned accordingly. Concepts generating the same cluster are suggested to be merged. The generation of the merged ontology from the pruned concept lattice is semi-automatic and requires human interaction. The knowledge engineer is responsible for resolving possible conflicts and duplicates. A number of heuristics are used for assisting such a process.

3. The **GLUE** [DMDH02] system uses machine learning techniques for discovering mappings. Given two ontologies to be merged, for each concept in one ontology GLUE finds the most similar concept in the other ontology. GLUE uses multiple learning techniques for establishing concept similarity that exploit the information stored in both the TBox and the data. The similarity measures that can be employed are definable solely on the joint probability distribution of the concepts involved.

4. The **IF-MAP** technique, presented in [KS03a], is an automatic method for discovering mappings between ontologies. The IF-MAP algorithms are grounded on the theory of information flow. The mappings are formalized in terms of logic infomorphisms. The IF-MAP tool works in four main steps: 1) Ontology harvesting; 2) Translation; 3) Infomorphism generation and 4) Display of results. In the ontology harvesting step, ontologies are collected across the Web. These ontologies can be represented in different formats and thus the need for an ontology translation step. The algorithm then tries to establish logic informorphisms between two ontologies and the results are stored in an RDF knowledge base for future reference and maintenance reasons.

5. The **ONION** system [MW02] provides a generator of mapping (articulation) rules between ontologies. These rules are established manually with the help of a library of heuristic matchers. A human expert is in charge of validating the matches between concepts in the ontologies to be integrated and a learning algorithm is introduced that takes advantage of user's feedback in order to generate better mappings in future integrations.

6. Ehrig and Staab [ES04] present **QOM**, a mapping generation algorithm, and argue its computational efficiency and the substantial quality of the obtained mappings evaluated upon expert validation. The QOM algorithm runs in $\mathcal{O}(n \log n)$ in the worst-case, with n being the size of the signature of the ontologies to be integrated. The ontologies accepted as an input are constrained to be represented in RDF-Schema and thus are pretty light-weight. Mappings are defined in terms of a correspondence partial function between the signatures (atomic concepts, roles, and individuals) of the input ontologies. The mappings are established between entities with the same ontological status (i.e., concepts to concepts, roles to roles, and

individuals to individuals). The authors propose a similarity measure between entities, based on different heuristics. In order to optimize the process, QOM avoids a complete pair-wise comparison of class hierarchies and uses an incomplete top-down strategy. Bijection is always enforced to the mapping function.

7. Finally, in [PDYP05] the authors present a methodology for ontology mapping based on **BayesOWL**, a probabilistic framework for handling uncertainty on the Semantic Web. The ontologies to be integrated are translated into Bayesian Networks. A concept mapping module takes a set of learned similarities between concepts as input and computes the mappings as a result of reasoning across the Bayesian Networks.

5.2 Frameworks for Representing Connections and Correspondences Between Ontologies

Once a set of mappings have been obtained, the ontologies to be integrated typically contain a *common signature*. In some frameworks, the terms in the common signature of the input ontologies are given a special *semantics*. In this section, we develop a comprehensive review of the different semantics proposed in the literature.

5.2.1 Distributed Description Logics

Borgida and Serafini [BS03] propose the *Distributed Description Logics* (DDL) formalism for representing the mappings between expressive DL ontologies. The particularization of DDLs to OWL resulted in an extension of Web Ontology Language, called C-OWL [BGvHS03].

Distributed Description Logics were devised for combining different DL knowledge bases in a loosely coupled information system. The idea of the combination is to preserve the "identity" and independence of each local ontology. The coupling is established by allowing a new set of inter-ontology axioms, called *bridge rules*. From the modeling point of view, bridge rules have been conceived for establishing directional ("view dependent") subsumption relationships between classes and correspondences between individuals in different ontologies.

5.2.2 $\mathcal{E}$-Connections

Cuenca Grau et al. [CG05, CGPS05] propose the $\mathcal{E}$-*Connections* framework for integrating ontologies that deal with largely different subject matters. The $\mathcal{E}$-Connections framework is a technique for combining logical formalisms, first presented in [KLWZ04, Kut04]. $\mathcal{E}$-Connections have many desirable properties and, in particular, they are characterized by a very robust computational behavior.

The general idea behind $\mathcal{E}$-Connections is that the interpretation domains of the connected knowledge bases (which are possibly written in different logical languages) are kept *disjoint* and interconnected by means of *link relations*. The language of the $\mathcal{E}$-Connection incorporates a set of operators associated to the link relations, which talk about the relationships between the connected knowledge bases.

From the modeling perspective, each of the connected ontologies in an $\mathcal{E}$-Connection is modeling a different application domain, while the $\mathcal{E}$-Connection itself is the *union* of all these domains. For example, an $\mathcal{E}$-Connection could be used to model all the relevant information referred to a certain university, and each of its component ontologies could model, respectively, the domain of people involved in the university, the domain of schools and departments, the domain of courses, etc. Support for $\mathcal{E}$-Connections has been provided in major Semantic Web tools, such as Manchester's OWL-API [BLV03], the OWL-DL reasoner Pellet [SPCG+05] and the ontology editor SWOOP [KPS+05, KPH05] (see Section 3.3). An extension of the Web Ontology Language for supporting $\mathcal{E}$-Connections has also been proposed. In [KLWZ04] DDLs have been proven to be subformalisms of $\mathcal{E}$-Connections, yet they have pursued an independent development.

5.2.3 Further approaches

Madhavan et al. [MBDH02] propose a language, with a formal semantics, for representing mappings. Based on the semantics, the authors identify a set of reasoning services associated with the mappings. In the context of the proposed framework, the mappings are understood as formulae that provide a semantic relationship between the concepts in the ontologies to be integrated. The authors claim that the proposed framework enables mappings between ontologies represented in vastly different representation languages, without first translating the models into a common language. The paper identifies a core set of reasoning services that can be used to determine whether a mapping is adequate for a particular context, namely: 1) query answerability, 2) mapping equivalence and 3) mapping composition.

5.3 Methods for Assessing the Consequences of the Integration

Once the correspondences between the ontologies under consideration have been found and represented, it is of prime importance for the ontology modelers to predict, understand and control the *consequences* of the integration. In particular, it often happens that, due to the interaction between the integrated ontologies, unintended consequences are introduced, which are often undesirable and hard to keep track of.

Therefore, there is a need to define suitable reasoning services to verify whether the integrated ontologies behave as expected. It is also of the utmost importance to describe the most common ontology integration scenarios for applications and identify which kinds of consequences are desirable/undesirable in each case.

Surprisingly, as opposed to the problems of mapping discovery and mapping representation, the problem of predicting and controlling the consequences of ontology integration has been largely overlooked by the Ontology Engineering and Semantic Web communities.

To the best of our knowledge, the problem has only been tackled, very recently, in [GLW05]. The authors propose a set of reasoning services based on the notion of a conservative extension of an ontology and assume that two ontologies with overlapping signature are to be merged into a single, reconciled ontology. The authors also provide decidability and complexity results for the proposed services, assuming that the ontologies to be integrated are represented in the basic description logic $\mathcal{ALC}$.

5.4 Open Problems and Future Directions

We have seen that the ontology integration problem can be focused from three complementary perspectives, namely *(i)* the problem of discovering likely correspondences between the vocabularies of the given ontologies, *(ii)* the problem of representing these correspondences in a logically principled way and finally, *(iii)* the problem of controlling and predicting the impact of the integration on the component ontologies.

From this problems, we believe that the latter remains mostly unexplored and requires a thorough investigation.

On the other hand, we believe that the problem of ontology integration is strongly dependent on the modeling paradigm adopted for the integration. Thus, for example, different assumptions and goals may be appropriate from a modeling perspective, different application scenarios require different kinds of coupling between the integrated ontologies. We identify two basic ways of integrating ontologies:

1. An ontology (or set of ontologies) dealing with a certain subject matter are integrated with an *upper ontology*, that describes more general terms, that are often domain-independent. Several well-known ontologies have already been developed specifically to be used as formal upper ontologies. Prominent examples are the Suggested Upper Merged Ontology (SUMO) [NP01] and DOLCE [GGMO03]. The IEEE Standard Upper Ontology Working Group is in the process of developing a standard upper ontology to be used in a wide variety of applications.

2. Different ontologies describing largely different subject matters are integrated to describe a broader subject. The maintenance and evolution of each ontology can be then assigned to a different group of experts. This is the case of some prominent ontologies such as the OWL-S ontologies [The05], NASA's SWEET Ontologies and the National Cancer Institute (NCI) Thesaurus [GFH+03]

We believe that different assumptions on the way the shared terms are used can be adopted depending on the integration paradigm. Also, different semantic preservation assumptions can be defined. We aim at exploring these issues in detail in the near future.

6 Ontology Interoperation

In order to achieve full interoperability among different ontologies, not only the issue of their integration and merging, which has been faced in Section 5, is important, but other main problems need to be addressed, such as *(i)* the problem of processing users' requests, e.g., queries, formulated over an ontology, by exploiting the entire knowledge provided by the set of interoperating ontologies; *(ii)* the problem of exchanging knowledge and data between autonomous and independent ontologies; *(iii)* the problem of ontology update; or *(iv)* the problem of service interoperation over the ontology-based interoperability environment. In all the above problems, the specification of how different ontologies are connected one another plays a crucial role, i.e., any possible approach strongly depends on the form of the mapping that specifies the semantic relationship between the ontologies. Therefore, the issues of definition, discovery, composition, and management of the

mapping, which have been faced in the section on ontology integration, are crucial for ontology interoperability and orthogonal to the problems listed above.

In the following, we consider the research fields which constitute the scientific base over which relies the achievement of ontologies interoperation. We introduce the main characteristics of each of such fields, survey the state of the art, and describe the impact of having ontologies, highlighting what have been done in the literature for such a scenario and what still needs investigation. Before this, we start with a brief description on how data management is evolved during the years in order to introduce the scientific contexts which are the object of our investigation. Finally, We conclude this section by grouping together open problems and possible research directions.

6.1 Data management

Data management systems have been continuously evolving during the years to respond to customer demand and the new market requirements. Starting from the late 80s, centralized systems, which had often produced huge, monolithic, and generally inefficient databases, had been replaced by decentralized systems in which data are maintained in different sites with autonomous storage and computation capabilities. All such systems are characterized by an architecture in which data returned to a user query might be not physically stored at the site queried by the user. In distributed databases, decentralization of data is generally achieved to enhance system performance, and it is precisely designed and controlled. However, such an architecture is not able to support the integration of previously existing systems, where data dispersed over several sources are required to be accessed in a centralized and uniform way. Database federation tools enable data from multiple heterogeneous data sources to appear as if it was contained in a single federated database. Such tools provide mechanisms which mask the native characteristics of each source and represent it in a common format, thus enabling a centralized and transparent data access. Mediator-based data integration systems provide in addition the capability of defining a global schema representing the unified view of the application domain, which is related to the sources through a suitable mapping establishing a semantic relationship between them. Here, the integration can be performed in a declarative way, and query answering is by means of powerful mechanisms and advanced techniques. Actually, mediator-based data integration systems represent the first attempt to achieve semantic interoperability among different (pre-existing) data management systems. Therefore, they are the starting point of our investigation.

