Reasoning with Ontologies

Meghyn Bienvenu, Michel Leclère, Marie-Laure Mugnier, Marie-Christine Rousset

▶ **To cite this version:**

Meghyn Bienvenu, Michel Leclère, Marie-Laure Mugnier, Marie-Christine Rousset. Reasoning with Ontologies. A Guided Tour of Artificial Intelligence Research, pp.185-215, 2020, Volume I: Knowledge Representation, Reasoning and Learning, 978-3-030-06163-0. 10.1007/978-3-030-06164-7_6 . lirmm-02922020

HAL Id: lirmm-02922020
https://hal-lirmm.ccsd.cnrs.fr/lirmm-02922020

Submitted on 25 Aug 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Reasoning with Ontologies

Meghyn Bienvenu, Michel Leclère, Marie-Laure Mugnier, and Marie-Christine Rousset

This chapter is part of the following collective book: *A Guided Tour of Artificial Intelligence Research*, Volume I: Knowledge Representation, Reasoning and Learning. Pierre Marquis, Odile Papini and Henri Prade (Eds). Springer, Cham, 2020.
https://doi.org/10.1007/978-3-030-06164-7

Abstract This chapter considers the notion of a formal ontology, which is a conceptual vocabulary equipped with a logical semantics. Three families of knowledge representation and reasoning formalisms that put ontologies at the core of any knowledge base are presented, namely: description logics, conceptual graphs and existential rules. We present the main knowledge constructs and dialects of these families, as well as the main reasoning problems with their complexity. We highlight the relationships between these families and compare them from an expressivity viewpoint.

1 Introduction

Knowledge-based systems exploit formal representations of knowledge to solve different kinds of problems. The fundamental formalism to represent and do reasoning on knowledge is classical first-order logic. Whereas a significant amount of work in knowledge representation aimed to extend classical logic to handle more complex notions (like time, modalities, preferences, ...), most work on ontologies was devoted to simpler logical fragments and to the study of tradeoffs between the ex-

Meghyn Bienvenu
LIRMM-CNRS and Université de Montpellier- Inria, Montpellier, France,
e-mail: meghyn@lirmm.fr

Michel Leclère
LIRMM-CNRS and Université de Montpellier- Inria, Montpellier, France,
e-mail: leclere@lirmm.fr

Marie-Laure Mugnier
LIRMM-CNRS and Université de Montpellier- Inria, Montpellier, France,
e-mail: mugnier@lirmm.fr

Marie-Christine Rousset
LIG-CNRS and Université de Grenoble, Grenoble, France,
e-mail: Marie-Christine.Rousset@imag.fr

pressivity of the representation languages and the computational complexity of reasoning in these languages.

A commonly adopted definition of an *ontology* is that of an explicit specification of the conceptualisation of a domain [Gruber, 1993]. All ontologies include at least a conceptual vocabulary, i.e., a set of terms (in the natural language sense) used to model a domain, provided with a specification of their meaning. These terms represent *concepts* (or classes), i.e., the categories of entities in the modeled domain, as well as *relations* (or properties, roles) which may stand between entities. Concepts and relations may be further specified in different ways, depending on the expressivity of the ontological language. They are usually organized into a specialisation/generalisation hierarchy by means of axioms stating that a concept (respectively a relation) is a subconcept (respectively a subrelation) of another. Other typical ontological axioms include concept disjointness (which expresses that two concepts cannot have common instances), the domain and range of binary relations (which specifies the classes of entities that can be linked by this relation), algebraic properties of relations (for instance that a relation is symmetric or transitive), mandatory relations for instances of a class (for instance that every entity of a given class fulfils a given property), and so on.

Ontologies are widely used in data and knowledge management and they are at the core of the Semantic Web (see Chapter 6 of Volume 3). We refer the reader to the chapter on knowledge engineering (Chapter 23 of this volume) for developments on building and using ontologies.

Without denying the importance of the linguistic aspects in ontologies, we focus in this chapter on formal ontologies. Therefore, an ontology will be seen as a logical theory that specifies the expected meaning of the conceptual vocabulary [Guarino, 1998]. More specifically, an ontology is given by a formal vocabulary (or signature) and a set of formulas built on this vocabulary, which define the acceptable models of the considered domain. Hence, any reasoning that takes into account an ontology $\mathcal{O}$ considers only the models of $\mathcal{O}$: for instance, given two pieces of knowledge G and F, deciding if G is a logical consequence of F, which we denote by $F \models G$, becomes $\mathcal{O}, F \models G$, i.e., is every model of $\mathcal{O}$ and F a model of G? Moreover, in most settings, the *unique name assumption* is made: in this case, distinct logical constants are necessarily interpreted by distinct elements of the domain of any interpretation.

In this chapter, we consider *knowledge bases* composed of two types of knowledge: on the one hand, ontological knowledge, which is general knowledge about the modelled domain, and factual knowledge, composed of facts or assertions about specific entities. Usually, a fact is a ground atom, i.e., has no variable. [1]

A parallel can be drawn between a knowledge base and a classical database (e.g., a relational database). Indeed, the database schema, which includes a vocabulary and integrity constraints, can be associated with an ontology, while data can be seen as factual knowledge. However, some important differences should be noted. In databases, data are supposed to encode a complete description of the 'world'. In other words, the *closed world assumption* is made (everything that is not asserted

[1] In the Semantic Web area, and specifically concerning the OWL language, the term ontology often includes both kinds of knowledge. Hence, it corresponds to our notion of knowledge base.

in the database is considered as false), as well as the related closed domain assumption (the only existing entities are those encoded in the data). By contrast, the *open world assumption* is made in knowledge bases (as well as the related open domain assumption); this often leads to more complex reasoning since a knowledge base encodes a possibly infinite set of all of the descriptions of the world that include the known facts and comply with the ontology. For that reason, the use of negation is often restricted, as the excluded-middle law (stating that a proposition is either true or false) leads to combinatorial explosion. The open world assumption may lead to considering existentially quantified variables in facts (and not only constants) to denote unknown individuals. Moreover, the primary aim of databases is to store and retrieve data with efficient query answering techniques, whereas knowledge bases are used to infer new knowledge that was only implicitly represented in the ontology. However, the two domains are becoming progressively closer, especially under the impulse of the Semantic Web. Indeed, there is an increasing interest in answering complex queries on large knowledge bases, on the one hand, and, on the other hand, dropping the closed world assumption in databases.

This chapter is devoted to several knowledge representation and reasoning formalisms used to build and exploit knowledge bases: description logics, graph-based representations (issued from conceptual graphs) and the more recent existential rule framework. Although description logics and graph-based representations are both rooted in semantic networks [Lehmann, 1992], their development from the 80's followed different research lines, as explained in the next sections. Existential rules can be seen both as the logical counterpart of the graph-based framework and as a generalisation of Datalog, the deductive database querying language.

Several different kinds of reasoning over knowledge bases have been considered, among which we distinguish the following fundamental problems. Given a knowledge base (KB) composed of an ontology $\mathcal{O}$ and a set of facts I, we consider the following questions:

- *Knowledge base satisfiability:* determine if the KB is satisfiable (or consistent), i.e., if it has at least one model.
- *Ontological knowledge entailment:* determine if a piece of ontological knowledge o is entailed by the ontology $\mathcal{O}$, i.e., if $\mathcal{O} \models o$ holds.
- *Fact entailment:* determine if a fact is entailed by the KB, i.e., if $\mathcal{O}, I \models o$ holds.
- *Ontology-mediated query answering:* compute the answers to a query q over the KB; when q is a Boolean query (i.e., a query with a yes/no answer), the problem is whether q is entailed by the KB, i.e., whether $\mathcal{O}, I \models q$ holds. The general form of a query q is a first-order formula with possibly free variables, say $(x_1, \ldots, x_k)$. Then an answer to q in the KB is a tuple of constants $(c_1, \ldots, c_k)$ such that the Boolean query obtained from q by substituting each variable x_i by the constant c_i is entailed by the KB.

The three formalisms presented in this chapter tackle the above problems, the difference being in their expressivity and the kind of query considered. Description logics traditionally allowed for rich descriptions of ontological axioms using different kinds of constructors; standard reasoning tasks were KB satisfiability, concept

subsumption (determine if a concept is a specialisation of another, which is a special case of ontological knowledge entailment) and instance checking (determine if a specific individual is an instance of a given concept, which is a special case of fact entailment). Hence, only very specific queries were considered (single atoms without variables). The growing interest for exploiting large and complex data led the description logic community to investigate more expressive queries, however at the price of less expressive description logics, known as lightweight description logics. The queries most commonly considered so far in the context of ontology-mediated query answering are so-called *conjunctive queries*, which are the basic queries in databases: these are existentially quantified conjunctions of atoms. Conjunctive queries are natural queries in the graph-based and existential rule frameworks, but, on the other hand, these formalisms do not offer the variety of ontological axioms found in classical description logics. Some lightweight description logics, however, can be seen as special cases of the graph-based and existential rule frameworks.

The sequel of this chapter introduces each of these three formalisms and compares them from an expressivity viewpoint.

2 Description Logics

Description logics (DLs) [Baader et al., 2003, 2017] are family of knowledge representation languages corresponding to decidable[2] fragments of first-order logic using only unary and binary predicates. While the lack of higher-arity predicates may seem a strong restriction, it turns out that unary and binary predicates (classes and properties) capture a large part of modelling needs. Indeed, DLs provide the basis of the OWL Web Ontology Language [W3C, 2004a], a W3C-standardized ontology language for the Semantic Web [Berners-Lee et al., 2001], and RDF [W3C, 2004b], a popular format for Web data, is likewise restricted to unary and binary predicates.

A DL knowledge base (KB) has two parts: a *TBox* that contains general knowledge about the application domain, and an *ABox* that contains facts about particular individuals. The TBox can be viewed as an ontology, which provides a conceptual model for the data stored in the ABox. What distinguishes different DLs is the type of knowledge that can be expressed in the TBox.

Traditionally, the main reasoning problems considered by the DL community are: KB satisfiability, subsumption, and instance checking. Satisfiability testing is essential for identifying modelling errors, while instance checking and subsumption are used to identify TBox axioms and ABox assertions that follow from the knowledge of the KB. As the latter two tasks correspond to forms of logical entailment, they can be reduced to unsatisfiability testing for all DLs that admit full negation. Our discussion of DLs will center on these traditional reasoning tasks. However, we should point out that over the past decade, several additional reasoning tasks for DLs have

[2] A few undecidable DLs have been studied.

been investigated, most notably, ontology-mediated conjunctive query answering, which allows for richer queries to be posed over the ABox, but which cannot be reduced to satisfiability testing and thus required the development of new algorithmic techniques (see survey [Bienvenu and Ortiz, 2015] and references therein). There has also been quite a lot of work on reasoning support for building, debugging, and evolving ontologies, e.g., providing explanations for why a given entailment holds [Schlobach and Cornet, 2003; Sebastiani and Vescovi, 2009; Peñaloza and Sertkaya, 2017], or extracting modules of an ontology that conform to some criterion [Grau et al., 2008; Kontchakov et al., 2010; Konev et al., 2013].

Early work on DLs in the 1980's mostly focused on building reasoning systems, and it was only later that it was discovered that some of these DLs were in fact undecidable or at the very least intractable. These initial negative results led to the introduction of simple DLs for which polynomial-time reasoning was possible, but which turned out to be too limited in their expressivity. In the late 1990's, however, new systems were developed based upon highly optimized tableaux algorithms, which demonstrated acceptable performance for expressive DLs despite their high worst-case complexity. This line of work continues to this day, with ever more sophisticated optimisations targeting ever more expressive DLs. At the same time, there has been renewed interest in lightweight DLs that provide the required scalability for applications involving very large TBoxes and/or ABoxes. Importantly, however, this new breed of low-complexity DLs provides combinations of modelling constructs that are much better suited to the needs of real-world applications than the previous generation of simple DLs.