6.2 Mediator-based Semantic Interoperability

6.2.1 The problem

The goal of mediator-based data integration systems is to provide clients with the access to data stored in heterogeneous and autonomous sources, without the need to know the physical characteristics of such sources and the precise location of the data. Starting from the late 90s, research in this field has mostly focused on declarative approaches [Ull97, Len02] (as opposed to procedural approaches), where the mediator-based data integration

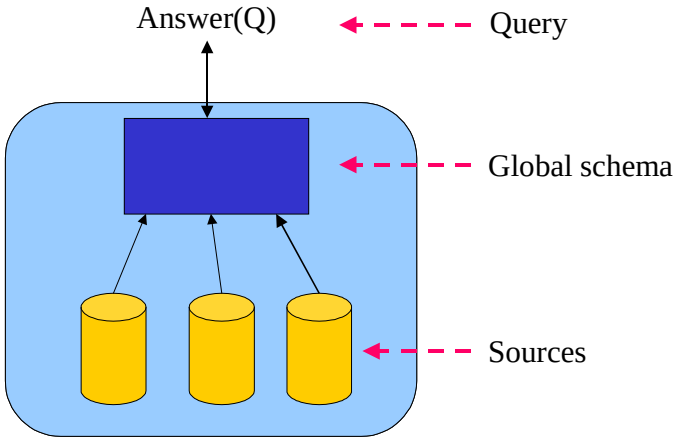

Figure 1: Mediator-based data integration system

system exports to the user a global reconciled view of the data (called global schema) in terms of which user requests are formulated, and the system maintains a declarative specification (i.e., a mapping) of the interrelationships between the global schema and the data at the sources, as shown in Figure 1.

Two basic approaches for specifying the mapping have been proposed in the literature. The first approach, called *global-as-view* (GAV), requires that a view, i.e., a query, over the sources is associated with every element of the global schema, so that its meaning is specified in terms of the data residing at the sources [GMPQ$^+$97, TRV98, GBMS99]. Conversely, the second approach, called *local-as-view* (LAV), requires the sources to be defined as views over the global schema [KLSS95, DG97, CDGL$^+$01b]. A third approach, which accounts for both GAV and LAV mappings, is called GLAV [FLM99]. A further generalization of this, described in [Len02], considers mappings constituted by assertions in which a query over the global schema is put in correspondence with a query over the sources. Obviously, this is the most general form of mapping among the ones listed above. However, data integration under such form of mapping is still largely unexplored.

Among the various problems related to data integration, the problem of answering queries posed over the global schema is the one that has been addressed most intensively. It is well known that this aspect calls for addressing the problem of query processing using views [Ull97, Len02], which concerns the issue on how to use the information about the global schema, the mappings, and the data stored at the sources, to answer the user queries posed over the global schema. Often, this problem has been investigated in those cases in which integrity constraints (ICs) are specified over the global schema. ICs allow for enriching the representation of the integration domain, therefore constitute a powerful tool from a modelling point of view. However, they strongly affect the query answering process, since data stored at the sources may be in general incomplete or inconsistent with respect to such constraints, and both such aspects need to be considered in order to provide complete and consistent answers to user queries. With the term complete we intend all those answers that are logically implied by the (first-order logic formalization) of the data integration system and with the term consistent we intend the possibility of obtaining meaningful answers also in the presence of data that contradict global integrity

constraints. Notice that according to classical first-order formalization of data integration systems, in these cases, we should obtain any tuple as answer to any query, therefore dealing with inconsistent data requires, in some sense, to go beyond first-order logic. Notably, mediator-based data integration in the presence of global integrity constraints is strongly related to the problem of data integration when the global schema is expressed in terms of an ontology.

Query answering has been addressed in various settings, such as the relational setting under various assumptions on the languages for the mapping and the queries [GM99, DGL00, Lev00], a global schema formulated in an expressive conceptual data model [CDGL$^+$98b, CDGL$^+$01b], semistructured data sources and schemas [CDGLV99a, CDGLV99b, CDGLV00, CDGLV02, CDGLV05], and is still the subject of intensive investigations. In the following, we briefly describe some of the main proposals in the field.

6.2.2 State of the art

Despite the accurate work done for precisely formalizing the problem of data integration, first systems (developed in the middle 90s) can not be always framed in terms of the formal framework described above.

Systems like TSIMMIS (The Stanford-IBM Manager of Multiple Information Sources) [CGMH$^+$94], or Garlic [CHS$^+$95] can be essentially considered as (simple) hierarchies of wrappers and mediators. Wrappers are modules that hide the real nature of a data source, and present it and its data in a suitable format adopted within the system. Each wrapper manages the access to a single source and is in charge of translating queries over such a source in the specific language it uses, taking the answer the source returns, and providing them to the mediators. Each mediator is in charge of performing actual integration, by triggering the wrappers in order to provide the answers to the users' queries, putting together data returned by the wrappers, and providing answers to users (or feeding in turn other mediators). It has to be stressed that in TSIMMIS no global integration is ever performed, and each mediator works in an independent manner. As a result, for example, a certain concept may be seen in completely different and even inconsistent ways by different mediators. This form of integration (which can be classified within the GAV approach) can be called *query-based*, since each mediator supports a certain set of queries, i.e., those related to the view it provides, and is clearly characterized by a procedural approach.

Conversely, systems like Information Manifold (IM) [LRO96, LSK95], or INFOMASTER [GKD97, DGL00] follow a more declarative approach, according to which query answering in data integration is actually considered a form of reasoning in the presence of incomplete information. Such systems allow for the declarative specification of a global schema, a source schema (both schemas are assumed to be relational), and a mapping between them, which for both systems is specified according the LAV approach (queries in the mapping are conjunctive queries). With respect to the problem of query processing, IM adopts an algorithm for the processing of conjunctive queries, called the *bucket algorithm* [LRO96], which suitably rewrites a user query into a union of conjunctive queries specified over the source schema, whose evaluation over the source extension returns the

answers to the user query. Such an algorithm is proved to be sound and complete with respect to the problem of answering user queries (under a first-order logic formalization of the system), only in the absence of integrity constraints on the global schema, but it is in general not complete when integrity constraints are issued on it. The procedure at the basis of query processing in INFOMASTER is called *inverse rules algorithm* [DGL00]. Developed originally for user queries expressed as conjunctive queries, the inverse rules algorithm has been then extended in INFOMASTER to handle also recursive Datalog queries, the presence of functional dependencies over the global schema, or the presence of limitations in accessing the sources (binding patterns). In all such cases the algorithm is shown to be sound and complete (assuming that data at the sources are not inconsistent with respect to global integrity constraints).

As for the GAV approach, an in-depth analysis on data integration in the presence of integrity constraints has been carried out in IBIS (Internet-Based Information System) [CCDG$^+$03]. In such a system, cases are considered (and solutions are provided) where the global schema is relational and present key and foreign key dependencies. However, analogously to all the other systems mentioned so far, IBIS does not allow for the integration of inconsistent data (but according to the first-order based semantics adopted in the system, it allows for integration of only incomplete data).

In this respect, we can say that data integration systems have rarely faced the problem of inconsistency of data in a formal and declarative way. Often, the approach adopted to remedy to this problem has been through data cleaning [BL01b]. This approach is procedural in nature, and is based on domain-specific transformation mechanisms applied to the data retrieved from the sources. Only very recently first academic prototype implementations have appeared, which provide declarative approaches to the treatment of inconsistency of data, in the line of the studies on consistent query answering [ABC99]. Among the most interesting proposals, we mention the INFOMIX system heterogeneous data sources (e.g., relational, XML, HTML) accessed through relational global schemas over which powerful forms of integrity constraints can be issued (e.g., key, inclusion, and exclusion dependencies), and user queries are specified in a pawerful query language (e.g., Datalog). Even if also the LAV approach has been studied in the INFOMIX project, the INFOMIX prototype currently supports only GAV data integration. The query answering technique proposed in such a system is based on query rewriting in Datalog enriched with negation and disjunction, under stable model semantics [CLR03, GLRR05]. A setting similar to the one considered in INFOMIX is the one at the basis of the DIS@DIS system [CLRR04]. Even if limited in its capability of integrating sources with different data formats (the system actually considers only relational data sources), DIS@DIS however provides mechanisms also for integration of inconsistent data in LAV.

Among the studies on consistent query answering in data integration systems, we also cite [BB05, BB03], where an approach similar to the one followed in INFOMIX is adopted. However, the repair semantics considered in that papers is different from the one adopted in INFOMIX, and, to some extent, it seems not adequate to capture also incomplete data. A prototype system implementing the above techniques is currently under development [BB04], but no details on implementation are available at the present time.

Other interesting proposals on consistent query answering are the Hippo sys-

tem [CMS04b, CMS04a], and the ConQuer system [FFM05, FM05]. However, such proposals have been essentially developed in the context of a single database system, and therefore do not deal with all aspects of a complex data integration environment, including source wrapping, global schema and mapping definition, data alignment and transformation. Furthermore, w.r.t. classes of constraints and query language considered, the Hippo and the ConQuer systems are to some extent orthogonal to the INFOMIX and the DIS@DIS systems. They are geared towards highly efficient query answering for specific, polynomial-time classes of queries, whereas INFOMIX and DIS@DIS, instead, aim at supporting more general, highly expressive classes of queries (including also queries intractable under worst case complexity).

A different approach in mediator-based semantic interoperability looks at data management under the perspective of exchanging data between the sources and the global schema. Sources are again connected by means of mappings to the global schema, but in this case, the focus is on materializing the data flowing from the sources to the global schema. This problem is addressed in particular by the studies on Data Exchange. In short, Data Exchange is the problem of taking data structured under a source schema and creating an instance of a target schema that reflects the source data as accurately as possible. Since there may be many solutions to the data exchange problem for a given source instance, identifying universal solutions which are homomorphic to every possible solution is a crucial issue in Data Exchange. Furthermore, in order to materialize databases which are as small as possible, "smallest" universal solutions need to be identified. Among several papers produced in the field, we mention [FKMP03, FKP03, ABFL04].