Nowadays, there is an extensive body of results pinpointing the exact computational complexity of performing different kinds of reasoning in the whole range of DLs, allowing one to choose the optimal trade-off between expressivity and efficiency of reasoning for the application at hand. For an overview of the complexity landscape, interested readers can consult the surveys [Ortiz and Simkus, 2012] and [Bienvenu and Ortiz, 2015].

In this section, we introduce the basics of description logics, and then present several concrete DLs and show how varying the expressivity of the DL impacts the complexity of reasoning.

2.1 Preliminaries: DL syntax and semantics

In DL jargon, classes are called *concepts* and properties are called *roles*. DL knowledge bases are built starting from a set $\mathsf{N_C}$ of *atomic concepts* (unary predicates), a set $\mathsf{N_R}$ of *atomic roles* (binary predicates), and a set $\mathsf{N_I}$ of *individuals* (constants). We typically use $A, B, \ldots$ for atomic concepts, $P, Q, \ldots$ for atomic roles, and $a, b, \ldots$ for individuals. More complex concept and role expressions can be built using different constructors, with the set of available constructors depending on the particular DL (see further for more details). We will use $C, D, \ldots$ to denote (possibly complex) concepts and R, S for (possibly complex) roles.

A DL *knowledge base* $\mathcal{K}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$, consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. A *TBox* is a finite set of axioms expressing the relationships holding between different concepts and roles. The types of axioms allowed in the TBox depends on the choice of DL, but the most common forms of TBox axioms are *concept inclusions* ($C \sqsubseteq D$, with C, D possibly complex concepts) and *role inclusions* ($R \sqsubseteq S$, with R, S possibly complex roles). Equivalences between concepts ($C \equiv D$) and roles ($R \equiv S$) are also common and can be seen as shorthand for inclusions in both directions (i.e., $C \equiv D$ is an abbreviation for the pair of inclusions $C \sqsubseteq D$ and $D \sqsubseteq C$).

An *ABox* is a finite set of *assertions* expressing that an individual belongs to a given concept ($C(a)$) or that a pair of individuals belongs to a role ($R(a,b)$). To simplify the presentation, we will assume in what follows that ABoxes only contain assertions involving atomic concepts and roles. This assumption can usually be made without loss of generality. For example, if we want to include $C(a)$ in the ABox, with C a general concept, it suffices to use the atomic assertion A_C (with A_C a fresh atomic concept) and add the inclusion $C \equiv A_C$ to the TBox.

The semantics of DL knowledge bases is defined in terms of (first-order) interpretations. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to every atomic concept $A \in \mathsf{N_C}$, a binary relation $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to every atomic role $P \in \mathsf{N_R}$, and an element $a^{\mathcal{I}}$ to every individual $a \in \mathsf{N_I}$. It is common in DLs to adopt the *unique name assumption*, which states that distinct individuals are mapped to distinct elements of the interpretation domain.

An interpretation $\mathcal{I}$ *satisfies* a concept inclusion $C \sqsubseteq D$ (resp. role inclusion $R \sqsubseteq S$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (resp. $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$). We say $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if it satisfies every axiom in $\mathcal{T}$. A TBox $\mathcal{T}$ logically implies an axiom α, written $\mathcal{T} \models \alpha$, if every model of $\mathcal{T}$ satisfies α. A fundamental reasoning task for TBoxes is *testing subsumption* between different concepts: given a TBox $\mathcal{T}$ and two concepts C and D, decide whether $\mathcal{T} \models C \sqsubseteq D$.

An interpretation $\mathcal{I}$ *satisfies* a concept assertion $A(a)$ (resp. role assertion $P(a,b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$). We call $\mathcal{I}$ a model of an ABox $\mathcal{A}$ if $\mathcal{I}$ satisfies every assertion in $\mathcal{A}$. An interpretation $\mathcal{I}$ is a model of a knowledge base $\langle \mathcal{T}, \mathcal{A} \rangle$ if it is a model of both $\mathcal{T}$ and $\mathcal{A}$. A KB $\mathcal{K}$ is *satisfiable* (or *consistent*) if it possesses at least one model. Testing satisfiability of a given KB is another standard reasoning task.

A KB *logically entails* a TBox axiom or ABox assertion α, written $\mathcal{K} \models \alpha$, if every model of $\mathcal{K}$ satisfies α. The *instance checking problem* is defined as follows: given a KB $\langle \mathcal{T}, \mathcal{A} \rangle$, a concept C, and an individual a, decide whether $\mathcal{K} \models C(a)$.

In the following sections, we give the syntax and semantics of the principal DL constructors by presenting a variety of DLs, ranging from 'simple' DLs $\mathcal{EL}$, $\mathcal{FL}_0$, and DL-Lite which offer polynomial-time reasoning (Sections 2.2 and 2.3), to $\mathcal{ALC}$ (Section 2.4) which is often considered the prototypical DL, to highly expressive DLs like $\mathcal{SROIQ}$ (Section 2.5), which provide the logical foundations for OWL.

2.2 Lightweight description logics: $\mathcal{FL}_0$ and $\mathcal{EL}$

We begin by considering two DLs, $\mathcal{FL}_0$ and $\mathcal{EL}$, which are deliberately restricted in expressivity in order to allow for sound and complete polynomial-time reasoning. Both logics contain the *concept conjunction* constructor ($C_1 \sqcap C_2$), which corresponds to intersecting the classes represented by C_1 and C_2. Additionally, $\mathcal{FL}_0$ offers *qualified value restrictions* ($\forall R.C$), while $\mathcal{EL}$ offers *qualified existential restrictions* ($\exists R.C$), which provide suitably restricted forms of universal and existential quantification. In $\mathcal{EL}$, one can further use the top concept ($\top$), which denotes the class of all objects.

Figure 1 provides an example of a taxonomy of classes, formulated using a set of inclusions between atomic concepts. Such simple axioms form the backbone of real-world ontologies, and they are available in every DL (and in particular, in $\mathcal{FL}_0$ and $\mathcal{EL}$). The axioms in the first line of Figure 1 stipulate that professors and teaching assistants are both kinds of teaching staff, and every teaching assistant is a graduate student. The remaining axioms state that teaching staff and admin staff are two types of staff and that students (resp. courses) can be specialized into undergraduate and graduate students (resp. courses).

$$\begin{array}{lll} \mathsf{Prof} \sqsubseteq \mathsf{TeachingStaff} & \mathsf{TAssistant} \sqsubseteq \mathsf{TeachingStaff} & \mathsf{TAssistant} \sqsubseteq \mathsf{GradStudent} \\ \mathsf{TeachingStaff} \sqsubseteq \mathsf{Staff} & \mathsf{AdminStaff} \sqsubseteq \mathsf{Staff} & \mathsf{UndergradStudent} \sqsubseteq \mathsf{Student} \\ \mathsf{GradStudent} \sqsubseteq \mathsf{Student} & \mathsf{UndergradCourse} \sqsubseteq \mathsf{Course} & \mathsf{GradCourse} \sqsubseteq \mathsf{Course} \end{array}$$

Fig. 1 Example taxonomy of classes in the university domain

In $\mathcal{FL}_0$ and $\mathcal{EL}$, we can additionally use the conjunction constructor to state that every student that is part of the teaching staff must be a graduate student:

$$\mathsf{Student} \sqcap \mathsf{TeachingStaff} \sqsubseteq \mathsf{GradStudent}$$

By making use of qualified value restrictions, we can express in $\mathcal{FL}_0$ that graduate students only take graduate courses and that a student that takes only graduate courses is a graduate student:

$$\mathsf{GradStudent} \sqsubseteq \forall \mathsf{takes}.\mathsf{GradCourse} \qquad \mathsf{Student} \sqcap \forall \mathsf{takes}.\mathsf{GradCourse} \sqsubseteq \mathsf{GradStudent}$$

In $\mathcal{EL}$, we can use qualified existential restrictions to formulate the following axioms:

$$\begin{array}{rr} \mathsf{Student} \sqsubseteq \exists \mathsf{takes}.\mathsf{Course} & \exists \mathsf{teaches}.\top \sqsubseteq \mathsf{TeachingStaff} \\ \exists \mathsf{teaches}.\mathsf{GradCourse} \sqsubseteq \mathsf{Prof} & \mathsf{TeachingStaff} \sqsubseteq \exists \mathsf{teaches}.\mathsf{Course} \end{array}$$

which state respectively that every student must take some course, that everyone who teaches something is a member of the teaching staff, that everyone who teaches

a graduate course must be a professor, and that every member of teaching staff must teach some course.

The semantics of complex concepts built using the preceding constructors is defined recursively as follows (starting from the semantics of atomic concepts and roles which is directly provided by each interpretation):

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
- $(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
- $(\exists R.C)^{\mathcal{I}} = \{o_1 \mid \text{there exists } (o_1, o_2) \in R^{\mathcal{I}} \text{ such that } o_2 \in C^{\mathcal{I}}\}$
- $(\forall R.C)^{\mathcal{I}} = \{o_1 \mid (o_1, o_2) \in R^{\mathcal{I}} \text{ implies } o_2 \in C^{\mathcal{I}}\}$

Every DL concept can be translated into a first-order logic formula with one free variable. Figure 2 gives the first-order translation of concepts in $\mathcal{FL}_0$ and $\mathcal{EL}$, using X as the free variable. To improve readability, we commit a slight abuse of notation by using $C(X)$ to designate the translation of the concept C using free variable X. For example, if $C = A \sqcap \exists R.B$, with A and B atomic concepts, then $C(X)$ is the FOL formula $A(X) \wedge \exists Y[R(X,Y) \wedge B(Y)]$.

DL notation	*Corresponding FOL formula*
$C_1 \sqcap C_2$	$C_1(X) \wedge C_2(X)$
$\exists R.C$	$\exists Y[R(X,Y) \wedge C(Y)]$
$\forall R.C$	$\forall Y[R(X,Y) \rightarrow C(Y)]$

Fig. 2 Translation from DLs to first-order logic (1)

Every TBox axiom can be translated into a corresponding FOL sentence (that is, a formula without free variables): a concept inclusion $C \sqsubseteq D$ gives rise to the formula $\forall X(C(X) \rightarrow D(X))$ and an equivalence axiom $C \equiv D$ corresponds to the formula $\forall X(C(X) \leftrightarrow D(X))$.

The first polynomial-time reasoning procedures for lightweight DLs relied upon *structural subsumption*, in which concept expressions are first put into a normal form and then compared syntactically. This method can be used to show that subsumption between concepts w.r.t. an empty TBox is tractable in both $\mathcal{FL}_0$ and $\mathcal{EL}$. However, in most applications, one wishes to compute subsumption in the presence of a non-empty TBox. Rather interestingly, $\mathcal{FL}_0$ and $\mathcal{EL}$ exhibit dramatically different complexities for the general version of subsumption. In $\mathcal{FL}_0$, the problem becomes EXPTIME-complete (and thus provably intractable) [Baader et al., 2005] and remains coNP-hard even when restricted to TBoxes in the form of acyclic terminologies [Nebel, 1990]. By contrast, in $\mathcal{EL}$ (and several of its extensions), subsumption can be decided in PTIME in the presence of arbitrary TBoxes [Baader et al., 2005]. This tractability result relies upon forward-chaining algorithms that construct in an iterative manner a subset of the axioms that are entailed from the TBox. A similar approach can be used to handle instance checking in $\mathcal{EL}$.

Nowadays, $\mathcal{EL}$ and its extensions have become popular ontology languages, whereas $\mathcal{FL}_0$ is no longer much in use. This is due in large part to the much more

favourable computational properties of $\mathcal{EL}$, but also to the utility of the constructors provided by $\mathcal{EL}$. Indeed, while it was initially believed that value restrictions were more useful than existential restrictions, it turns out that (slight extensions of) $\mathcal{EL}$ closely match the modelling needs of many applications, particularly those in the biomedical domain. Indeed, the large-scale professional medical ontology SNOMED CT[3] (Systematized Nomenclature of Medicine, Clinical Terms), developed by an international consortium for use in the health-care systems of several countries, is expressed in a tractable DL of the $\mathcal{EL}$ family. The importance of $\mathcal{EL}$ is further witnessed by the inclusion of the OWL 2 EL profile [W3C, 2012b], based upon $\mathcal{EL}$, in the latest version of the W3C OWL standard.

2.3 DL-Lite: Another lightweight description logic

The DL-Lite family of description logics [Calvanese et al., 2007] was proposed in the mid-2000's with the aim of supporting tractable reasoning while at the same time capturing the principal modelling primitives from conceptual modelling (more precisely, the entity-relationship models utilized in databases and information systems [Chen, 1976]) and UML[4] diagrams from software engineering). Another important motivation for introducing the DL-Lite family was to make it possible to answer more expressive queries by means of a reduction to relational databases.

In DL-Lite, complex concepts and roles can be constructed from atomic concepts and roles according to the following syntax:

$$B ::= A \mid \exists R \qquad C ::= B \mid \neg B \qquad R ::= P \mid P^- \qquad E ::= R \mid \neg R$$

where A is an atomic concept, P is an atomic role, and P^- is the *inverse* of P. Here B is called a *basic concept*, and C is a *general concept*. Likewise, we have *basic roles* R and *general roles* E. For completeness, we formally state the semantics of non-atomic concepts and roles:

- $(P^-)^{\mathcal{I}} = \{(o_2, o_1) \mid (o_1, o_2) \in P^{\mathcal{I}}\}$
- $(\exists R)^{\mathcal{I}} = \{o_1 \mid \text{there exists } o_2 \text{ and } (o_1, o_2) \in R^{\mathcal{I}}\}$
- $(\neg B)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}}$ and $(\neg R)^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$

Figure 3 gives the corresponding logical formulas. Note that when we write $R(X,Y)$, we mean $P(X,Y)$ if R is an atomic role P and $P(Y,X)$ if R is the inverse role P^-.

There are several different DL-Lite dialects, each allowing for different TBox axioms. In the core DL-Lite dialect, TBoxes are comprised of concept inclusions $B \sqsubseteq C$, where B is a basic concept and C a general concept. Observe that since only basic concepts are allowed on the left-hand side of inclusions, negation can only occur on the right-hand side.

---

[3] http://www.snomed.org/snomed-ct

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

DL notation	*Corresponding FOL formula*
P^-	$P(Y,X)$
$\exists R$	$\exists Y R(X,Y)$
$\neg B$	$\neg B(X)$
$\neg R$	$\neg R(X,Y)$

Fig. 3 Translation from DLs to first-order logic (2)

$\exists \mathsf{teaches} \sqsubseteq \mathsf{TeachingStaff}$ $\exists \mathsf{teaches}^- \sqsubseteq \mathsf{Course}$
$\exists \mathsf{coordinatorFor} \sqsubseteq \mathsf{Prof}$ $\exists \mathsf{coordinatorFor}^- \sqsubseteq \mathsf{Course}$
$\exists \mathsf{takes} \sqsubseteq \mathsf{Student}$ $\exists \mathsf{takes}^- \sqsubseteq \mathsf{Course}$
$\mathsf{coordinatorFor} \sqsubseteq \mathsf{teaches}$

Fig. 4 Domain and range constraints and role inclusions in DL-Lite

To illustrate the expressive power of DL-Lite (core), we return to our running example of the university domain. We first remark that all of the atomic concept inclusions from Figure 1 can be expressed in DL-Lite. Figure 4 displays DL-Lite axioms that constrain the *domain* and *range* of the roles teaches, takes, and coordinatorFor:

- if X teaches Y, then X is a member of teaching staff and Y is a course
- if X takes Y, then X is a student and Y is a course
- if X coordinatorFor Y, then X is a professor Y is a course

In DL-Lite, we can also express *disjointness constraints*, stating that two classes or properties have no elements in common, as well as *mandatory participation constraints*, requiring that elements of a certain class appear in the first or second components of a given binary relation. For example, the following axioms express that Student and AdminStaff are disjoint classes, that every professor must teach something, and that every course is taught by someone:

$$\mathsf{Student} \sqsubseteq \neg \mathsf{AdminStaff} \qquad \mathsf{Prof} \sqsubseteq \exists \mathsf{teaches} \qquad \mathsf{Course} \sqsubseteq \exists \mathsf{teaches}^-$$

We remark that if we replaced the inclusion $\mathsf{Student} \sqsubseteq \neg \mathsf{AdminStaff}$ by $\mathsf{Student} \sqsubseteq \neg \mathsf{Staff}$, then this would lead to an anomaly in the ontology. Indeed, using the atomic concept inclusions in Figure 1, we would be able to infer that teaching assistants belong to the class Staff (from $\mathsf{TAssistant} \sqsubseteq \mathsf{TeachingStaff}$ and $\mathsf{TeachingStaff} \sqsubseteq \mathsf{Staff}$) as well as to its complement $\neg\mathsf{Staff}$ (using $\mathsf{TAssistant} \sqsubseteq \mathsf{GradStudent}$, $\mathsf{GradStudent} \sqsubseteq \mathsf{Student}$, and $\mathsf{Student} \sqsubseteq \neg\mathsf{Staff}$). This would mean that TAssistant must always be interpreted as the empty class, and thus that including even a single instance of TAssistant in the ABox would lead to an inconsistent KB.

Two other common dialects, DL-Lite$_\mathcal{R}$ and DL-Lite$_\mathcal{F}$, offer additional TBox axioms. The former allows for *role inclusions* of the form $R \sqsubseteq E$, while the latter authorizes *functionality axioms* of the form $(funct\ R)$, where R is a basic role, i.e., without negation. For example, the following two axioms are expressible in DL-Lite$_\mathcal{R}$ and DL-Lite$_\mathcal{F}$ respectively:

$$\mathsf{coordinatorFor} \sqsubseteq \mathsf{teaches} \qquad (\mathit{funct}\ \mathsf{coordinatorFor}^{-})$$

The first axiom expresses that when X coordinatorFor Y, then X teaches Y, while the second one expresses that the role coordinatorFor^{-} is functional: if Y coordinatorFor X and Z coordinatorFor X then Y=Z.

A role inclusion $R \sqsubseteq E$ is satisfied in an interpretation $\mathcal{I}$ if $R^{\mathcal{I}} \subseteq E^{\mathcal{I}}$, and a functionality statement $(\mathit{funct}\ R)$ is satisfied in $\mathcal{I}$ if the binary relation $R^{\mathcal{I}}$ is a function, i.e. $(o,o_1) \in R^I$ and $(o,o_2) \in R^I$ implies that $o_1 = o_2$.

It has been shown in [Calvanese et al., 2007] that in both DL-Lite$_{\mathcal{R}}$ and DL-Lite$_{\mathcal{F}}$, satisfiability, subsumption, and instance checking can all be performed in polynomial time (more precisely, these tasks are NLOGSPACE-complete). Rather surprisingly, however, if we consider the minimal DL that extends both DL-Lite$_{\mathcal{R}}$ and DL-Lite$_{\mathcal{F}}$, then satisfiability testing becomes EXPTIME-complete [Artale et al., 2009]. Moreover, in both DL-Lite$_{\mathcal{R}}$ and DL-Lite$_{\mathcal{F}}$, it is possible to answer conjunctive queries in polynomial time in the size of the ABox by means of a reduction to the problem of answering first-order queries over relational databases, whereas no such reduction is possible if both role inclusions and functionality axioms are allowed [Calvanese et al., 2007]. (Ontology-mediated conjunctive query answering and the technique of first-order query rewriting will be discussed in more detail in Sections 3 and 4.)

The importance of the DL-Lite family of DLs is witnessed by the inclusion of the OWL 2 QL profile [W3C, 2012b], based upon DL-Lite$_{\mathcal{R}}$, in the OWL 2 standard, which is specifically designed to be the ontology language of choice for applications involving querying of large amounts of data.

2.4 $\mathcal{ALC}$: The prototypical description logic

The description logic $\mathcal{ALC}$ can be seen as the result of adding (full) *concept negation* to $\mathcal{EL}$. In $\mathcal{ALC}$, it is possible to construct the disjunction (or union) of two concepts $C_1 \sqcup C_2$ (which is just shorthand for $\neg(\neg C_1 \sqcap \neg C_2)$), qualified value restrictions (since $\forall R.C$ is equivalent to $\neg(\exists R.\neg C)$), and the bottom concept $\bot$ (corresponding to $\neg\top$, which is always interpreted as the empty set).

$\mathcal{ALC}$ is often considered to be the prototypical DL because it is a fragment of a natural first-order logic (allowing the standard Boolean operators plus restricted forms of universal and existential quantification) and because $\mathcal{ALC}$ concepts correspond precisely to the formulas expressible in the basic multi-modal logic K_n [Blackburn et al., 2006].

Returning to our university example, we first note since $\mathcal{ALC}$ extends both $\mathcal{FL}_0$ and $\mathcal{EL}$, all axioms from Section 2.2 can be expressed in $\mathcal{ALC}$. Additionally, the new constructors available in $\mathcal{ALC}$ allow us to express *disjointness* constraints, as in DL-Lite, and *covering* constraints, e.g., that every course is either an undergraduate course or a graduate course:

$$\mathsf{Student} \sqsubseteq \neg \mathsf{Prof} \qquad \mathsf{Course} \sqsubseteq \mathsf{UndergradCourse} \sqcup \mathsf{GradCourse}$$

For completeness, we formally specify the semantics of the new constructors:

- $\bot^{\mathcal{I}} = \emptyset$
- $\neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$

The problem of testing satisfiability of $\mathcal{ALC}$ KBs has been shown to be EXPTIME-complete [Schild, 1991]. The same result holds for subsumption testing and instance checking, which can be straightforwardly reduced to (un)satisfiability:

- $\mathcal{T} \models C \sqsubseteq D$ iff the KB $\langle \mathcal{T} \cup \{A_C \equiv C, A_{\neg D} \equiv \neg D\}, \{A_C(a), A_{\neg D}(a)\}\rangle$ is unsatisfiable
- $\mathcal{K} \models C(a)$ iff the KB $\mathcal{K} \cup \{A_{\neg C} \equiv \neg C\}, \{A_{\neg C}(a)\}$ is unsatisfiable.

To determine whether a given $\mathcal{ALC}$ KB is satisfiable, one can use *tableaux algorithms*, which work by exploring in an exhaustive manner all ways of constructing a model of the KB. If a (compact representation of a) model is found, the KB is satisfiable, and if all attempts fail, then one can conclude that the KB is unsatisfiable.