6.2.3 The role of ontologies

Notably, the framework at the basis of mediator-based data integration systems can be generalized in order to deal with situation in which each source to be integrated in a unified system presents its local ontology, and semantic integration and reconciliation of source ontologies is required, in order to extract information from such sources. This is the setting considered for example in [CDGL02b], where the authors propose a formal framework for Ontology Integration Systems (OISs). Their view of a formal framework deals with a situation where there are various local ontologies, developed independently from each other, assisting the task to build an integrated, global ontology as a means for extracting information from the local ones. Ontologies in their framework are expressed as Description Logic (DL) knowledge bases, and mappings between ontologies are expressed through suitable mechanisms based on queries, which actually correspond to the GAV and LAV approaches adopted in data integration.

We point out that most of the work carried out so far on ontology is on which language or which method to use to build a global ontology on the basis of the local ones [BKFH00, DFvH$^+$00], whereas the problem of querying an integrated ontology, i.e., a global schema expressed in terms of an ontology, needs still further investigation. Among the first attempts we mention [CDGL$^+$98b, CCDGL01, CDGL$^+$01b].

6.3 Peer-to-Peer Semantic Interoperability

6.3.1 The problem

More recently, the issue of data integration, has been investigated in the more dynamic context of Peer-to-Peer (P2P) data computing [HIST03]. In short, a P2P system is characterized by an architecture constituted by various autonomous nodes that hold data and that are linked to other nodes by means of mappings. In a P2P system, each peer provides part of the overall information available from a distributed environment, without relying on a single global view, and acts both as a client and as a server in the system. Moreover, the various nodes adopt a suitable infrastructure for managing information.

In recent years, the P2P paradigm has been imposing in different contexts where the issue of cooperation, integration, and coordination between information nodes in a networked environment assumes a crucial role, including the Semantic Web [HH01], Grid computing, service oriented computing and distributed agent systems [PKY03, HBCS03]. In all these systems, the problem of interoperability still needs deep investigation, and the role of ontologies and the contribution to semantic integration that they can provide have still to be addressed.

6.3.2 State of the art

P2P systems have recently become popular for content sharing, and a number of different approaches have been studied to perform content retrieval in such networks (e.g., adaptation, deterministic placement of contents) [CMH+00, WDKF02]. In particular, the P2P paradigm was made popular by Napster, which employed a centralized database with references to the information items (files) on the peers. Gnutella, another well-known P2P system, has no central database, and is based on a communication-intensive search mechanism. More recently, a Gnutella-compatible P2P system, called Gridella [APHS02], has been proposed, which follows the so-called Peer-Grid (P-Grid) approach. A P-Grid is a virtual binary tree that distributes replication over community of peers and supports efficient search. P-Grid's search structure is completely decentralized, supports local interactions between peers, uses randomized algorithms for access and search, and ensures robustness of search against node failures.

As pointed out in [GHI+01], current P2P systems focus strictly on handling semantic-free, large-granularity requests for objects by identifier, which both limits their utility and restricts the techniques that might be employed to distribute the data. These current sharing systems are largely limited to applications in which objects are described by their name, and exhibit strong limitations in establishing complex links between peers. To overcome these limitations, data-oriented approaches to P2P have been proposed recently [HIST03, BGK+02, GHI+01]. For example, in the Piazza system [GHI+01], data origins serve original content, peer nodes cooperate to store materialized views and answer queries, nodes are connected by bandwidth-constrained links and advertise their materialized views to share resources with other peers.

Differently from the traditional mediator-based setting, integration in data-oriented P2P systems is not based on a global schema. Instead, each peer represents an autonomous information system, and information integration is achieved by establishing

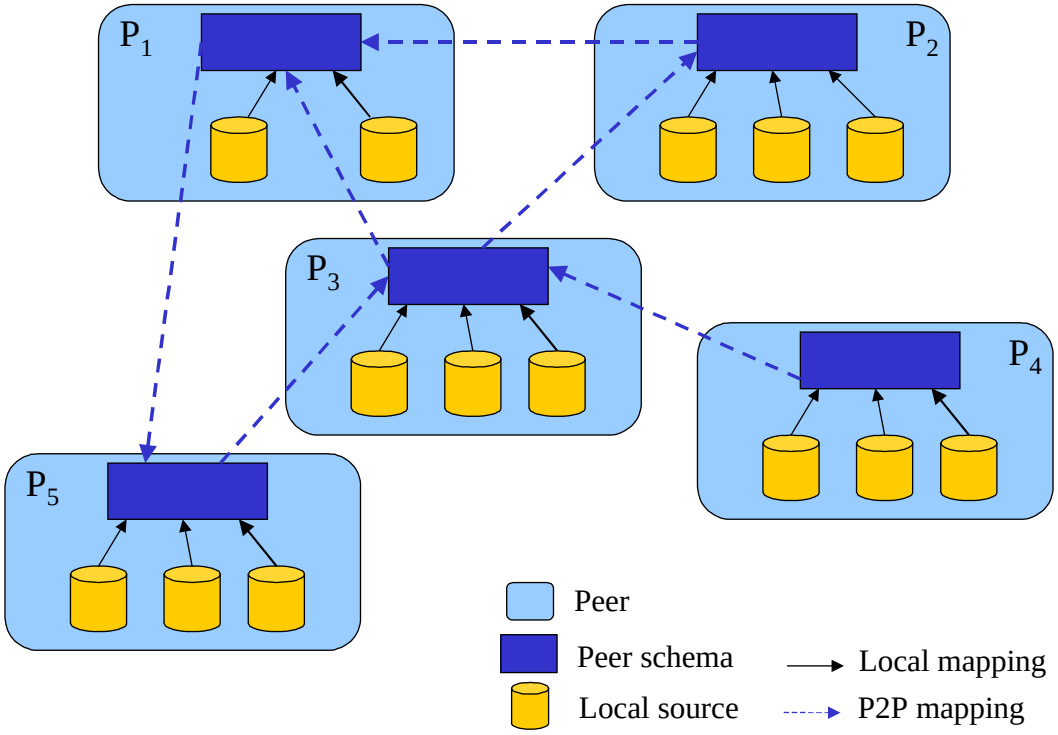

Figure 2: P2P data integration system

P2P mappings, i.e., mappings among the various peers. Queries are posed to one peer, and the role of query processing is to exploit both the data that are internal to the peer, and the mappings with other peers in the system.

More formally, in a *P2P data integration system* [HIST03] each peer is essentially a mediator-based data integration system, i.e., it manages a set of local data sources semantically connected, via a *local mapping*, to a (virtual) global schema called the *peer schema*. In addition, the specification of a peer includes a set of *P2P mappings* that specify the relationships with the data exported by other peers (as shown in Figure 2). Information in such systems can be either *queried* to a peer (by external users or other peers), or *exchanged* between peers. In the former case, the queried peer, by exploiting its P2P mappings, can make use of the data in the other peers for providing the answer, whereas in the latter case, the issue emerges of materializing in a peer data retrieved by other peers.

While in the traditional setting where a global schema is present, techniques for query answering and data exchange have been studied and developed extensively, there is still a fundamental lack of understanding behind the basic issues of data integration in P2P systems. Indeed, since no single actor is in charge of the whole system, it is unrealistic to assume restrictions on the overall topology of the P2P mappings [HIST03, Koc02, FKMP03]. Hence, one has to take into account that the mappings may have an arbitrary structure, possibly involving cycles among various nodes. This needs to be addressed both from the point of view of modeling the system and characterizing its semantics, and from the point of view of computing answers to queries posed to a peer. As for the modeling problem, it needs to be investigated whether the usual approach of resorting to a first-order logic

interpretation of P2P mappings (followed, e.g., by [CL93, HIST03, BGK+02]), is still appropriate in the presence of possibly cyclic mappings, or whether alternative semantic characterizations should be adopted [CDDG+03]. As for the computational perspective, the basic task of computing query answers in P2P systems is still largely uninvestigated. Difficulties arise from the necessity of distributing the overall computation to the single nodes, exploiting their local processing capabilities and the underlying technological framework.

Although correct from a formal point of view, the usual approach of resorting to a first-order logic interpretation of P2P mappings, has several drawbacks, both from the modeling and from the computational perspective. Consider, for example, three central desirable properties of P2P systems:

- *Modularity*: i.e., how autonomous are the various peers in a P2P system with respect to the semantics. Indeed, since each peer is autonomously built and managed, it should be clearly interpretable both alone and when involved in interconnections with other peers. In particular, interconnections with other peers should not radically change the interpretation of the concepts expressed in the peer.

- *Generality*: i.e., how free we are in placing connections (P2P mappings) between peers. This is a fundamental property, since actual interconnections among peers are not under the control of any actor in the system.

- *Decidability*: i.e., are sound, complete and terminating query answering mechanisms available? If not, it becomes critical to establish basic quality assurance of the answers returned by the system.

Actually, these desirable properties are weakly supported by approaches based directly on FOL semantics. Indeed, such approaches essentially consider the P2P system as a single flat logical theory. As a result, the structure of the system in terms of peers is lost and remote interconnections may propagate constraints that have a deep impact on the semantics of a peer. Moreover, under arbitrary P2P interconnections, query answering under the first-order semantics is undecidable, even when the single peers have an extremely restricted structure. Motivated by these observations, several authors proposed suitable limitations to the form of P2P mappings, such as acyclicity, thus giving up generality to retain decidability [HIST03, Koc02, FKMP03]. A different approach, which does not impose limitation on the topology of the P2P system, and which aims at guaranteeing modularity and generality of the system and at the same time decidability of query answering is the one followed in [CDGLR04, CDGL+05b, FKLS03]. In all such papers, an epistemic-logic interpretation of the P2P mappings, as opposed to the first-order interpretation discussed above, is proposed, and algorithms for query processing in P2P system (also in the presence of inconsistent data [CDGL+05b]) are given.

Analogously to the case of mediator-based data integration, in the P2P architecture a different approach to achieve cooperation between different peers can be the one of exchanging data between peers. Peers are again interconnected by means of mappings, but in this case, the focus is on materializing the data flowing from one peer to another. Whereas traditional Data Exchange has been the subject of several recent investigations,

P2P Data Exchange has so far received little attention. In [FKMT05] the problem of deciding the existence of a solution and establishing computational complexity of such a decision process is addressed in Peer Data Exchange, a setting in which only two peers interact that have different roles and capabilities. However, Data Exchange in a full-fledged P2P setting remains still unexplored.

6.3.3 The role of Ontologies

When Peers export an ontology (rather than a simple relational schema), they are also called knowledge-based peers [SG00, CDGL$^+$04a]. In knowledge-based peers the problem of how to exploit the mappings between peers in order to answer queries posed to one peer is in general a complex issue, even in very simple setting (e.g., when the whole system is constituted by two interoperating peers). Indeed, such a problem is in general related to the problem of finding a way to answer queries relying only on the query answering services available at the peers. Each peer of the P2P system provides the service of answering queries expressed over its exported schema, and in general such services are the only basic services that we can rely upon in order to answer queries.