2.5 From $\mathcal{ALC}$ to $\mathcal{SHIQ}$ to $\mathcal{SROIQ}$: Highly expressive DLs

The description logic $\mathcal{ALC}$ is the starting point for defining other (highly) expressive DLs. For example, the DL $\mathcal{SHIQ}$ [Horrocks et al., 1999] extends $\mathcal{ALC}$ with *inverse roles* (P^-, as in DL-Lite), (qualified) *cardinality restrictions* ($\geq nR.C$, $\leq nR.C$), *role inclusions* ($R \sqsubseteq R'$), and *transitive roles* (using transitivity axioms of the form $(Trans\ R)$), where R, R' can be either plain or inverse roles.

In our university example, we could use cardinality restrictions to express that every professor must teach at least 2 courses, and students that take at most 3 courses are part-time students:

$$\mathsf{Prof} \sqsubseteq \geq 2\mathsf{teaches.Course} \quad \mathsf{Student} \sqcap \leq 3\mathsf{takes.Course} \sqsubseteq \mathsf{PartTimeStudent}$$

The even more expressive $\mathcal{SROIQ}$ [Horrocks et al., 2006], which provides the logical underpinnings of OWL 2 (the latest version of OWL standard) [W3C, 2012a], extends $\mathcal{SHIQ}$ with *nominals* ($\{a\}$), the universal role (u), and more *complex role axioms* of the forms $R \circ S \sqsubseteq R$ and $S \circ R \sqsubseteq R$ (where $\circ$ denotes role composition). It further allows for roles to be declared as reflexive, irreflexive, or antisymmetric, and for pairs of role to be declared disjoint.

The semantics of the new constructors is as follows:

- $(\geq nP.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \sharp\{e \mid (d,e) \in P^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\} \geq n\}$
- $(\leq nP)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \sharp\{e \mid (d,e) \in P^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}}\} \leq n\}$
- $\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$
- $u^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

- $(R \circ S)^{\mathcal{I}} = \{(d_1, d_3) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid$ there exists $d_2 \in \Delta^{\mathcal{I}}$ such that $(d_1, d_2) \in R^{\mathcal{I}}$ and $(d_2, d_3) \in S^{\mathcal{I}}\}$

An axiom of the form $(Trans\ R)$ is satisfied in $\mathcal{I}$ if $R^{\mathcal{I}}$ is a transitive relation, i.e., $(d_1, d_2) \in R^{\mathcal{I}}$ and $(d_2, d_3) \in R^{\mathcal{I}}$ implies $(d_1, d_3) \in R^{\mathcal{I}}$. TBox axioms declaring roles to be reflexive, irreflexive, or antisymmetric are handled analogously.

In the DL $\mathcal{SHIQ}$, the standard reasoning tasks (satisfiability, subsumption, and instance checking) are all EXPTIME-complete, and thus of the same complexity as in $\mathcal{ALC}$. For the DL $\mathcal{SHOIQ}$ obtained by adding nominals to $\mathcal{SHIQ}$, the complexity rises to NEXPTIME-complete, and for $\mathcal{SRIQ}$, which extends $\mathcal{SHIQ}$ with complex role inclusions and additional types of axioms concerning roles, the problem becomes 2EXPTIME-complete. If we move all the way up to $\mathcal{SROIQ}$, then reasoning becomes even more difficult (2NEXPTIME-complete).

The preceding complexity results show that automated reasoning with (highly) expressive DLs like $\mathcal{ALC}$, $\mathcal{SHIQ}$, and $\mathcal{SROIQ}$ may require (doubly) exponential time in the worst case. However, in practice, modern DL reasoners[5], mainly employing highly optimized tableaux algorithms, demonstrate acceptable performance for reasonably-sized ontologies. The reason is that the type of ontological constraints that are needed to model even complex real-world applications do not give rise to the pathological combinations of constructors that are required for establishing the negative complexity results.

Finally, several proposals have been made to overcome a limitation of description logics, which is the tree-shaped structure of terminological descriptions, in particular by combining them with Datalog rules. These proposals impose restrictions on the interaction between DL axioms and datalog rules, as early work has shown undecidability of standard reasoning if no restriction was made [Levy and Rousset, 1998]. The existential rules presented in Section 4 can be seen as another way of overcoming the tree-shaped structure limitation of description logics.

3 Conceptual Graphs

Conceptual graphs [Sowa, 1976, 1984] are mainly rooted in semantic networks, natural language processing and Peirce's existential graphs, a diagrammatical system of logic alternative to predicate logic. They have been studied along different directions. One research line consists in developing conceptual graphs as a graphical interface to first-order logic. Another research line follows the existential graph approach: conceptual graphs are then seen as diagrams, rather than graphs in the graph-theoretic meaning, and inferences are based on diagrammatic operations that do not aim to be automated (see, in particular, [Dau, 2003]). A third research line, which is the one presented in this chapter, develops conceptual graphs as a knowledge representation and reasoning formalism, equipped with its own reasoning mechanisms. This formalism is both graph- and logic-based: on the one hand, the basic objects

[5] See http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/ for an up-to-date list of DL reasoners.

are labelled graphs and reasoning is based on graph operations, with graph homomorphism at the core; on the other hand, these objects have a logical semantics and reasoning mechanisms are sound and complete with respect to this semantics. This approach to conceptual graphs is similar to the description logic approach in the sense that it defines and studies a family of formalisms that offer different tradeoffs between expressivity and complexity of reasoning. However, we will see in Section 3.5 that the logical fragments studied in both families are quite different. The interested reader will find an in-depth presentation of theoretical and algorithmic results on conceptual graphs in [Chein and Mugnier, 2009]. All aspects presented here are implemented in the software tools CoGUI[6] and CoGITaNT[7].

3.1 The Kernel: Basic conceptual graphs

A basic conceptual graph (BG) defines entities and relationships among these entities. Hence, it is a bipartite graph: one class of nodes, called *concept* nodes, represents entities, and the other class, called *relation* nodes, represent relationships among these entities. Nodes are labelled according to a vocabulary, which contains a set of concept types and a set of relation symbols, with both sets being partially ordered by specialisation. This vocabulary can be seen as a lightweight ontology, which can be further enriched by rules and constraints in more complex conceptual graph fragments.

3.1.1 Syntax

A *vocabulary*, also called a *support*, is a triple (T_C, T_R, I) where:

- T_C is a finite set of *concept types*, partially ordered by $\leq$, and provided with a greatest element, denoted by $\top$;
- T_R is a finite set of *relation symbols* (or simply relations) of any arity, partially ordered by $\leq$, such that only relations with the same arity are comparable;
- I is a possibly infinite set of elements called *individual markers*; furthermore, the symbol $*$ denotes the *generic marker*, with $* \notin I$. The set of all markers $I \cup \{*\}$ is partially ordered by $\leq$ as follows: for all $m \in I$, $m \leq *$, and elements in I are pairwise incomparable.
- T_C, T_R and I are pairwise disjoint sets.

The partial orders on T_C and T_R encode a specialisation relation, i.e., $t' \leq t$ means that "t' is a specialisation of t". Figure 5 pictures a set of concept types, which correspond to the set of DL inclusions from Figure 1 except for the part in italics. Each individual marker refers to a specific and distinct entity (i.e., the unique name assumption is made) and the generic marker refers to an unspecified entity.

[6] `http://www.lirmm.fr/cogui`

[7] `http://cogitant.sourceforge.net`

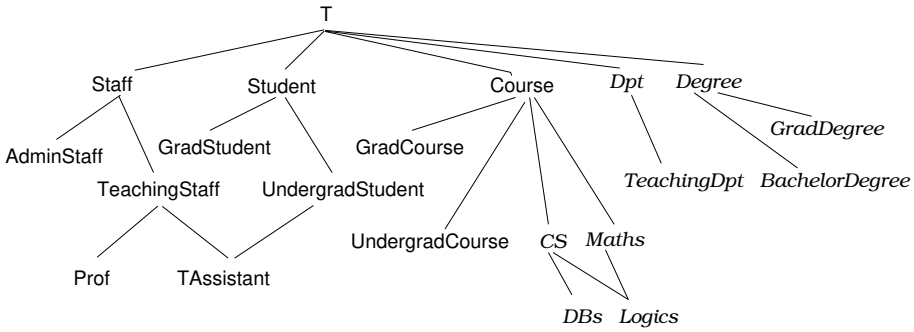

Fig. 5 A set of concept types

A *basic conceptual graph* (BG) $G = (C_G, R_G, E_G, l_G)$ defined over a support $S = (T_C,\ T_R,\ I)$ is a finite, labelled, undirected and bipartite multigraph (i.e., there may be several edges between two nodes), where C_G is the set of concept nodes, R_G is the set of relation nodes, E_G is the multiset of edges, and l_G is a labelling function of the nodes and edges that satisfies the following conditions:

- each concept node c is labelled by a concept type t_c and a marker m such that $(t_c, m) \in T_C \times (I \cup \{*\})$; if $m = *$, c is called *generic*, otherwise it is *individual*;
- each relation node r is labelled by a relation $t_r \in T_R$ and the number of edges incident to r is equal to the arity of t_r; these edges are labelled from 1 to the arity of t_r; we denote by $(c_1 \dots c_k)$ the list of arguments of r, where c_j denotes the extremity of the j-th edge incident to r.

Note that a BG is not necessarily connected. By convention, concept nodes are pictured as rectangles and relation nodes as ovals. For instance, the BG H pictured in Figure 6 may represent the following knowledge "there is a professor coordinator for a course on databases and a course on logics in the graduate degree MSc IA". A BG can also be seen as a hypergraph, with the relations being encoded by hyperedges. Then the graph view of a BG corresponds to the incidence bipartite graph of this hypergraph.

The notion of a support can be extended to allow for multi-instantiation. Then a concept node is labelled by a set of concept types, called a *conjunctive type*, instead of a single concept type. For instance, a course can be both a course in mathematics and a course in computer science, which is denoted by the conjunctive type $\{Maths, CS\}$. The set of concept types T_C is then defined in intension by a partially ordered set of primitive types, and its elements are all the conjunctive types that can be built with from primitive types. The partial order on conjunctive types is the natural extension of the order on primitives types: given conjunctive types t_1 and t_2, $t_2 \leq t_1$ if for all primitive type t_{1_i} in t_1, there is a primitive type t_{2_j} in t_2 with $t_{2_j} \leq t_{1_i}$. For instance, $\{Logics\} \leq \{CS, Maths\}$ (note that $\{CS, Maths\} \not\leq \{Logics\}$).

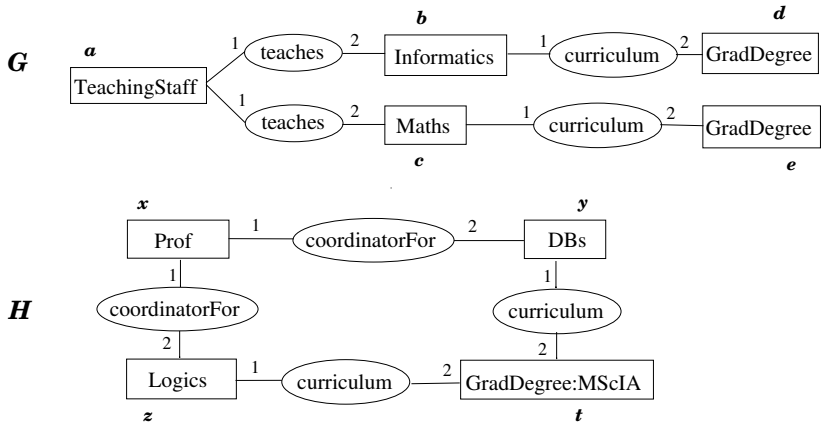

Fig. 6 Basic Conceptual Graphs

3.1.2 Semantics

This basic formalism is provided with a semantics in first-order logic by a translation denoted by Φ. Concept types are translated into unary predicates, relation symbols into predicates with the same arity and individual markers into constants (for simplicity, the same names are used for the elements of the support and their translation). To a support S is assigned a set of formulas $\Phi(S)$ that translates the partial orders on T_C and T_R, i.e., if $t_2 \leq t_1$, one has the formula $\forall x_1 \ldots x_k \; (t_2(x_1 \ldots x_k) \rightarrow t_1(x_1 \ldots x_k))$, where k is the arity of predicates t_1 and t_2.