Consider, for example, a music sharing system, and assume that the peer SongUniverse exports both actual songs and knowledge about various types of music, e.g., the fact that live rock songs are live performance songs. Assume now that SongUniverse stores live songs and also knows that other live rock songs can be retrieved from the remote peer RockPlanet. Now, when Carol visits SongUniverse to get live songs of U.K. artists, what this peer can do is: (*i*) directly provide her with the live songs of U.K. artists that it stores locally, and (*ii*) deduce that also live rock songs suit Carol's needs, and reformulate Carol's request by asking RockPlanet about live rock songs of U.K. artists.

The above example shows that query answering on knowledge-based peers is a complex form of query reformulation. This problem is crucial in several contexts, as, for example data integration, in particular in the case where the global schema is expressed as an ontology. Recent studies on query rewriting under integrity constraints [CLR03, CCDGL04] are strictly related to the problem of query reformulation above mentioned. Then, this problem is of clear relevance for the Semantic Web, even if research on the Semantic Web has focused more on the problem of ontology matching (i.e., finding the mapping between peers). The problem of reformulating queries over ontologies has been investigated in [CDGL01a, TCS01], whereas, the problem of query reformulation over ontology-based peers has been discussed and analyzed in [CDGL$^+$04a].

6.4 Semantic Grid Infrastructure

Let us now turn our attention to the architectures realizing the P2P paradigm. In this respect, we point out the growing importance of the notion of Grid. Grids aim at providing a suitable infrastructure for Virtual Organizations, based on standardized services that implement well-established and largely supported patterns, which hides the complexity of heterogeneous data sources and handles the dynamics of the networking environment. This motivates the current trend of modelling complex business infrastructures as Grids, and this is why Grid technologies attract so much interest in the industry. In fact, the

Open Grid Services Architecture (OGSA) is the foundational layer for developing integrated and virtualized (i.e. "On Demand") operating environments at IBM. In particular, Data Grids allow seeing heterogeneous, distributed, and dynamic informational resources as they were a single, uniform, stable, secure, and reliable database, with the aim of facilitating the application development, speed up business integration, and ultimately for the users' sake. Grid-based Virtual Databases are essentially loosely-coupled database federations, which integrate heterogeneous sources, with the purpose of responding to business demands in a flexible manner. However, as the integration logic is generally deemed as an "ad hoc" application, they achieve integration at a merely "functional" level. In other words, as they can't rely on any sort of integrated view of source's metadata (e.g. ontologies, metadata mappings), the best they can do is integrating sources case-by-case. Virtual Organizations urge therefore on the next fundamental step, namely Semantic Data Integration. It follows that results on P2P data integration will enable the adoption of the Grid architecture in open and dynamic distributed systems.

Current proposals for semantic integration on Grids adopt a hierarchical and centralized architecture based on the notion of global schema built over a collection of autonomous information sources, whereas no attempts to perform P2P data integration on a Grid infrastructure exist. As far as we know, the only exception is represented by Hyper [CDGL+04b], a joint research initiative of University of Rome "La Sapienza" and IBM Italia, which aims at developing principles and techniques for P2P data integration on a Grids. In the Hyper framework each peer represents an autonomous information system, and information integration is achieved by establishing mappings among the various peers without resorting to any hierarchical structure. Queries are posed to one peer, and the role of query processing is to exploit both the data that are internal to the peer, and the mappings with other peers in the system. In the Hyper framework, the semantics of a P2P data integration system is given in terms of epistemic logic, in the line of [CDGLR04], and a query answering algorithm is provided which is coherent with both the semantics and the Grid infrastructure over which the Hyper P2P framework is deployed.

6.5 Semantic Service Interoperability

6.5.1 The problem

Service Oriented Computing (SOC) is the computing paradigm that utilizes services as fundamental elements for realizing distributed applications/solutions. Services are self-describing, platform-agnostic computational elements that support rapid, low-cost and easy composition of loosely coupled distributed applications. From a technical standpoint services are modular applications that can be described, published, located and invoked over a network: any piece of code and any application component deployed on a system can be wrapped and transformed into a network-available service. The SOC paradigm allows organizations to expose their core competencies declaratively, over the a variety of networks (including the Internet), using standard (XML-based) languages and protocols, and is facilitated by open standards.

6.5.2 State of the art

Service Oriented Computing [PG03, CLM03, ACKM04] promises to give rise to new opportunities in developing and deploying distributed software applications, by suitably assembling services offered by different organizations. This is facilitated by the use of open (XML-based) standard languages (e.g., WSDL, WS-BPEL - formerly known as BPEL4WS, WS-CDL) and protocols (such as SOAP and XML Protocol), which provide a basic substrate for wiring together the different services constituting the distributed application.

Supported by such a technological layer, research on service oriented computing has mainly concentrated on (*i*) service description and modeling (i.e., what properties of a service should be described, and at which abstraction level)), (*ii*) service discovery (i.e., how to efficiently query against service descriptions), (*iii*) service composition (i.e., how to specify goals and constraints of a composition, how to build a composition, how to analyze a composition), and (*iv*) orchestration and choreography (i.e., invocation, enactment and monitoring of both simple and composite services) [PG03, CLM03, ACKM04].

Existing web services today support operations, which are limited to independent calls over a network, hard-wired collaboration or predefined integration scenarios. Built on existing paradigms from distributed artificial intelligence and object-oriented approaches, Service Oriented Architectures (SOA) are being promoted in the industry as the next evolutionary step in software architectures [B$^+$03]. In general, SOA is an application architecture in which functions are defined as independent services with well-defined invocable interfaces that can be called in sequences to form business processes. Built on SOA concepts, the Semantic Web as well as Web Services technologies, Semantic Web Services (SWS) is the most recent approach to semantic-enabled system integration [C$^+$04]. SWS identifies concepts, architectures and technologies to facilitate a process of intelligent services discovery, selection, composition and invocation in a distributed environment. Nowadays, SWS are subject of extensible research. Among several proposals, the most interesting are WSMO [dB$^+$04], OWL-S [The05], IRS [M$^+$03], and METEOR-S [SMV04].

6.5.3 The role of Ontologies

In SWS, ontologies are well suited for the description of the web services provided by each node in a networking environment and for the conceptualization of a shared understanding of the underlying SOA. Therefore, reasoning over ontologies in order to achieve service discovery, selection, composition and invocation in a distributed environment turns out to be a crucial and challenging issue. In this respect, for example, query reformulation techniques over ontologies in the P2P scenario mentioned in the above subsection, can be also seen as providing a service-oriented architecture, where the algorithms aim at computing the "composition" of the query answering services provided by the peers. This problem is tackled in [TAK03]. However, a deep investigation on this matter is still missing. Furthermore, in spite of the notion of service is considered crucial in the Semantic Web community, there is still a fundamental lack of understanding on how to integrating ontologies that describe static information (possibly by means of a rich DL) with dynamic processes as described by services. First results in this directions are reported in [BCDG$^+$05b, BLM$^+$05, PMBT05].

6.6 Open Problems and Future Research Directions

Many research problems related to the achievement ontology interoperation still need a deep investigation and a clear understanding, as well as proper and effective solutions. Below we list some of those that we consider more crucial and challenging:

- The studies on *query answering*, in both (mediator-based) ontology integration systems, i.e., mediator-based systems characterized by a global schema expressed in terms of an ontology, and in ontology-based P2P systems, i.e., P2P systems in which each peer exports an ontology to its clients. Both semantic and computational problems related to query answering need to be addressed, in order to establish which is the more appropriate semantic characterization of the problem, provide query answering techniques (sound and complete with respect to such characterization), and establish computational complexity of the problem. In this respect, a desirable property of query answering is that it is solvable in efficient, also in those cases in which the extension of the ontology is constituted by a large amount of data. Therefore, cases for which query answering stays polynomial with respect to data complexity, i.e., complexity computed with respect to the size of the data (e.g., the size of the ABox in those cases in which the ontology is specified in terms of a Description Logic) assume particular relevance. On the other hand, since both queries and data schemas (TBoxes in the case of DLs ontologies) are in general not huge, query answering procedures which are exponential w.r.t. the size of the schema and/or the query are in general acceptable [CDGL$^+$05b].

- The problem of *ontology update*, which presents semantic and computational complexity problems similar to the ones described for query answering. Ontology update is largely uninvestigated also for the setting of a single ontology. In the case of multiple interoperating ontologies, however, a further difficulty is deciding how to propagate an update among the different ontologies (e.g., from the one with which the user interacts to the other interconnected ontologies).

- The problem of *inconsistency*, i.e., dealing with situations in which interoperating ontologies are mutually inconsistent. In particular, in the context of P2P semantic interoperability, inconsistency between interoperating ontology-based peers may arise for different reasons: an ontology may result locally inconsistent because its data (possibly coming from local sources) violate assertions specified at intensional level (e.g., a TBox for a DL ontology); data (and knowledge) coming into a peer from other peers may contradict assertions when combined with data locally managed by the peer; data coming into a peer from different peers may result mutually inconsistent, i.e., combined together may violate integrity constraints of the peer schema. It is immediate to verify that inconsistency dramatically affects the ability of the system of providing meaningful answers to queries.

- The problem of *information hiding or authorization*, i.e., controlling the accessibility of the data and the knowledge of an ontology. Control on how to access data has been recently studied under a logical perspective in the context of a single database [ZM05]. The proposed method is based on the idea of specifying, for each

user, a set of authorized views, representing the information that the user is allowed to access. Users still express their queries in terms of the whole database schema (and not of the views), but only queries that can be completely rewritten using such views are answered by the system. We think that this idea nicely captures the logical essence of access control, and should be somehow transferred in the context of ontology interoperation (e.g., in semantic P2P setting, where it is crucial that each peer is able to specify data privacy policy in a declarative way).

- The problem of *coordination between dynamic and static behavior* in Semantic Web Services. In spite of the notion of service is considered crucial in the Semantic Web community, there is still a fundamental lack of understanding on how to integrating ontologies that describe static information (possibly by means of a rich DL) with dynamic processes as described by services. First results in this directions are reported in [BCDG$^+$05b, BLM$^+$05, PMBT05].

References

[AAHM00] E. Agirre, O. Ansa, E. H. Hovy, and D. Martínez. Enriching very large ontologies using the WWW. In *Proc. of ECAI 2000 Workshop on Ontology Learning*, 2000.

[ABC99] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99)*, pages 68–79, 1999.

[ABFL04] M. Arenas, P. Barcelo, R. Fagin, and L. Libkin. Locally consistent transformations and query answering in data exchange. In *Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004)*, pages 229–240, 2004.

[Abr74] J. R. Abrial. Data semantics. In J. W. Klimbie and K. L. Koffeman, editors, *Data Base Management*, pages 1–59. North-Holland Publ. Co., 1974.

[ACKM04] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services. Concepts, Architectures and Applications.* Springer, 2004.