A BG is translated into a formula $\Phi(G)$ built as follows: to each concept node is assigned a term, which is a new variable if its marker is generic, otherwise the constant assigned to its individual marker; to each relation (resp. concept) node is assigned an atom $t(e_1, \ldots, e_k)$ where t is the predicate assigned to its label (resp. concept type) and $(e_1, \ldots, e_k)$ is the list of terms assigned to its arguments (resp. the term assigned to the concept node); $\Phi(G)$ is then the existential closure of the conjunction of these atoms. For instance, for H in Figure 6: $\Phi(H) = \exists x \exists y \exists z (Prof(x) \wedge DBs(y) \wedge Logics(z) \wedge coordinatorFor(x,y) \wedge coordinatorFor(x,z) \wedge curriculum(y, MScIA) \wedge curriculum(z, MScIA) \wedge GradDegree(MScIA))$.

The BG fragment is equivalent to the existential, positive and conjunctive fragment of first-order logic (without functional symbols except for constants). Indeed, a polynomial translation of the support and BGs, which preserves logical entailment, allows one to obtain a "flat" support S' for which $\Phi(S') = \emptyset$.

The fundamental notion for reasoning on BGs is a *homomorphism* (often called "projection" in the conceptual graph community). A homomorphism from a BG G to a BG H is a mapping from $C_G \cup R_G$ to $C_H \cup R_H$ that preserves the node bipartition and the edges, and may specialize node labels, i.e.,

- for all relation $r \in R_G$ with arguments $(c_1 \dots c_k)$, $h(r)$ has arguments $(h(c_1) \dots h(c_k))$ (equivalently: for all edge rc in G, there is an edge $h(r)h(c)$ with the same label in H) ;
- for all node $x \in C_G \cup R_G$, $l_H(h(x)) \leq l_G(x)$ (for concept nodes one considers the product order on $T_C \times (I \cup \{\star\})$, i.e., $(t,m) \leq (t',m')$ if $t \leq t'$ and $m \leq m'$).

Consider the graphs in Figure 6 and assume that *coordinatorFor* $\leq$ *teaches* is the only comparable pair of distinct relations: there are two homomorphisms from G to H. The first one maps concept nodes in this way: $a \mapsto x$, $b \mapsto y$, $c \mapsto z$, $d \mapsto t$, $e \mapsto t$; each relation node *teaches* is mapped to a relation node *coordinatorFor* and each relation node *curriculum* is mapped to a node with the same label. Note that both entities of type *GradDegree* are mapped to a single entity, which is identified as "the MSc AI". The second homomorphism maps concept nodes in this way: $a \mapsto x$, $b \mapsto z$, $c \mapsto z$, $d \mapsto t$, $e \mapsto t$. This homomorphism uses the fact that *Logics* is a specialisation of both *Maths* and *CS*, which allows one to map b and c to z.

BG-homomorphism induces a preorder on BGs, called the "specialisation / generalisation" relation: in the following, we note $H \leq G$ (H is a specialisation of G) if there is a homomorphism from G to H. This relation is sound and complete with respect to logical entailment on the formulas assigned to the BGs (also using the formulas assigned to the support), i.e., for all BGs G and H on a support S, $H \leq G$ if and only if $\Phi(S), \Phi(H) \models \Phi(G)$. Completeness is up to a normality condition for H: an individual marker has to occur at most once in H (in other words, two distinct nodes cannot refer to the same identified entity). The following problem, called **BG-Homomorphism** is thus the fundamental problem on BGs: given two BGs G and H, is there a homomorphism from G to H? This problem is NP-complete in general, but belongs to PTime when the source graph (i.e., G) is an acyclic graph (or an acyclic hypergraph, this latter notion being more general than the former), and more generally if it has a bounded treewidth (or hypertreewidth).

Homomorphism being a fundamental notion in the study of relational structures, it is not surprising that BG-Homomorphism is strongly equivalent to other fundamental problems in AI and databases, which allows one to import algorithmic techniques from one domain to another. The logical translation of a BG is the same as a (Boolean) conjunctive query (CQ) in databases. The problems of evaluating a conjunctive query (e.g., given a CQ q and a relational database instance D, does D contain an answer to q?) or determining if a conjunctive query is contained in another (given two CQs q_1 and q_2, is the set of answers to q_1 included in the set of answers to q_2 for any database instance?) are essentially the same as BG-Homomorphism. The same remark holds for the basic constraint satisfaction problem (CSP): given a constraint network (in which constraints are given in extension), does this network have a solution? (see Chapter 6 of Volume 2).

Finally, let us consider the ontology-mediated query answering problem in the conceptual graph framework, where the knowledge base is composed of a support (the ontology) and of BGs (the facts), and the query is itself a BG. Checking if

the query is entailed by the KB is NP-complete in combined complexity (since it amounts to a BG-homomorphism test) and polynomial in data complexity.[8]

3.2 Simple extensions of the support

Two simple extensions of the support are often considered, namely relation signatures and concept type incompatibility. A relation signature specifies the maximal type of each of its arguments. Formally, one adds to the support a mapping σ that assigns to each relation r with arity k a signature $\sigma(r) \in (T_C)^k$. Let $\sigma_i(r)$ denote the i-th element of $\sigma(r)$; the formula assigned to $\sigma(r)$ is:

$$\forall x_1 \ldots x_k (r(x_1 \ldots x_k) \rightarrow \sigma_1(r)(x_1) \wedge \ldots \wedge \sigma_k(r)(x_k))$$

Figure 7 shows a partially ordered set of relations with their signature that corresponds to the set of DL inclusions from Figure 4, except for the part in italics.

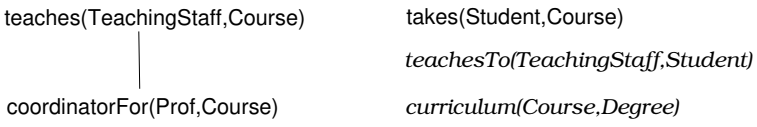

Fig. 7 Relations and their signatures

Relation signatures must be covariant with respect to the partial orders on concept types and relations: for all relations r_1 et r_2 with arity k, if $r_1 \leq r_2$ then $\sigma(r_1) \leq \sigma(r_2)$, i.e., for all i, $\sigma_i(r_1) \leq \sigma_i(r_2)$. This covariance condition translates the fact that when a relation is specialized into another, the maximal type of each argument can be specialized as well, but not generalized. For instance, if the relation *teaches* links an entity of type *TeachingStaff* to an entity of type *Course*, its specialisation into the relation *coordinatorFor* may enforce that the first argument is of type *Prof*, which is a specialisation of *TeachingStaff*.

The support added with relation signatures can be seen as a generalisation of the ontological part of RDFS (i.e., the schema) with relations of any arity (see [Baget et al., 2010] for translations between RDFS and basic conceptual graphs).

When multi-instantiation is allowed, i.e., when conjunctive concept types are considered, it is useful to express incompatibility between concept types. This can be achieved by stating that some conjunctive types are forbidden. The logical formula assigned to a *banned type* $\{t_1, t_2\}$ is the following:

[8] For query answering problems, the distinction between combined and data complexities is often made: data complexity is the complexity with respect to the size of the data (here the fact base), while combined complexity considers all components of the problem (here, the knowledge base and the query).

$$\forall x \neg (t_1(x) \wedge t_2(x))$$

The set of banned types is said to be compatible with the set of primitive types if no primitive type is a specialization of a banned type. For instance, the banned type $\{Student, Staff\}$ (which corresponds to the DL inclusion $Student \sqsubseteq \neg Staff$) is not compatible with $Student \leq Staff$, *a fortiori* with $GradStudent \leq Staff$. A BG complies with the set of banned types if no node is labelled by a concept type that specializes a banned type. Note that the logical translation $\Phi(S)$ of a support S added with banned types is always consistent. However, for a BG F on S, $\Phi(S) \cup \Phi(F)$ may be inconsistent.

3.3 Conceptual graph rules

Rules of the form " if *premise* then *conclusion*" are an essential knowledge construct in AI. They represent implicit knowledge that can be made explicit by applying them to factual knowledge. A *basic graph rule* is a pair $R = (P(c_{1_1} \dots c_{1_k}), C(c_{2_1} \dots c_{2_k}))$, where $k \geq 0$, P and C are BGs, and the c_{1_i} (respectively c_{2_i}) are distinct generic concept nodes from P (respectively C) called the *frontier nodes* of the rule. In Figure 8 the bijection between the frontier nodes of the premise and of the conclusion is depicted by dotted lines; the blue nodes form the conclusion of the rule. This rule represents the following knowledge: "if a student X takes a course Y then there is a teaching staff member Z who teaches Y and teaches to X".

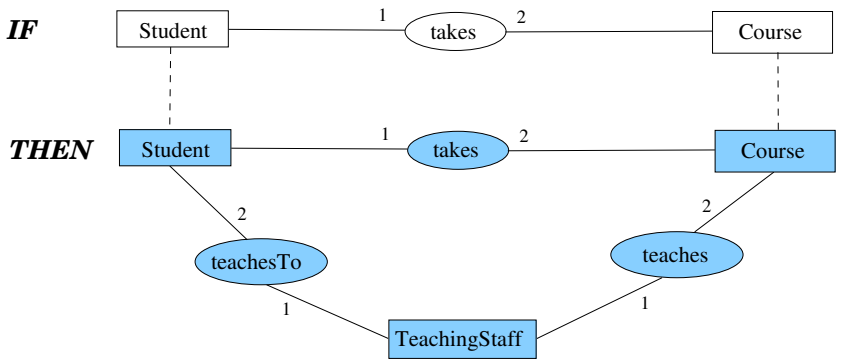

Fig. 8 Conceptual Graph Rule

The logical translation of a BG-rule $R = (P(c_{1_1} \dots c_{1_k}), C(c_{2_1} \dots c_{2_k}))$ is the formula $\Phi(R) = \forall x_1 \dots x_k \ (\Phi'(P) \rightarrow \Phi'(C))$, in which the same variable x_i is assigned to frontier nodes c_{1_i} and c_{2_i}, and $\Phi'(P)$ (resp. $\Phi'(C)$) is obtained from $\Phi(P)$ (resp. $\Phi(C)$) by leaving variables $x_1 \dots x_k$ free. Equivalently, all the variables in the premise of the rule can be universally quantified, in which case their scope is the whole formula. The logical translation of a rule is thus exactly an *ex-*

istential rule, as defined in the next section. For instance, the logical translation of the rule R from Figure 8 is $\Phi(R) = \forall x \forall y((Student(x) \wedge Course(y) \wedge takes(x,y)) \rightarrow \exists z(TeachingStaff(z) \wedge teaches(z,y) \wedge teachesTo(z,x)))$.

BG-rules are provided with forward and backward chaining mechanisms that proceed directly on their graphical form. A BG-rule R is applicable to a BG F if there is a homomorphism h from its premise to F; applying R to F according to h consists of adding C to F, then merging each frontier node c_{2_i} from C with the node $h(c_{1_i})$ from F. [9] Rule application is the basis of a sound and complete forward chaining mechanism, i.e., given a knowledge base $\mathcal{K} = (S, F, \mathcal{R})$, where S is the support, F is the fact base (remember that a BG needs not to be connected) and $\mathcal{R}$ is the set of rules, and a BG Q (which can be seen as a query), $\Phi(\mathcal{K}) \models \Phi(Q)$ if and only if there is a sequence of applications of rules in $\mathcal{R}$ leading from F to a BG F' such that $F' \leq Q$.

The backward chaining mechanism relies on a specific unification operation (between two subgraphs, respectively of a rule conclusion and of the current BG query), which exploits the complex structure of rule conclusions induced by non-frontier concept nodes (see the existential variables in existential rules). Hence, instead of processing a goal atom by atom as backward chaining *a la* Prolog would do, entire subgraphs are unified at once. This mechanism is also sound and complete.

Note that the partial orders on concept types and relations can be encoded by BG-rules. Indeed, $t_1 \leq t_2$ is logically translated into the logical rule $\forall x_1 \ldots x_k(t_1(x_1 \ldots x_k) \rightarrow t_2(x_1 \ldots x_k))$, where k is the arity of the associated predicates. However, the fact that the partial orders are intrinsically taken into account in BG-homomorphism (which allows one to compare concept types or relations in constant time, or almost constant time, depending on the order encoding) leads to better algorithmic efficiency.

BG-rules are able to simulate the behavior of a Turing machine, hence they provide a model of computation. Therefore, the associated entailment problems are undecidable. However, many decidable cases obtained by syntactic restrictions on rules, or sets of rules, have been defined, mostly in the framework of existential rules (see Section 4.3).

3.4 Conceptual graph constraints

A BG-constraint has the same shape as a BG-rule. It can be positive or negative, depending on whether it expresses an obligation or a prohibition. A positive constraint (P, C) expresses knowledge of the form "whenever P is true, C must also be true".

[9] If c_{1_i} and c_{2_i} have the same concept type, the obtained node is labelled by the same label as $h(c_{1_i})$; if the type of c_{2_i} is strictly more specific than the type of c_{1_i}, it may happen that the labels of $h(c_{1_i})$ and c_{2_i} are incompatible (with respect to banned types), which points to an inconsistency in the knowledge base; otherwise, the label of the obtained node is the greatest lower bound of both labels: the obtained type is the conjunction of the types of $h(c_{1_i})$ and c_{2_i} and the obtained marker is the smallest of both markers.

It is satisfied by a BG if every homomorphism from P to F can be extended to a homomorphism from C to F (i.e., given h the considered homomorphism from P to F, there is a homomorphism h' from C to F such that $h'(c_{2_i}) = h(c_{1_i})$ for all frontier nodes). A negative constraint (P,C) expresses knowledge of the form "whenever P is true, C must not be true". It is satisfied by a BG if no homomorphism from P to F can be extended to a homomorphism from C to F. A negative constraint can also be represented as a BG, let C^-, obtained by merging P and C on their frontier nodes (with each c_{1_i} being merged with c_{2_i}); then C^- is satisfied by F if there is no homomorphism from C^- to F.

For instance, the constraint that a student cannot belong to the administrative staff can be expressed by the formula $\forall x(Student(x) \rightarrow \neg AdminStaff(x))$, which corresponds to the form (P,C), or by the equivalent formula $\neg\exists x(Student(x) \wedge AdminStaff(x))$, which amounts to forbid a BG. Note that the extensions to the support introduced in Section 3.2 can be encoded by constraints, namely relation signatures by positive constraints and banned types by negative constraints. Other frequent forms of constraints in ontologies are cardinality constraints: positive constraints allow one to express the condition "at least 1" (like "every professor must teach at least one undergraduate course") and negative constraints the condition "at most 0" (like "no teaching assistant can be coordinator for a course").

Negative constraints can actually be seen as particular positive constraints (with C restricted to a concept node with banned type). Positive constraints strictly generalize negative constraints, in the sense that the associated consistency problems are not in the same complexity class: the problem of determining whether a given BG satisfies a given constraint is co-NP-complete if the constraint is negative, and Π^2_P-complete otherwise.

Finally, equality is represented in conceptual graphs by so-called "co-reference links" which pairwise connect concept nodes that refer to the same entity. While co-reference links do not increase the expressivity of BG (though their use may be interesting for visualisation purposes), they do increase the expressivity of BG-rules, allowing in particular to express functional dependencies.

3.5 Relationships with description logics

Description logics and conceptual graphs are both rooted in semantic networks. They both remedy two criticisms on these common ancestors, i.e., the lack of distinction between factual and ontological knowledge, and the lack of formal semantics. Due to these common properties, their relationships have often been questioned.

Provided that relations are restricted to binary relations, a support can be seen as a simple TBox composed of atomic concept and atomic role inclusions. Relation signatures then correspond to the notions of domain and range, and banned concept types to class disjointness constraints. On the other hand, an ABox can be seen as a particular BG without generic concept nodes.

With the aim of characterizing the intersection of BGs (on a simple support) and DLs, two equivalent fragments were identified in [Baader et al., 1999b]. On the CG side, we obtain rooted BG trees with binary relations. On the DL side, we obtain the DL $\mathcal{ELIRO}_1$ (a DL specially tailored for the comparison), in which the constructors are $\exists R.C$ (existential restriction), $C \sqcap D$ (concept intersection), R^-(role inverse), $R \sqcap R'$ (role intersection) and $\{i\}$ (unary one-of, where i is an individual, which allows one to integrate specific individuals in concept expressions). It is to be noticed that this comparison with conceptual graphs was one of the sources of the $\mathcal{EL}$ family, in which homomorphism is a central notion [Baader et al., 1999a].

In this intersection, both formalisms lose some natural features: on the CG side, relations of any arity and unrestricted structure, in particular cycles on generic concept nodes, while, on the DL side, the variety of constructors.

Other results support the claim that both formalisms are quite "orthogonal". On the one hand, it is known that even the most expressive DLs cannot express the whole existential positive conjunctive fragment of first-order logic [Borgida, 1996]. On the other hand, BG-homomorphism cannot handle negation in a logically complete way, even when restricted to atomic negation on primitive concept types.

More relationships between DLs and CGs can be found if we turn our attention to richer fragments of conceptual graphs including some classes of BG-rules and negative BG-constraints on the one hand, and to the ontology-mediated query answering problem on the other hand. Indeed, description logics historically focused on reasoning about the ontology (i.e., the TBox). The instance checking problem can only be seen as a very specific querying problem, which asks if a given individual belongs to a given concept. To handle conjunctive queries, new description logics were considered more recently (see Sections 2.2 and 2.3), such as the DL-Lite family, specifically designed to query data, the $\mathcal{EL}$ family, and more generally Horn description logics. These DLs can be seen as specific fragments of the existential rule framework (see the next section), which in turn can be seen as the logical translation of the conceptual graph framework described in this section.

4 Existential Rules

As already mentioned, the increasing volume of complex and heterogeneous data has spurred an intense research effort on the issue of ontology-mediated query answering in recent years. This work has deeply modified the description logic field and led to the emergence of new dialects and algorithmic techniques (Section 2.3). Meanwhile, the framework of existential rules has been developed to address this issue. The existential rule framework has a double origin: on the one hand it corresponds to the logical translation of the conceptual graph fragment (BGs, rules and negative constraints) presented in the previous section [Baget et al., 2011a], on the other hand it has been proposed as an extension to Datalog, the language of deductive databases, under the name Datalog$\pm$ [Calì et al., 2009].

In the relational database field, Datalog was originally designed to provide first-order queries (or equivalently, core SQL queries) with recursivity [Abiteboul et al., 1995]. In its plain version (i.e., without negation nor disjunction), a Datalog query can be seen as a set of rules, which are closed formulas of the form $\forall x_1 \ldots x_n \ (B \rightarrow H)$, where B and H, respectively called the body and the head of the rule (according to the logic programming terminology), are conjunctions of atoms; moreover, these rules satisfy the constraint of being "range-restricted", i.e., all the variables that occur in the head of a rule must also occur in its body. Hence, a plain Datalog rule is logically translated into a Horn clause without function symbols. These rules could be used as a means of encoding implicit background knowledge. However, they lack a property considered crucial for representing ontological knowledge, which is the ability to reason on open domains. Indeed, when the open-world assumption is made, it cannot be assumed that the only existing entities are those encoded in the data. Hence, one should be able to infer knowledge on unknown individuals, which may (or may not) be equal to entities from the data. These considerations motivated the extension of Datalog rules with existentially quantified variables in rule heads.

4.1 The existential rule framework

Formally, an existential rule is of the form $R = \forall \mathbf{x} \forall \mathbf{y} \ (B[\mathbf{x},\mathbf{y}] \rightarrow \exists \mathbf{z} \ H[\mathbf{y},\mathbf{z}])$, where $\mathbf{x},\mathbf{y}$ and $\mathbf{z}$ are sets of variables, and B, H are conjunctions of atoms, also denoted by $body(R)$ and $head(R)$. The *frontier* of R is the set of variables shared between the body and the head of R, i.e., $\mathbf{y}$. The *existential variables* in R are the existentially quantified variables, i.e., $\mathbf{z}$.

We now consider knowledge bases of the form $\mathcal{K} = (F, \mathcal{R})$, where F is a fact base[10] and $\mathcal{R}$ is a set of (pure) existential rules.

The logical translation of the BG-rules seen in the preceding section yields existential rules. For instance, the formula assigned to the BG-rule R from Figure 8 is $\Phi(R) = \forall x \forall y((Student(x) \wedge Course(y) \wedge takes(x,y)) \rightarrow \exists z(TeachingStaff(z) \wedge teaches(z,y) \wedge teachesTo(z,x)))$, where the frontier is $\{x,y\}$ (note that here all the variables from the body are frontier variables) and the only existential variable is z. Any conceptual graph KB of the form $\mathcal{K} = (S,F,\mathcal{R})$ can be translated into a logically equivalent existential rule KB of the form $\mathcal{K}' = (F',\mathcal{R}')$, and reciprocally. In the following, we omit quantifiers in rules as there is no ambiguity.

Beside these "pure" existential rules, two other kinds of rules are generally considered in the framework: *negative constraints*, which are existential rules with a head restricted to $\bot$, and *equality rules*, which are existential rules with a head restricted to an equality of the form $e_1 = e_2$, where the e_i are variables from the body or constants. These rules also correspond to constructs in the conceptual graph frame-

[10] A fact is usually defined as a ground atom. However, in the existential rule setting, a more general notion of a fact can be considered, where a fact is an existentially closed conjunction of atoms, which is in line with the view of a fact as a rule with an empty body. This generalized notion allows one to encode unknown values in a natural way.

work, namely negative BG-constraints and BG-rules with a conclusion restricted to two co-referent concept nodes.

Existential rules and classical description logics like $\mathcal{ALC}$ are incomparable with respect to expressivity. For instance, the $\mathcal{ALC}$ inclusions $\exists \mathsf{coordinatorFor}.\mathsf{Course} \sqsubseteq \mathsf{Prof}$ or the $\mathcal{SHIQ}$ transitivity axiom $(Trans\ P)$ can be expressed by existential rules, but not the $\mathcal{ALC}$ inclusion $\mathsf{Course} \sqsubseteq \mathsf{UndergradCourse} \sqcup \mathsf{GradCourse}$ which would require a disjunctive head, and the existential rule R from the previous example cannot be expressed in a description logic.