[AdMRV02] P. Auillans, P. O. de Mendez, P. Rosenstiehl, and B. Vatant. A formal model for Topic Maps. In *Proc. of the 1st Int. Semantic Web Conf. (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2002.

[AF82] J. F. Allen and A. M. Frisch. What's in a Semantic Network? In *Proc. of the 20th Conf. of the Association for Computational Linguistics*, pages 19–27, Toronto (Canada), 1982.

[AH87] S. Abiteboul and R. Hull. IFO: A formal semantic database model. *ACM Trans. on Database Systems*, 12(4):297–314, 1987.

[AM02] E. Alfonseca and S. Manandhar. Extending a lexical ontology by a combination of distributional semantics signatures. In *Proc. of the 13th Int. Conf. on Knowledge Engineering and Knowledge Management – Ontologies and the Semantic Web (EKAW 2002)*, pages 1–7, 2002.

[APHS02] K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt. Improving data access in P2P systems. *IEEE Internet Computing*, 2002.

[B+01] K. Baclawski et al. Extending UML to support ontology engineering for the Semantic Web. In *Proc. of the 4th Int. Conf. on the Unified Modeling Language (UML 2001)*, pages 342–360, 2001.

[B+03] K. Brown et al. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions.* Addison Wesley Publ. Co., 2003.

[Baa91] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, 1991.

[BB03] L. Bravo and L. Bertossi. Logic programming for consistently querying data integration systems. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 10–15, 2003.

[BB04] L. Bertossi and L. Bravo. Consistent query answering under inclusion dependencies. In *Proc. of the Annual IBM Center for Advanced Studies Coference (CASCON 2004)*, 2004.

[BB05] L. Bravo and L. Bertossi. Disjunctive deductive databases for computing certain and consistent answers to queries from mediated data integration systems. *J. of Applied Logic – Special Issue on Logic-based Methods for Information Integration*, 3(2):329–367, 2005.

[BBK01] F. Baader, S. Brandt, and R. Küsters. Matching under side conditions in description logics. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 213–218, 2001.

[BC99] M. Berland and E. Charniak. Finding parts in very large corpora. In *Proc. of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 57–64, 1999.

[BCDG05a] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1–2):70–118, 2005.

[BCDG+05b] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic composition of transition-based Semantic Web services with messaging. In *Proc. of the 31st Int. Conf. on Very Large Data Bases (VLDB 2005)*, pages 613–624, 2005.

[BCM+03] 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.

[BCN92] C. Batini, S. Ceri, and S. B. Navathe. *Conceptual Database Design, an Entity-Relationship Approach*. Benjamin and Cummings Publ. Co., 1992.

[BDS93] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *J. of Artificial Intelligence Research*, 1:109–138, 1993.

[BFH+94] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *Applied Artificial Intelligence. Special Issue on Knowledge Base Management*, 4:109–132, 1994.

[BFH+99] A. Borgida, E. Franconi, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider. Explaining $\mathcal{ALC}$ subsumption. In *Proc. of the 1999 Description Logic Workshop (DL'99)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-22/`, 1999.

[BG04] D. Brickley and R. V. Guha. RDF vocabulary description language 1.0: RDF Schema – W3C recommendation. Technical report, World Wide Web Consortium, Feb. 2004. Available at `http://www.w3.org/TR/rdf-schema/`.

[BGK+02] 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.

[BGvHS03] P. Bouquet, F. Giunchiglia, F. van Harmelen, and L. Serafini. C-OWL: Contextualizing ontologies. In *Proc. of the 2nd Int. Semantic Web Conf. (ISWC 2003)*, 2003.

[BH91a] F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 452–457, 1991.

[BH91b] F. Baader and B. Hollunder. $\mathcal{KRIS}$: $\mathcal{K}$nowledge $\mathcal{R}$epresentation and $\mathcal{I}$nference $\mathcal{S}$ystem. *SIGART Bull.*, 2(3):8–14, 1991.

[BH92] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 306–317. Morgan Kaufmann, 1992.

[BH95a] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. *J. of Automated Reasoning*, 14:149–180, 1995.

[BH95b] F. Baader and B. Hollunder. Priorities on defaults with prerequisites and their application in treating specificity in terminological default logic. *J. of Automated Reasoning*, 14:41–68, 1995.

[BHGS01] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: A Reasonable ontology editor for the semantic web. In *Proc. of the Joint German/Austrian Conf. on Artificial Intelligence (KI 2001)*, number 2174 in Lecture Notes in Artificial Intelligence, pages 396–408. Springer, 2001. Appeared also in Proc. of the 2001 Description Logic Workshop (DL 2001).

[BHLW03] F. Baader, J. Hladik, C. Lutz, and F. Wolter. From tableaux to automata for description logics. *Fundamenta Informaticae*, 57:1–33, 2003.

[BHN+92] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 270–281. Morgan Kaufmann, 1992.

[BK98] F. Baader and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proc. of the 22nd German Annual Conf. on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1998.

[BK00] F. Baader and R. Küsters. Matching in description logics with existential restrictions. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 261–272, 2000.

[BK01] F. Baader and R. Küsters. Unification in a description logic with transitive closure of roles. In R. Nieuwenhuis and A. Voronkov, editors, *Proc. of the 8th Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2001)*, volume 2250 of *Lecture Notes in Computer Science*, pages 217–232. Springer, 2001.

[BKBM99] F. Baader, R. Küsters, A. Borgida, and D. L. McGuinness. Matching in description logics. *J. of Logic and Computation*, 9(3):411–447, 1999.

[BKFH00] J. Broekstra, M. Klein, D. Fensel, and I. Horrocks. Adding formal semantics to the Web: building on top of RDF Schema. In *Proc. of the ECDL 2000 Workshop on the Semantic Web*, 2000.

[BKM99] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 96–101, 1999.

[BKW03] F. Baader, R. Küsters, and F. Wolter. Extensions to description logics. In Baader et al. [BCM+03], chapter 6, pages 219–261.

[BL85] R. J. Brachman and H. J. Levesque, editors. *Readings in Knowledge Representation.* Morgan Kaufmann, 1985.

[BL01a] T. Berners Lee. Conceptual Graphs and the Semantic Web. Available at `http://www.w3.org/DesignIssues/CG.html`, 2001.

[BL01b] M. Bouzeghoub and M. Lenzerini. Introduction to the special issue on data extraction, cleaning, and reconciliation. *Information Systems*, 26(8):535–536, 2001.

[BLC96] A. Bernaras, I. Laresgoiti, and J. M. Corera. Building and reusing ontologies for electrical network applications. In *Proc. of the 12th Eur. Conf. on Artificial Intelligence (ECAI'96)*, pages 298–302, 1996.

[BLHL01] T. Berners Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[BLM+05] F. Baader, C. Lutz, M. Milicic, U. Sattler, and F. Wolter. Integrating description logics and action formalisms: First results. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 572–577, 2005.

[BLV03] S. Bechhofer, P. Lord, and R. Volz. Cooking the Semantic Web with the OWL API. In *Proc. of the 2nd Int. Semantic Web Conf. (ISWC 2003)*, 2003.

[BMC03] S. Bechhofer, R. Möller, and P. Crowther. The DIG description logic interface. In *Proc. of the 2003 Description Logic Workshop (DL 2003)*, 2003.

[BN01] F. Baader and P. Narendran. Unification of concepts terms in description logics. *J. of Symbolic Computation*, 31(3):277–305, 2001.

[BN03] F. Baader and W. Nutt. Basic description logics. In Baader et al. [BCM+03], chapter 2, pages 43–95.

[BPGL85] R. J. Brachman, V. Pigman Gilbert, and H. J. Levesque. An essential hybrid reasoning system: Knowledge and symbol level accounts in KRYPTON. In *Proc. of the 9th Int. Joint Conf. on Artificial Intelligence (IJCAI'85)*, pages 532–539, 1985.

[BPS94] A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.

[BPSM98] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0 — W3C recommendation. Technical report, World Wide Web Consortium, 1998. Available at `http://www.w3.org/TR/1998/REC-xml-19980210`.

[Bra79] R. J. Brachman. On the epistemological status of semantic networks. In N. V. Findler, editor, *Associative Networks*, pages 3–50. Academic Press, 1979. Republished in [BL85].

[Bra83] R. J. Brachman. What IS-A is and isn't. *IEEE Computer*, 16(10):30–36, 1983.

[BS01] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001.

[BS03] A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *J. on Data Semantics*, 1:153–184, 2003.

[BvHH+04] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language reference – W3C recommendation. Technical report, World Wide Web Consortium, Feb. 2004. Available at `http://www.w3.org/TR/owl-ref/`.

[C+04] L. Cabral et al. Approaches to Semantic Web services: An overview and comparisons. In *Proc. of the European Semantic Web Symposium (ESWS 2004)*, 2004.

[CCDG+03] A. Calì, D. Calvanese, G. De Giacomo, M. Lenzerini, P. Naggar, and F. Vernacotola. IBIS: Semantic data integration at work. In *Proc. of the 15th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2003)*, pages 79–94, 2003.

[CCDGL01] 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.

[CCDGL04] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data integration under integrity constraints. *Information Systems*, 29:147–163, 2004.

[CDDG+03] D. Calvanese, E. Damaggio, G. De Giacomo, M. Lenzerini, and R. Rosati. Semantic data integration in P2P systems. In *Proc. of the Int. Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2003)*, 2003.

[CDG03] D. Calvanese and G. De Giacomo. Expressive description logics. In Baader et al. [BCM+03], chapter 5, pages 178–218.

[CDGL98a] 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.

[CDGL+98b] 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.

[CDGL00] 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.

[CDGL01a] D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In *Proc. of the 2001 Int. Semantic Web Working Symposium (SWWS 2001)*, pages 303–316, 2001. Available at `http://www.semanticweb.org/SWWS/program/full/SWWSProceedings.pdf`.

[CDGL+01b] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Data integration in data warehousing. *Int. J. of Cooperative Information Systems*, 10(3):237–271, 2001.

[CDGL02a] D. Calvanese, G. De Giacomo, and M. Lenzerini. 2ATAs make DLs easy. In *Proc. of the 2002 Description Logic Workshop (DL 2002)*, pages 107–118. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-53/`, 2002.

[CDGL02b] 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.

[CDGL+04a] 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.

[CDGL+04b] 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.

[CDGL+05a] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-147/`, 2005.

[CDGL+05b] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[CDGLN01] D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi. Reasoning in expressive description logics. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 23, pages 1581–1634. Elsevier Science Publishers, 2001.

[CDGLR04] 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.

[CDGLV99a] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Vardi. Query answering using views for data integration over the web. In *Proc. of the 2nd Int. Workshop on the Web and Databases (WebDB'99)*, pages 73–78, 1999.