On the other hand, existential rules are strictly more expressive than so-called Horn description logics, which can be seen as DLs whose logical translation yield existential rules (in other words, the skolemisation of their logical translation yields Horn clauses with possibly functional symbols). The lightweight description logics $\mathcal{EL}$ and the DL-Lite dialects seen in Section 2 are examples of Horn description logics. Existential rules can be seen as overcoming two limitations of (Horn) description logics: first, predicates of any arity are allowed; second, there is no restriction on the atoms composing the body and the head of a rule, which allows one to describe complex relationships between entities (see e.g., the above rule R), whereas description logics are essentially limited to "acyclic" structures.

4.2 Relationships with database theory

An important connection with relational database theory has to be pointed out. Indeed, existential rules have the same logical form as Tuple-Generating Dependencies (TGDs), a high-level class of database constraints that generalize many constraints of practical database systems (and correspond to the CG positive constraints from the preceding section). Negative constraints are also considered in databases and equality rules (with equality between two variables) have the same logical form as the database Equality Generating Dependencies (EGDs), which generalize constraints on keys (see e.g., [Abiteboul et al., 1995]). Note that, despite their syntactic correspondence, database constructs and rules have different roles: TGDs/EGDs act as constraints to check the consistency of a database instance, whereas rules act as ontological knowledge to generate new data. However, in the database setting, it is possible to repair constraint violations with respect to TGDs/EGDs by applying them in a forward chaining manner as if they were rules. This process, known as the *chase*, is considered as one of the fundamental tools in database theory. The similarities between the studied objects explain that many theoretical results obtained in one domain are actually of interest to the other. In particular, it has long been shown that the entailment of an atom from a set of TGDs (hence a set of pure existential rules) and a database instance is an undecidable problem when no restriction is made.

An existential rule R can be applied to a fact base F if there is a homomorphism h from $body(R)$ to F, i.e., a substitution h of the variables in $body(R)$ by terms in F such that $h(body(R)) \subseteq F$ (both seen as sets of atoms). Applying R to F accord-

ing to h consists in adding $h(head(R))$ to F, where $h(head(R))$ is obtained from H by substituting each frontier variable x by $h(x)$ and safely renaming existential variables by fresh existential variables. The saturation of the fact base consists in iteratively applying rules on it until no rule application is possible. This process may of course not terminate since entailment is undecidable. Several forward chaining (or chase) variants have been defined, which differ in how they deal with the possible redundancies introduced by existential variables. It is well known that the (possibly infinite) saturation obtained by any of these chase variants forms a *universal model* of the knowledge base, i.e., a model that can be mapped by homomorphism to any other model of the KB. Hence, a universal model acts as a representative of all models of the KB, sufficient to decide conjunctive query entailment from the KB.

4.3 Decidability results

Interest in the existential rule framework gave rise to fruitful work on finding classes of existential rules for which (conjunctive) query answering is decidable. A wide range of rule classes offering various expressivity/tractability tradeoffs is now known (see e.g., [Mugnier, 2011; Gottlob et al., 2012; Thomazo, 2013; Mugnier and Thomazo, 2014] for syntheses). Most of these classes can be understood according to abstract properties that underlie decidability:

1. The set of rules $\mathcal{R}$ ensures that any KB $\mathcal{K} = (F, \mathcal{R})$ has a finite universal model. In other words, some chase variant is guaranteed to halt on any fact base. Hence, for any (Boolean) CQ q, $\mathcal{K} \models q$ if and only if $F^* \models q$, where F^* is the saturation of F. Such sets of rules are called *finite expansion sets (fes)* [Baget et al., 2011a].
2. The set of rules $\mathcal{R}$ ensures that any (Boolean) CQ q can be rewritten using the rules into a (finite) union of conjunctive queries Q such that for any KB $\mathcal{K} = (F, \mathcal{R})$ holds that $\mathcal{K} \models q$ if and only if $F \models Q$. Such sets of rules are called UCQ-rewritable or *finite unification sets (fus)* [Baget et al., 2011a]. More general forms of rewritings can be considered, such as first-order queries, which may produce a more succinct rewriting, or Datalog queries, which may provide a finite rewriting when there is no finite rewriting as a first-order query (see [Gottlob and Schwentick, 2012; Bienvenu et al., 2018] among others). It is known that UCQ-rewritability and first-order rewritability are actually equivalent properties (e.g., [Gottlob et al., 2014]).
3. The existence of a finite universal model may not be guaranted, but the set of rules $\mathcal{R}$ ensures that the saturation of any KB $\mathcal{K} = (F, \mathcal{R})$, seen as a graph, has a bounded treewidth. This allows for finite encodings of infinite saturations. Such sets of rules are called *(greedy) bounded-treewidth sets ((g)bts)* [Baget et al., 2011a,b; Thomazo, 2013].

The two first families of rules clearly enable one to come back to a classical database query answering problem: either the knowledge that can be entailed by the rules is encoded in the facts, or the relevant part of the rules is encoded in the

query. In the third case, querying the finite encoding is more involved. Deciding whether a given set of rules satisfies one of these three abstract properties is undecidable, however each abstract property admits some "concrete" cases defined by recognizable syntactic criteria. Generalisations or combinations of these properties have been defined, however their presentation is outside the scope of this chapter.

Table 1 presents the main currently known concrete rule classes for which ontology-mediated conjunctive query answering has polynomial time data complexity. We chose to present the simplest classes, in order to highlight the fundamental ideas, even if most of these classes admit generalisations that often keep the same data complexity. The existence of a finite universal model (*fes* property) is ensured by some acyclicity conditions that prevent infinite creation of new variables during the chase. Such classes include range-restricted rules (i.e., Datalog rules), *weakly-acyclic* rules, and *aGRD* rules. These two last classes are both defined by an acyclicity condition on a directed graph, which encodes variables sharing between positions in predicates in the first case, and dependencies between rules in the second case. In the first graph, called *position (dependency) graph* [Fagin et al., 2005], the nodes represent all positions in predicates occurring in rules, i.e., the node *(p,i)* represents the position i in some predicate p. Then, for each rule R and each variable x in $body(R)$ occurring in position (p,i), edges with origin (p,i) are built as follows: if x is a frontier variable, there is an edge from (p,i) to each position of x in $head(R)$; furthermore, for each existential variable y in $head(R)$ occurring in position (q,j), there is a special edge from (p,i) to (q,j). A set of rules is said to be *weakly acyclic* if its position graph has no circuit passing through a special edge. Intuitively, this condition ensures that the introduction of an existential variable in a given position can never lead to create another existential variable in the same position, hence an infinite number of existential variables.

For example, let $R_1 = h(x) \rightarrow p(x,y)$ and $R_2 = p(u,v), q(v) \rightarrow h(v)$. The position graph of $\{R_1, R_2\}$ contains a special edge from $(h,1)$ to $(p,2)$ due to R_1 and an edge from $(p,2)$ to $(h,1)$ due to R_2. Hence, $\{R_1, R_2\}$ is not weakly-acyclic.

Range-restricted rules are a special case of weakly-acyclic rules since they do not have existential variables at all.

The second graph is called *graph of rule dependencies* (GRD) [Baget et al., 2011a; Grau et al., 2013]. Intuitively, a rule R_j *depends* on a rule R_i if there is a fact base such that an application of R_i on this fact base leads to a new application of R_j. This abstract condition can be effectively computed by a specific unifier between the head of R_i and the body of R_j. The GRD of a set of rules $\mathcal{R}$ has a set of nodes in bijection with $\mathcal{R}$ and edges (R_i, R_j) whenever the rule R_j depends on the rule R_i. A set of rules is *aGRD* if its GRD has no circuit. In the above example, R_1 depends on R_2 but not the contrary (indeed, one can check that an application of R_1 can never lead to trigger an application of R_2: it produces an atom of the form $p(x,y)$, where y is a new existential variable, but it does not produce the atom $q(y)$, which on the other hand cannot exist in the fact base since y is new, hence no new application of R_2 is made possible); hence, $\{R_1, R_2\}$ is aGRD. Weak-acyclicity and aGRD are in fact incomparable properties, but they admit common generalisations [Grau et al., 2013; Rocher, 2016].

The *fus* property is ensured by conditions that allow one to bound the maximal size of a non-redundant CQ generated during the rewriting. Concrete *fus* classes include in particular *linear* rules and *sticky* rules (these two classes being incomparable). A linear rule has a body restricted to a single atom. The stickiness of a set of rules is defined by a marking procedure of the variables occurring in rules; then the set of rules is said to be sticky if no marked variable in a rule body occurs in two different atoms; intuitively, this ensures that a variable generated during the rewriting process occurs in at most one atom [Calì et al., 2010; Thomazo, 2013]. The decidability of ontology-mediated query answering for sets of rules with the *bts* property comes from an indirect argument (following a result by Courcelle), which does not directly provide a suitable algorithm. However, the expressive subclass known as *gbts* allows one to greedily build a tree decomposition of the (possibly infinite) saturated fact base, such that this tree decomposition has a bounded width. Concrete rule classes in the *gbts* family are also known as the guarded family, inspired by the guarded fragment of first-order logic. We list here the main members of this family [Calì et al., 2008; Baget et al., 2011a]. A rule is *guarded* if an atom of its body (called a guard) contains all the variables that occur in its body. Note that a linear rule is by definition guarded, hence it is not only *fus* but also *gbts*. A rule is *frontier-one* if it has only one frontier variable. A rule is *frontier-guarded* if an atom of its body guards all the variables of its frontier (hence, this class generalizes both guarded and frontier-one rules).

Rule Class	**Data Complexity**
Datalog	PTime-c [Dantsin et al., 2001]
weakly-acyclic	PTime-c [Dantsin et al., 2001](LB) [Fagin et al., 2005](UB)
aGRD	AC_0(1)
linear	AC_0 [Calì et al., 2009] (1)
sticky	AC_0 [Calì et al., 2010] (1)
guarded	PTime-c [Calì et al., 2009]
frontier-guarded	PTime-c [Baget et al., 2011b]
frontier-1	PTime-c [Baget et al., 2011b]

(1) The AC_0 membership for data complexity follows from the FO-rewritability

Table 1 Fundamental classes of existential rules with polynomial-time data complexity

Most Horn description logics belong to the *gbts* family, except those including transitivity, and more generally composition, of binary relations. Indeed, transitivity destroys the tree-like structure of the saturation. For instance, $\mathcal{EL}$ and $\mathcal{ELHI}$ are frontier-guarded, while DL-Lite$_\mathcal{R}$ is linear (hence, also *fus*).

5 Conclusion

Reasoning with ontologies is becoming central in many data-centric applications for which ontologies are a way to integrate heterogeneous data by providing a common conceptual vocabulary. In this setting, it is crucial to deeply understand the impact of the ontological constructs on the complexity of the main reasoning problems. This chapter provides the required formal background and results to help data practitioners to choose the knowledge representation formalism with the best expressivity / complexity tradeoff regarding their application needs.

Ontology-mediated query answering is a vibrant area at the crossroads of several domains, namely data management, knowledge representation and reasoning, and the Semantic Web. Undoubtedly, many issues remain to be solved before the widespread adoption of the framework in practice. We will mention some of the challenges currently addressed in the area, without any claim to be exhaustive. Up to recently, most work were limited to conjunctive queries, or slight extensions of them, while the ability to process more complex queries is required. The combination of conjunctive queries and navigational queries has begun to be investigated (e.g., [Stefanoni et al., 2014; Bienvenu et al., 2015; Baget et al., 2017]). New algorithmic techniques are being developed to meet the challenge of scalability beyond simple ontological languages (e.g., approaches that combine materialization of inferences and query rewriting [Lutz et al., 2013; Feier et al., 2015]). The integration of heterogeneous data under the form of a (possibly virtual) fact base relies on so-called mappings from these data to facts over the ontological vocabulary [Poggi et al., 2008]: while mappings are a classical notion in data integration, their introduction poses new challenges in the presence of an ontology (e.g., [Bienvenu and Rosati, 2016; Botoeva et al., 2016]). Representing and reasoning with temporal and spatial data, as well as information about their reliability and provenance, are of uttermost importance in most data-centric applications and have only recently begun to be explored in the context of ontology-mediated query answering [Artale et al., 2015; Borgwardt et al., 2015; Bereta and Koubarakis, 2016; Brandt et al., 2017]. Last but not least, practically robust query answering has to be tolerant to data inconsistencies, which are likely to occur in large datasets especially when the data issues from multiple data sources (e.g., [Lembo et al., 2015; Lukasiewicz et al., 2015; Bienvenu and Bourgaux, 2016]).