[CDGLV99b] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Rewriting of regular expressions and regular path queries. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99)*, pages 194–204, 1999.

[CDGLV00] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Query processing using views for regular path queries with inverse. In *Proc. of the 19th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2000)*, pages 58–66, 2000.

[CDGLV02] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Rewriting of regular expressions and regular path queries. *J. of Computer and System Sciences*, 64(3):443–465, 2002.

[CDGLV05] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing: On the relationship between rewriting, answering and losslessness. In *Proc. of the 10th Int. Conf. on Database Theory (ICDT 2005)*, volume 3363 of *Lecture Notes in Computer Science*, pages 321–336. Springer, 2005.

[CFF+98] V. K. Chaudhri, A. Farquhar, R. Fikes, P. D. Karp, and J. Rice. OKBC: A programmatic foundation for knowledge base interoperability. In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI'98)*, pages 600–607, 1998.

[CFFK98] V. K. Chaudhri, A. Farquhar, R. Fikes, and P. D. Karp. Open Knowledge Base Connectivity 2.0. Technical Report KSL-09-06, Stanford University Knowledge Systems Laboratory, 1998.

[CG05] B. Cuenca Grau. *Combination and Integration of Ontologies on the Semantic Web*. PhD thesis, Universidad de Valencia, 2005.

[CGMH+94] S. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of the 10th Meeting of the Information Processing Society of Japan (IPSJ'94)*, pages 7–18, 1994.

[CGPS05] B. Cuenca Grau, B. Parsia, and E. Sirin. Combining OWL ontologies using $\mathcal{E}$-connections. *J. of Web Semantics*, 2005. In Press.

[CH94] W. W. Cohen and H. Hirsh. Learning the CLASSIC description logics: Theorethical and experimental results. In *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 121–133, 1994.

[Che76] P. P. Chen. The Entity-Relationship model: Toward a unified view of data. *ACM Trans. on Database Systems*, 1(1):9–36, Mar. 1976.

[CHS+95] M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams, and E. L. Wimmers. Towards heterogeneous multimedia information systems: The Garlic approach. In *Proc. of the 5th Int. Workshop on Research Issues in Data Engineering – Distributed Object Management (RIDE-DOM'95)*, pages 124–131. IEEE Computer Society Press, 1995.

[CHS05] P. Cimiano, A. Hotho, and S. Staab. Learning concept hierarchies from text corpora using formal concept analysis. *J. of Artificial Intelligence Research*, 24:305–339, 2005.

[CHW05] C. Chen, V. Haarslev, and J. Wang. LAS: Extending Racer by a Large ABox Store. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-147/`, 2005.

[CL93] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *J. of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[CLM03] J. Chung, K. Lin, and R. Mathieu. Introduction to the special issue on web service computing. *IEEE Computer*, 36(10), 2003.

[CLN99] D. Calvanese, M. Lenzerini, and D. Nardi. Unifying class-based representation formalisms. *J. of Artificial Intelligence Research*, 11:199–240, 1999.

[CLR03] 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.

[CLRR04] A. Calì, D. Lembo, R. Rosati, and M. Ruzzi. Experimenting data integration with DIS@DIS. In *Proc. of the 16th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2004)*, volume 3084 of *Lecture Notes in Computer Science*, pages 51–56. Springer, 2004.

[CLS05] Common Logic Standard, 2005. Official ISO FCD draft available at `http://philebus.tamu.edu/cl/`.

[CM92] M. Chein and M.-L. Mugnier. Conceptual graphs: Fundamental notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.

[CMH+00] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. Freenet: A distributed anonymous information storage and retrieval system. In *Proc. of the Int. Workshop on Design Issues in Anonymity and Unobservability (DIAU 2000)*, 2000.

[CMS04a] J. Chomicki, J. Marcinkowski, and S. Staworko. Computing consistent query answers using conflict hypergraphs. In *Proc. of the 13th Int. Conf. on Information and Knowledge Management (CIKM 2004)*, pages 417–426, 2004.

[CMS04b] J. Chomicki, J. Marcinkowski, and S. Staworko. Hippo: a system for computing consistent query answers to a class of SQL queries. In *Proc. of the 9th Int. Conf. on Extending Database Technology (EDBT 2004)*, pages 841–844. Springer, 2004.

[Cod70] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[CS93] M. Cadoli and M. Schaerf. A survey of complexity results for non-monotonic logics. *J. of Logic Programming*, 17:127–160, 1993.

[dB+04] J. de Bruijn et al. D2V1.1. Web Service Modeling Ontology (WSMO). Available at `http://www.wsmo.org/2004/d2/v1.1/20041126/`, 2004.

[DEFS99] S. Decker, M. Erdmann, D. Fensel, and R. Studer. *Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information*. Kluwer Academic Publisher, 1999.

[DFvH+00] S. Decker, D. Fensel, F. van Harmelen, I. Horrocks, S. Melnik, M. Klein, and J. Broekstra. Knowledge representation on the web. In *Proc. of the 2000 Description Logic Workshop (DL 2000)*, pages 89–97. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-33/`, 2000.

[DG95] G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1995.

[DG97] O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, pages 109–116, 1997.

[DGL00] O. M. Duschka, M. R. Genesereth, and A. Y. Levy. Recursive query plans for data integration. *J. of Logic Programming*, 43(1):49–73, 2000.

[DGM00] G. De Giacomo and F. Massacci. Combining deduction and model checking into tableaux and algorithms for converse-PDL. *Information and Computation*, 160(1–2):117–137, 2000.

[DH05] A. Doan and A. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine*, 26(1):83–94, 2005.

[DLNN97] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134:1–58, 1997.

[DLNS91] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. A hybrid system integrating Datalog and concept languages. In *Proc. of the 2nd Conf. of the Ital. Assoc. for Artificial Intelligence (AI*IA'91)*, volume 549 of *Lecture Notes in Artificial Intelligence*. Springer, 1991. An extended version appeared also in the Working Notes of the AAAI Fall Symposium on "Principles of Hybrid Reasoning".

[DLNS94] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *J. of Logic and Computation*, 4(4):423–452, 1994.

[DLNS96] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[DM00] F. M. Donini and F. Massacci. EXPTIME tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, 124(1):87–138, 2000.

[DMDH02] A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Learning to map between ontologies on the Semantic Web. In *Proc. of the 11th Int. World Wide Web Conf. (WWW 2002)*, pages 662–673, 2002.

[Don03] F. M. Donini. Complexity of reasoning. In Baader et al. [BCM+03], chapter 3, pages 96–136.

[ES04] M. Ehrig and S. Staab. QOM - Quick Ontology Mapping. In *Proc. of the 3rd Int. Semantic Web Conf. (ISWC 2004)*, volume 3298 of *Lecture Notes in Computer Science*. Springer, 2004.

[Euz95] J. Euzenat. Building consensual knowledge bases: Context and architecture. In *Building and Sharing Large Knowledge Bases*, pages 143–155. IOP Press, 1995.

[Euz96] J. Euzenat. Cooperative memory through cooperative creation of knowledge bases and hyper-documents. In *Proc. of 10th KAW*, 1996.

[Fen01] D. Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2001.

[FFM05] A. Fuxman, E. Fazli, and R. J. Miller. ConQuer: Efficient management of inconsistent databases. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 155–166, 2005.

[FFMM94] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an agent communication language. In N. Adam, B. Bhargava, and Y. Yesha, editors, *Proc. of the 3rd Int. Conf. on Information and Knowledge Management (CIKM'94)*, pages 456–463, 1994.

[FFR96] A. Farquhar, R. Fikes, and J. Rice. The Ontolingua server: A tool for collaborative ontology construction. Technical Report KSL-96-26, Stanford University Knowledge Systems Laboratory, 1996.

[FGPJ97] M. Fernandez, A. Gomez-Perez, and N. Juristo. METHONTOLOGY: From ontological art towards ontological engineering. In *Proc. of the AAAI Symposium on Ontological Engineering*, pages 33–40, 1997.

[FHH04] R. Fikes, P. Hayes, and I. Horrocks. OWL-QL – a language for deductive query answering on the Semantic Web. *J. of Web Semantics*, 2(1):19–29, 2004.

[FJDP05] T. Finin, A. Joshi, L. Ding, and R. Pan. Swoogle manual, 2005. Available at `http://swoogle.umbc.edu/modules.php?name=Documents&file=manual`.

[FK85] R. Fikes and T. Kehler. The role of frame-based representation in reasoning. *Communications of the ACM*, 28(9):904–920, 1985.

[FKLS03] 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.

[FKMP03] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *Proc. of the 9th Int. Conf. on Database Theory (ICDT 2003)*, pages 207–224, 2003.

[FKMT05] A. Fuxman, P. G. Kolaitis, R. Miller, and W. C. Tan. Peer data exchange. In *Proc. of the 24rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2005)*, pages 160–171, 2005.

[FKP03] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*, pages 90–101, 2003.

[FL79] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *J. of Computer and System Sciences*, 18:194–211, 1979.

[FLM99] 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.

[FM05] A. Fuxman and R. J. Miller. First-order query rewriting for inconsistent databases. In *Proc. of the 10th Int. Conf. on Database Theory (ICDT 2005)*, volume 3363 of *LNCS*, pages 337–351. Springer, 2005.

[FN00] E. Franconi and G. Ng. The i.com tool for intelligent conceptual modeling. In *Proc. of the 7th Int. Workshop on Knowledge Representation meets Databases (KRDB 2000)*, pages 45–53. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-29/`, 2000.

[FW04] D. C. Fallside and P. Walmsley. XML Schema Part 0: Primer second edition — W3C recommendation. Technical report, World Wide Web Consortium, 2004. Available at `http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/`.

[GBMS99] C. H. Goh, S. Bressan, S. E. Madnick, and M. D. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Trans. on Information Systems*, 17(3):270–293, 1999.

[GF92] M. R. Genesereth and R. E. Fikes. Knowledge Interchange Format, version 3.0 reference manual. Technical Report Logic-92-1, Stanford University, 1992. Available at `http://www.cs.umbc.edu/kse/`.

[GF95] M. Grüninger and M. S. Fox. Methodology for the design and evaluation of ontologies. In *Proc. of the IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[GFH+03] J. Golbeck, G. Fragoso, F. Hartel, J. Hendler, B. Parsia, and J. Oberthaler. The National Cancer Institute's thesaurus and ontology. *J. of Web Semantics*, 1(1), 2003.

[GGMO03] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari. Sweetening WordNet with DOLCE. *AI Magazine*, 24(3):13–24, 2003.

[GHI+01] S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suciu. What can databases do for peer-to-peer? In *Proc. of the 4th Int. Workshop on the Web and Databases (WebDB 2001)*, 2001.

[Gin87] M. L. Ginsberg, editor. *Readings in Nonmonotonic Reasoning*. Morgan Kaufmann, 1987.