References

Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley.

Artale, A., Calvanese, D., Kontchakov, R., and Zakharyaschev, M. (2009). The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR)*, 36:1–69.

Artale, A., Kontchakov, R., Kovtunova, A., Ryzhikov, V., Wolter, F., and Zakharyaschev, M. (2015). First-order rewritability of temporal ontology-mediated queries. In Yang, Q. and Wooldridge, M., editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2706–2712. AAAI Press.

Baader, F., Brandt, S., and Lutz, C. (2005). Pushing the EL envelope. In Kaelbling, L. P. and Saffiotti, A., editors, *Proceedings of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 364–369.

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

Baader, F., Horrocks, I., Lutz, C., and Sattler, U. (2017). *An Introduction to Description Logic*. Cambridge University Press.

Baader, F., Küsters, R., and Molitor, R. (1999a). Computing least common subsumers in description logics with existential restrictions. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 96–103.

Baader, F., Molitor, R., and Tobies, S. (1999b). Tractable and Decidable Fragments of Conceptual Graphs. In *International Conference on Conceptual Structures*, volume 1640 of *LNAI*, pages 480–493. Springer.

Baget, J., Bienvenu, M., Mugnier, M., and Thomazo, M. (2017). Answering conjunctive regular path queries over guarded existential rules. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 793–799.

Baget, J.-F., Croitoru, M., Gutierrez, A., Leclère, M., and Mugnier, M.-L. (2010). Translations between rdf(s) and conceptual graphs. In *International Conference on Conceptual Structures (ICCS'10)*, pages 28–41.

Baget, J.-F., Leclère, M., Mugnier, M.-L., and Salvat, E. (2011a). On rules with existential variables: Walking the decidability line. *Artificial Intelligence*, 175(9-10):1620–1654.

Baget, J.-F., Mugnier, M.-L., Rudolph, S., and Thomazo, M. (2011b). Walking the complexity lines for generalized guarded existential rules. In *IJCAI'11*, pages 712–717.

Bereta, K. and Koubarakis, M. (2016). Ontop of geospatial databases. In Groth, P. T., Simperl, E., Gray, A. J. G., Sabou, M., Krötzsch, M., Lécué, F., Flöck, F., and Gil, Y., editors, *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, volume 9981 of *Lecture Notes in Computer Science*, pages 37–52.

Berners-Lee, T., Hendler, J., and O.Lassila (2001). The semantic web. *Scientific American, 279*.

Bienvenu, M. and Bourgaux, C. (2016). Inconsistency-tolerant querying of description logic knowledge bases. In *Reasoning Web: Logical Foundation of Knowledge Graph Construction and Query Answering - 12th International Summer School 2016, Aberdeen, UK, September 5-9, 2016, Tutorial Lectures*, pages 156–202.

Bienvenu, M., Kikot, S., Kontchakov, R., Podolskii, V. V., and Zakharyaschev, M. (2018). Ontology-mediated queries: Combined complexity and succinctness of rewritings via circuit complexity. *Journal of the ACM*. Forthcoming.

Bienvenu, M. and Ortiz, M. (2015). Ontology-mediated query answering with data-tractable description logics. In *Lecture Notes of the 11th International Reasoning Web Summer School*, volume 9203 of *LNCS*, pages 218–307. Springer.

Bienvenu, M., Ortiz, M., and Simkus, M. (2015). Regular path queries in lightweight description logics: Complexity and algorithms. *J. Artif. Intell. Res.*, 53:315–374.

Bienvenu, M. and Rosati, R. (2016). Query-based comparison of mappings in ontology-based data access. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016.*, pages 197–206.

Blackburn, P., Benthem, J. V., and Wolter, F. (2006). *Handbook of Modal Logic*. Springer.

Borgida, A. (1996). On the relative expressiveness of description logics and predicate logics. *Artificial intelligence*, 82:353–367.

Borgwardt, S., Lippmann, M., and Thost, V. (2015). Temporalizing rewritable query languages over knowledge bases. *J. Web Sem.*, 33:50–70.

Botoeva, E., Calvanese, D., Santarelli, V., Savo, D. F., Solimando, A., and Xiao, G. (2016). Beyond OWL 2 QL in OBDA: rewritings and approximations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 921–928.

Brandt, S., Kalayci, E. G., Kontchakov, R., Ryzhikov, V., Xiao, G., and Zakharyaschev, M. (2017). Ontology-based data access with a horn fragment of metric temporal logic. In Singh, S. P. and Markovitch, S., editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 1070–1076. AAAI Press.

Calì, A., Gottlob, G., and Kifer, M. (2008). Taming the infinite chase: Query answering under expressive relational constraints. In *International Conference on Principles of Knowledge Representation and Reasoning(KR)*, pages 70–80.

Calì, A., Gottlob, G., and Lukasiewicz, T. (2009). A general datalog-based framework for tractable query answering over ontologies. In *International Conference on Principles of Database Systems (PODS)*, pages 77–86.

Calì, A., Gottlob, G., and Pieris, A. (2010). Advanced processing for ontological queries. *PVLDB*, 3(1):554–565.

Calvanese, D., Giacomo, G. D., Lembo, D., Lenzerini, M., and Rosati, R. (2007). Tractable reasoning and efficient query answering in description logics: The *dl-lite* family. *Journal of Automated Reasoning (JAR)*, 39(3):385–429.

Chein, M. and Mugnier, M.-L. (2009). *Graph-based Knowledge Representation and Reasoning—Computational Foundations of Conceptual Graphs*. Advanced Information and Knowledge Processing. Springer.

Chen, P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1):9–36.

Dantsin, E., Eiter, T., Gottlob, G., and Voronkov, A. (2001). Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425.

Dau, F. (2003). *The Logic System of Concept Graphs with Negation And Its Relationship to Predicate Logic*, volume 2892 of *Lecture Notes in Computer Science*. Springer.

Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. (2005). Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124.

Feier, C., Carral, D., Stefanoni, G., Grau, B. C., and Horrocks, I. (2015). The combined approach to query answering beyond the OWL 2 profiles. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2971–2977.

Gottlob, G., Kikot, S., Kontchakov, R., Podolskii, V. V., Schwentick, T., and Zakharyaschev, M. (2014). The price of query rewriting in ontology-based data access. *Artif. Intell.*, 213:42–59.

Gottlob, G., Orsi, G., Pieris, A., and Simkus, M. (2012). Datalog and its extensions for semantic web databases. In *Reasoning Web*, pages 54–77.

Gottlob, G. and Schwentick, T. (2012). Rewriting ontological queries into small nonrecursive datalog programs. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*.

Grau, B. C., Horrocks, I., Kazakov, Y., and Sattler, U. (2008). Modular reuse of ontologies: Theory and practice. *J. Artif. Intell. Res.*, 31:273–318.

Grau, B. C., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., and Wang, Z. (2013). Acyclicity notions for existential rules and their application to query answering in ontologies. *J. Artif. Intell. Res. (JAIR)*, 47:741–808.

Gruber, R. (1993). A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220.

Guarino, N. (1998). Formal ontology and information systems. In Guarino, N., editor, *Formal Ontology and Information Systems*, pages 3–15. IOS Press.

Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible SROIQ. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 57–67.

Horrocks, I., Sattler, U., and Tobies, S. (1999). Practical reasoning for expressive description logics. In *Logic Programming and Automated Reasoning, 6th International Conference, LPAR'99, Tbilisi, Georgia, September 6-10, 1999, Proceedings*, pages 161–180.

Konev, B., Lutz, C., Walther, D., and Wolter, F. (2013). Model-theoretic inseparability and modularity of description logic ontologies. *Artif. Intell.*, 203:66–103.

Kontchakov, R., Wolter, F., and Zakharyaschev, M. (2010). Logic-based ontology comparison and module extraction, with an application to dl-lite. *Artif. Intell.*, 174(15):1093–1141.

Lehmann, F. (1992). *Semantic Networks in Artificial Intelligence*. Elsevier Science Inc., New York, NY, USA.

Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., and Savo, D. F. (2015). Inconsistency-tolerant query answering in ontology-based data access. *J. Web Sem.*, 33:3–29.

Levy, A. and Rousset, M.-C. (1998). Combining horn rules and description logics in carin. *Artificial Intelligence*, 101.

Lukasiewicz, T., Martinez, M. V., Pieris, A., and Simari, G. I. (2015). From classical to consistent query answering under existential rules. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 1546–1552.

Lutz, C., Seylan, I., Toman, D., and Wolter, F. (2013). The combined approach to OBDA: taming role hierarchies using filters. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, pages 314–330.

Mugnier, M. and Thomazo, M. (2014). An introduction to ontology-based query answering with existential rules. In *Reasoning Web. Reasoning on the Web in the Big Data Era - 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings*, pages 245–278.

Mugnier, M.-L. (2011). Ontological query answering with existential rules. In *Rules and Reasoning the Web (RR'11)*, pages 2–23.

Nebel, B. (1990). Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2):235–249.

Ortiz, M. and Simkus, M. (2012). Reasoning and query answering in description logics. In *Lecture Notes of the 8th International Reasoning Web Summer School*, pages 1–53.

Peñaloza, R. and Sertkaya, B. (2017). Understanding the complexity of axiom pinpointing in lightweight description logics. *Artif. Intell.*, 250:80–104.

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., and Rosati, R. (2008). Linking data to ontologies. *J. Data Semantics*, 10:133–173.

Rocher, S. (2016). *Querying Existential Rule Knowledge Bases: Decidability and Complexity*. PhD thesis, Université de Montpellier.

Schild, K. (1991). A correspondence theory for terminological logics: Preliminary report. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Schlobach, S. and Cornet, R. (2003). Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 355–362.

Sebastiani, R. and Vescovi, M. (2009). Axiom pinpointing in lightweight description logics via horn-sat encoding and conflict analysis. In Schmidt, R. A., editor, *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*, volume 5663 of *Lecture Notes in Computer Science*, pages 84–99. Springer.

Sowa, J. F. (1976). Conceptual Graphs. *IBM Journal of Research and Development*.

Sowa, J. F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley.

Stefanoni, G., Motik, B., Krötzsch, M., and Rudolph, S. (2014). The complexity of answering conjunctive and navigational queries over OWL 2 EL knowledge bases. *J. Artif. Intell. Res.*, 51:645–705.

Thomazo, M. (2013). *Conjunctive Query Answering Under Existential Rules - Decidability, Complexity, and Algorithms*. PhD thesis, Montpellier 2 University, France.

W3C (2004a). OWL web Ontology Language. http://www.w3.org/2004/OWL/.

W3C (2004b). RDF vocabulary description language 1.0: RDF schema. http://www.w3.org/TR/rdf-schema/.

W3C (2012a). OWL 2 web Ontology Language. https://www.w3.org/TR/owl-syntax/.

W3C (2012b). OWL 2 Web Ontology Language Profiles. https://www.w3.org/TR/owl2-profiles/.

Contents