[GKD97] M. R. Genereseth, A. M. Keller, and O. M. Duschka. Infomaster: An information integration system. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 539–542, 1997.

[GLRR05] L. Grieco, D. Lembo, M. Ruzzi, and R. Rosati. Consistent query answering under key and exclusion dependencies: Algorithms and experiments. In *Proc. of the 14th Int. Conf. on Information and Knowledge Management (CIKM 2005)*, pages 792–799, 2005.

[GLW05] S. Ghilardi, C. Lutz, and F. Wolter. Did i damage my ontology? A case for conservative extensions in description logics. Technical report, Institute for Theoretical Computer Science. Dresden University of Technology, 2005.

[GM99] G. Grahne and A. O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 1999.

[GM03] M. Grüninger and C. Menzel. Process Specification Language: Theory and applications. *AI Magazine*, 24:63–74, 2003.

[GMF+03] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubezy, H. Eriksson, N. F. Noy, and S. W. Tu. The evolution of Protege: an environment for knowledge-based systems development. *Int. J. of Human-Computer Studies*, 58(1):89–123, 2003.

[GMPQ+97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *J. of Intelligent Information Systems*, 8(2):117–132, 1997.

[GPJP95] A. Gomez-Perez, N. Juristo, and J. Pazos. Evaluation and assessment of knowledge sharing technology. In N. Mars, editor, *Towards Very Large Knowledge Bases. Knowledge Building and Knowledge Sharing 1995*, pages 289–296. IOS Press, 1995.

[Gru93a] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publisher, 1993.

[Gru93b] T. R. Gruber. A translation approach to portable ontology specification. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Gru95] T. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *Int. J. of Human and Computer Studies*, 43(5/6):907–928, 1995.

[Grü03] M. Grüninger. Guide to the ontology of the Process Specification Language. In S. Staab and R. Studer, editors, *Handbook of Ontologies*, pages 575–592. Springer, 2003.

[Gua98] N. Guarino. Formal ontology in information systems. In *Proc. of the Int. Conf. on Formal Ontology in Information Systems (FOIS'98)*, Frontiers in Artificial Intelligence, pages 3–15. IOS Press, 1998.

[GW04] N. Guarino and C. A. Welty. An overview of OntoClean. In S. Staab and R. Studer, editors, *Handbook on Ontologies*. Springer, 2004.

[GWGvS04] G. Guizzardi, G. Wagner, N. Guarino, and M. van Sinderen. An ontologically well-founded profile for UML conceptual models. In *Proc. of the 16th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2004)*, pages 112–126, 2004.

[Hay04] P. Hayes. RDF semantics – W3C recommendation. Technical report, World Wide Web Consortium, Feb. 2004. Available at `http://www.w3.org/TR/rdf-mt/`.

[HBCS03] R. Hull, M. Benedikt, V. Christophides, and J. Su. E-services: a look behind the curtain. In *Proc. of the 22nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2003)*, pages 1–14. ACM Press and Addison Wesley, 2003.

[HH01] J. Heflin and J. Hendler. A portrait of the Semantic Web in action. *IEEE Intelligent Systems*, 16(2):54–59, 2001.

[HIST03] 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.

[HK87] R. B. Hull and R. King. Semantic database modelling: Survey, applications and research issues. *ACM Computing Surveys*, 19(3):201–260, Sept. 1987.

[HLTB04] I. Horrocks, L. Li, D. Turi, and S. Bechhofer. The Instance Store: DL reasoning with large numbers of individuals. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-104/`, 2004.

[HM81] M. Hammer and D. McLeod. Database description with SDM: A semantic database model. *ACM Trans. on Database Systems*, 6(3):351–386, 1981.

[HM01] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 161–168, 2001.

[HM04] V. Haarslev and R. Möller. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *Proc. of the 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2004)*, 2004.

[HMS05] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 466–471, 2005.

[Hor03] I. Horrocks. Implementation and optimisation techniques. In Baader et al. [BCM+03], chapter 9, pages 306–346.

[HPS98] I. Horrocks and P. F. Patel-Schneider. DL systems comparison. In *Proc. of the 1998 Description Logic Workshop (DL'98)*, pages 55–57. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-11/`, 1998.

[HPS99] I. Horrocks and P. F. Patel-Schneider. Optimizing description logic subsumption. *J. of Logic and Computation*, 9(3):267–293, 1999.

[HPSvH03] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[HS99] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 9(3):385–410, 1999.

[HS01] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204, 2001.

[HS05] I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 2005.

[HST99] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer, 1999.

[HST00] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In D. McAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000)*, volume 1831 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2000.

[JDM03] M. Jarrar, J. Demey, and R. Meersman. On using conceptual data modeling for ontology engineering. *J. on Data Semantics*, 1:185–207, 2003.

[JLVV99] M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis, editors. *Fundamentals of Data Warehouses*. Springer, 1999.

[JQC+00] M. Jarke, C. Quix, D. Calvanese, M. Lenzerini, E. Franconi, S. Ligoudistianos, P. Vassiliadis, and Y. Vassiliou. Metadata-driven management of data warehouses: The DWQ demonstrators. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, page 591, 2000.

[JU99] R. Jasper and M. Uschold. A framework for understanding and classifying ontology applications. In *Proc. of the 12th Workshop on Knowledge Acquisition Modeling and Management (KAW'99)*, 1999.

[KBF+03] M. Klein, J. Broekstra, D. Fensel, F. van Harmelen, and I. Horrocks. Ontologies and schema languages on the web. In D. Fensel, J. Hendler, and H. Lieberman, editors, *Spinning the Semantic Web*. The MIT Press, 2003.

[KC04] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax – W3C recommendation. Technical report, World Wide Web Consortium, Feb. 2004. Available at `http://www.w3.org/TR/rdf-concepts/`.

[KCT99] P. D. Karp, V. K. Chaudhri, and J. Thomere. XOL: An XML-based ontology exchange language. Technical Report SRI AI Technical Note 559, SRI International, Menlo Park (CA, USA), 1999.

[Ken99] R. E. Kent. Conceptual Knowledge Markup Language: The central core. In *Electronic Proc. of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW'99)*, 1999.

[Ken00] R. E. Kent. The information flow foundation for conceptual knowledge organization. In *Proc. of the 6th Int. ISKO Conference on Advances in Knowledge Organization 7*, pages 111–117. Ergon Verlag, 2000.

[KFKO02] M. C. A. Klein, D. Fensel, A. Kiryakov, and D. Ognyanov. Ontology versioning and change detection on the web. In *Proc. of the 13th Int. Conf. on Knowledge Engineering and Knowledge Management – Ontologies and the Semantic Web (EKAW 2002)*, volume 2473 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 2002.

[KKT76] L. Kerschberg, A. Klug, and D. Tsichritzis. A taxonomy of data models. In *Systems for Large Data Bases*, pages 43–64. North-Holland Publ. Co., 1976.

[KLSS95] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pages 85–91, 1995.

[KLW95] M. Kifer, G. Lausen, and J. Wu. Logical foundations of Object-Oriented and frame-based languages. *J. of the ACM*, 42(4):741–843, 1995.

[KLWZ04] O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. $\mathcal{E}$-Connections of Abstract Description Systems. *Artificial Intelligence*, 156(1):1–73, 2004.

[KM01] R. Küsters and R. Molitor. Approximating most specific concepts in description logics with existential restrictions. In F. Baader, G. Brewka, and T. Eiter, editors, *Proc. of the Joint German/Austrian Conf. on Artificial Intelligence (KI 2001)*, volume 2174 of *Lecture Notes in Artificial Intelligence*, pages 33–47. Springer, 2001.

[KMG95] P. D. Karp, K. L. Myers, and T. Gruber. The Generic Frame Protocol. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, pages 768–774, 1995.

[KN03] M. Klein and N. F. Noy. A component-based framework for ontology evolution. In *Proc. of IJCAI 2003 Workshop on Ontologies and Distributed Systems*, 2003.

[Koc02] C. Koch. Query rewriting with symmetric constraints. In *Proc. of the 2nd Int. Symp. on Foundations of Information and Knowledge Systems (FoIKS 2002)*, volume 2284 of *Lecture Notes in Computer Science*, pages 130–147. Springer, 2002.

[KPH05] A. Kalyanpur, B. Parsia, and J. Hendler. A tool for working with web ontologies. *Int. J. on Semantic Web and Information Systems*, 1(1):36–49, 2005.

[KPS+05] A. Kalyanpur, B. Parsia, E. Sirin, B. Cuenca Grau, and J. Hendler. SWOOP - a web ontology editing browser. *J. of Web Semantics*, 1(4), 2005.

[KPSH05] A. Kalyanpur, B. Parsia, E. Sirin, and J. Hendler. Debugging unsatisfiable classes in OWL ontologies. *J. of Web Semantics – Special Issue on the Semantic Web Track of WWW 2005*, 3(4), 2005.

[KS03a] Y. Kalfoglou and M. Schorlemmer. IF-Map: an ontology mapping method based on information flow theory. *J. on Data Semantics*, 1:98–127, 2003.

[KS03b] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: The state of the art. *The Knowledge Engineering Review*, 18:1–31, 2003.

[KSV02] O. Kupferman, U. Sattler, and M. Y. Vardi. The complexity of the graded mu-calculus. In *Proc. of the 18th Int. Conf. on Automated Deduction (CADE 2002)*, 2002.

[Kut04] O. Kutz. *$\mathcal{E}$-Connections and Logics of Distance*. PhD thesis, University of Liverpool, 2004.

[LAHS05] C. Lutz, C. Areces, I. Horrocks, and U. Sattler. Keys, nominals, and concrete domains. *J. of Artificial Intelligence Research*, 23:667–726, 2005.

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

[Len95] D. B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):32–38, 1995.

[Len02] 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.

[Lev00] A. Y. Levy. Logic-based techniques in data integration. In J. Minker, editor, *Logic Based Artificial Intelligence*. Kluwer Academic Publisher, 2000.

[LG90] D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison Wesley Publ. Co., 1990.

[LM05] C. Lutz and M. Milicic. A tableaux algorithm for description logics with concrete domains and gcis. In *Proc. of the 14th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2005)*, pages 201–216, 2005.

[LR98] A. Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[LRL+97] H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *J. of Logic Programming*, 31:59–84, 1997.

[LRO96] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogenous information sources using source descriptions. In *Proc. of the 22nd Int. Conf. on Very Large Data Bases (VLDB'96)*, 1996.

[LSK95] 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.

[Lut03] C. Lutz. Description logics with concrete domains: A survey. In P. Balbiani, N.-Y. Suzuki, F. Wolter, and M. Zakharyaschev, editors, *Advances in Modal Logics*, volume 4. King's College Publications, 2003.

[M+03] E. Motta et al. IRS II: A framework and infrastructure for Semantic Web services. In *Proc. of the 2nd Int. Semantic Web Conf. (ISWC 2003)*, 2003.

[Mae03] A. Maedche. *Ontology learning for the Semantic Web*. Kluwer Academic Publisher, 2003.

[MBDH02] J. Madhavan, P. A. Bernstein, P. Domingos, and A. Y. Halevy. Representing and reasoning about mappings between domain models. In *Proc. of the 18th Nat. Conf. on Artificial Intelligence (AAAI 2002)*, pages 80–86, 2002.

[MBJK90] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Trans. on Information Systems*, 8(4):325–362, 1990.

[MBW80] J. Mylopoulos, P. A. Bernstein, and H. K. T. Wong. A language facility for designing database-intensive applications. *ACM Trans. on Database Systems*, 5(2):185–207, 1980.

[McG96] D. L. McGuinness. *Explaining Reasoning in Description Logics*. PhD thesis, Department of Computer Science, Rutgers University, Oct. 1996. Also available as Rutgers Technical Report Number LCSR-TR-277.

[MH69] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of aritificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[MH03] R. Möller and V. Haarslev. Description logic systems. In Baader et al. [BCM+03], chapter 8, pages 282–305.

[Min75] M. Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, 1975.

[MLB05] T. Meyer, K. Lee, and R. Booth. Knowledge integration for description logics. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 645–650, 2005.

[MNV02] M. Missikoff, R. Navigli, and P. Velardi. Integrated approach to web ontology learning and engineering. *IEEE Computer*, 35(11):60–63, 2002.

[Mot98] E. Motta. An overview of the OCML modelling language. In *Proc. of the 8th Workshop on Knowledge Engineering Methods and Languages (KEML'98)*, 1998.

[MS00] A. Maedche and S. Staab. Semi-automatic engineering of ontologies from text. In *Proc. of the 12th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE 2000)*, 2000.

[MS01] A. Maedche and S. Staab. Ontology learning for the Semantic Web. *IEEE Intelligent Systems*, 2(16), 2001.

[MT03] M. Missikoff and F. Taglino. SymOntoX: A web-ontology tool for eBusiness domains. In *Proc. of the 4th Int. Conf. on Web Information Systems Engineering (WISE 2003)*, pages 343–346, 2003.

[MW02] P. Mitra and G. Wiederhold. Resolving terminological heterogeneity in ontologies. In *Proc. of the ECAI 2002 Workshop on Ontologies and Semantic Interoperability*, 2002.

[Neb90] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer, 1990.

[NH97] N. Noy and C. Hafner. The state of the art in ontology design. A survey and comparative review. *AI Magazine*, 18(3):53–74, 1997.

[NK04] N. F. Noy and M. C. A. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6(4):428–440, 2004.

[NM03] N. Noy and M. Musen. The PROMPT suite: Interactive tools for ontology mapping and merging. *Int. J. of Human-Computer Studies*, 59:983–1024, 2003.

[Noy04] N. Noy. Semantic integration: A survey on ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.

[NP01] I. Niles and A. Pease. Towards a standard upper ontology. In *Proc. of the 2nd Int. Conf. on Formal Ontology in Information Systems (FOIS 2001)*, 2001.

[PDYP05] R. Pan, Z. Ding, Y. Yu, and Y. Peng. A Bayesian Network approach to ontology mapping. In *Proc. of the 4th Int. Semantic Web Conf. (ISWC 2005n)*, 2005.

[PG03] M. Papazoglou and D. Georgakopoulos. Introduction to the special issue on service oriented computing. *Communications of the ACM*, 46(10):24–28, 2003.

[PH03] J. Pan and I. Horrocks. RDFS(FA) and RDF MT: Two semantics for RDFS. In *Proc. of the 2nd Int. Semantic Web Conf. (ISWC 2003)*, volume 2870 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2003.

[PKY03] M. P. Papazoglou, B. J. Kramer, and J. Yang. Leveraging Web-services and peer-to-peer networks. In *Proc. of the 15th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2003)*, pages 485–501, 2003.

[PMBT05] M. Pistore, A. Marconi, P. Bertoli, and P. Traverso. Automated composition of web services by planning at the knowledge level. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 1252–1259, 2005.

[Pol02] J. Pollock. Integration dirty little secret: It's a matter of semantics. Whitepaper, Modulant, the Interoperability Company, Feb. 2002.

[PSHH04] P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language semantics and abstract syntax – W3C recommendation. Technical report, World Wide Web Consortium, Feb. 2004. Available at `http://www.w3.org/TR/owl-semantics/`.

[Qui68] M. R. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*, pages 216–270. The MIT Press, 1968.

[RB01] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *Very Large Database J.*, 10(4):334–350, 2001.

[Rei84] R. Reiter. Towards a logical reconstruction of relational database theory. In M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, editors, *On Conceptual Modeling: Perspectives from Artificial Intelligence Databases and Programming Languages*. Springer, 1984.

[Rei01] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, 2001.

[RJB98] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison Wesley Publ. Co., 1998.

[Ros99] R. Rosati. Towards expressive KR systems integrating Datalog and description logics: Preliminary report. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 160–164. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-22/`, 1999.

[SC03] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies (extended abstract). In *Proc. of the 15th Belgium-Netherlands Conf. on Artificial Intelligence*, 2003.

[SCC97] K. A. Spackman, K. E. Campbell, and R. A. Cote. SNOMED RT: A reference terminology for health care. *J. of the American Medical Informatics Association*, pages 640–644, 1997. Fall Symposium Supplement.

[Sch91] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.

[Sch93] A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. of Intelligent Information Systems*, 2:265–278, 1993.

[SCM03] U. Sattler, D. Calvanese, and R. Molitor. Relationship with other formalisms. In Baader et al. [BCM+03], chapter 4, pages 137–177.

[SEA+02] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology development for the Semantic Web. In *Proc. of the 1st Int. Semantic Web Conf. (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science*, pages 221–235. Springer, 2002.

[SG00] L. Serafini and C. Ghidini. Using wrapper agents to answer queries in distributed information systems. In *Proc. of the 1st Int. Conf. on Advances in Information Systems (ADVIS-2000)*, volume 1909 of *Lecture Notes in Computer Science*. Springer, 2000.

[Shi81] D. W. Shipman. The functional data model and the data language DAPLEX. *ACM Trans. on Database Systems*, 6(1):140–173, 1981.

[SM01] G. Stumme and A. Maedche. FCA-Merge: Bottom-up merging of ontologies. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 225–234, 2001.

[SMV04] A. Sheth, J. Miller, and K. Verma. METEOR-S: Semantic Web services and processes. Available at `http://lsdis.cs.uga.edu/Projects/METEOR-S/`, 2004.

[Sow84] J. F. Sowa. *Conceptuals Structures: Information Processing in Mind and Machine*. Addison Wesley Publ. Co., 1984.

[SPCG+05] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. Technical report, University of Maryland Institute for Advanced Computer Studies (UMIACS), 2005.

[SPKR97] B. Swartout, R. Patil, K. Knight, and T. Russ. Towards distributed use of large-scale ontologies. In *Proc. of the AAAI Symposium on Ontological Engineering*, 1997.

[SSS91] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Sta96] M. Stanley. CML: A knowledge representation language with applications to requirements modeling. Master's thesis, Dept. of Computer Science, University of Toronto, 1996.

[SV01] U. Sattler and M. Y. Vardi. The hybrid μ-calculus. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, pages 76–91, 2001.

[SW03] J. Siméon and P. Wadler. The essence of XML. In *Proc. of the 30th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL 2003)*, pages 1–13, 2003.

[SWM04] M. K. Smith, C. Welty, and D. L. McGuiness. OWL Web Ontology Language guide – W3C recommendation. Technical report, World Wide Web Consortium, Feb. 2004. Available at `http://www.w3.org/TR/owl-guide/`.

[TAK03] S. Thakkar, J.-L. Ambite, and C. A. Knoblock. A view integration approach to dynamic composition of Web services. In *Proc. of 2003 ICAPS Workshop on Planning for Web Services*, 2003.

[TCS01] Y. Tzitzikas, P. Constantopoulos, and N. Spyratos. Mediators over ontology-based information sources. In *Proc. of the 2nd Int. Conf. on Web Information Systems Engineering (WISE 2001)*, pages 31–40, 2001.

[The05] The OWL Services Coalition. OWL-S: Semantic markup for web services, 2005. Available at `http://www.daml.org/services/owl-s/`.

[TK04] A.-Y. Turhan and C. Kissig. SONIC—Non-standard inferences go OILED. In *Proc. of the 2nd Int. Joint Conf. on Automated Reasoning (IJCAR 2004)*, volume 3097 of *Lecture Notes in Computer Science*, pages 321–325. Springer, 2004.

[Tob01] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2001.

[Top02] ISO/IEC 13250, Topic Maps, 2nd edition. Available at `http://www.y12.doe.gov/sgml/sc34/document/0322_files/iso13250-2%nd-ed-v2.pdf`, 2002.

[TRV98] A. Tomasic, L. Raschid, and P. Valduriez. Scaling access to heterogeneous data sources with DISCO. *IEEE Trans. on Knowledge and Data Engineering*, 10(5):808–823, 1998.

[UG96] M. Uschold and M. Grüninger. Ontologies: Principles, methods and applications. *Knowledge Sharing and Review*, 11(2):93–155, 1996.

[UG04] M. Uschold and M. Grüninger. Ontologies and semantics for seamless connectivity. *SIGMOD Record*, 33(4):58–64, 2004.

[UK95] M. Uschold and M. King. Towards a methodology for building ontologies. In *Proc. of the IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[Ull97] 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.

[Ull00] J. D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.

[UML05] Unified Modeling Language (UML) superstructure, version 2.0. Available at `http://www.uml.org/`, Aug. 2005.

[Var82] M. Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, pages 137–146, 1982.

[Var85] M. Y. Vardi. The taming of converse: Reasoning about two-way computations. In R. Parikh, editor, *Proc. of the 4th Workshop on Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 413–424. Springer, 1985.

[VW84] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. In *Proc. of the 16th ACM SIGACT Symp. on Theory of Computing (STOC'84)*, pages 446–455, 1984.

[VW86] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. of Computer and System Sciences*, 32:183–221, 1986.

[WDKF02] S. R. Waterhouse, D. M. Doolin, G. Kan, and Y. Faybishenko. Distributed search in P2P networks. *IEEE Internet Computing*, 6(1):68–72, 2002.

[WM05] M. Wessel and R. Möller. A high performance Semantic Web query answering engine. In *Proc. of the 2005 Description Logic Workshop (DL 2005)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-147/`, 2005.

[ZM05] Z. Zhang and A. Mendelzon. Authorization views and conditional query containment. In *Proc. of the 10th Int. Conf. on Database Theory (ICDT 2005)*, 2005. To appear.