Existential Rules: A Graph-Based View
(Extended Abstract)

Marie-Laure Mugnier

University of Montpellier, France

1 Introduction

We consider rules that allow to assert the existence of new individuals, an ability called *value invention* in databases [AHV95]. These rules are of the form *body* $\rightarrow$ *head*, where the body and the head are function-free conjunctions of atoms, and variables that occur only in the head are *existentially* quantified, hence their name $\forall\exists$-rules in [BLMS09, BLM10] or existential rules in [BMRT11, KR11]. Existential rules have long been studied in databases as high-level constraints called *tuple generating dependencies* (TGDs) [BV84]. Recently, there has been renewed interest for these rules in the context of ontology-based data access (OBDA), a new paradigm that seeks to exploit the semantics encoded in ontologies while querying data. The deductive database language Datalog could be seen as a natural candidate for expressing ontological knowledge in this context, however its limitation is that it does not allow for value invention, since all variables in a rule head necessarily occur in the rule body. Value invention has been recognized as a necessary prerequisite in an open-world perspective, where all individuals are not known *a priori*. It is in particular a feature of description logics (DLs), well-known languages dedicated to ontological representation and reasoning. This prerequisite motivated the recent extension of Datalog to existential rules, which gave rise to the *Datalog +/-* formalism [CGK08, CGL09].

Existential rules have indeed some particularly interesting features in the context of OBDA. On the one hand, they generalize lightweight DLs dedicated to query answering (DL-Lite [CGL$^+$07] and $\mathcal{EL}$ [BBL05] families, and more generally Horn DLs) while being more powerful and flexible [CGL09, BLM10, BMRT11]. In particular, they have unrestricted predicate arity (while DLs consider unary and binary predicates only). This allows for a natural coupling with database schemas, in which relations may have any arity; moreover, adding pieces of information, for instance to take contextual knowledge into account, is made easier, since these pieces can be added as new predicate arguments. On the other hand, existential rules cover plain Datalog, while allowing for incompleteness in the data.

Historically, we studied existential rules as part of another research line that seeks to develop a knowledge representation and reasoning formalism based on (hyper)graphs. This formalism is *graph-based* not only in the sense that all objects are defined as graphs, while being equipped with a logical semantics, but also in the sense that reasoning relies on graph mechanisms, which are sound and complete with respect to the logical semantics. This framework, presented thoroughly in [CM09], is rooted in conceptual graphs [Sow84]. The logical translation of the graph rules yield exactly

P. Barceló and R. Pichler (Eds.): Datalog 2.0, LNCS 7494, pp. 21–26, 2012.

existential rules (and other Datalog+/- constructs, like constraints, have their equivalent in this framework).

In this talk, we present a graph view of the existential rule framework and some related results. Generally speaking, seeing formulas as graphs or hypergraphs allows to focus on their structure: paths, cycles or decompositions are then fundamental notions. Two examples of results exploiting the graph structure will be detailed: the decidable class of *(greedy) bounded-treewidth sets* of rules, which is based on the tree decomposition of a graph, and a backward chaining mechanism based on subgraphs called *pieces*.

2 The Logical Framework

An *existential rule* is a first-order formula $R = \forall \mathbf{x} \forall \mathbf{y}(B[\mathbf{x}, \mathbf{y}] \rightarrow (\exists \mathbf{z} H[\mathbf{y}, \mathbf{z}]))$ where B and H are conjunctions of atoms (without function symbol except constants). A *fact* is the existential closure of a conjunction of atoms. Note that we extend the classical notion of a fact as a ground atom in order to take existential variables produced by rules into account. Moreover, this allows to cover naturally languages such as RDF/S, in which a blank node is logically translated into an existentially quantified variable, or basic conceptual graphs. In this talk, a knowledge base is composed of a set of facts, seen as a single fact, and of existential rules (other components could be added, see e.g. [CGL09] [BMRT11]). Query answering consists of computing the set of answers to a query in the knowledge base. We consider conjunctive queries (CQs), which are the standard basic queries. Boolean CQs have the same form as facts. The fundamental decision problem associated with query answering can be expressed in several equivalent ways, in particular as a Boolean CQ entailment problem: is a Boolean CQ logically entailed by a knowledge base ? In the following this problem is refered as the “entailment” problem.

A fundamental tool for query answering is homomorphism: given two facts/Boolean queries F and Q seen as sets of atoms, a *homomorphism* h from Q to F is a substitution of the variables in Q by terms in F such that $h(Q) \subseteq F$. It is well-known that F logically entails Q iff there is a homomorphism from Q to F. Sound and complete mechanisms for entailment with rules are obtained with classical paradigms, namely *forward chaining* (also called bottom-up approach, and chase when applied to TGDs) and *backward chaining* (also called top-down approach). Forward chaining enriches the initial fact by applying rules —with rule application being based on homomorphism— and checks if a fact can be derived to which the query maps by homomorphism. Backward chaining uses the rules to rewrite the query in different ways —with rewriting being based on unification— with the aim of producing a query that maps to the initial fact by homomorphism. Note that, due to the existential variables in the rule heads, unification cannot operate atom by atom as it is classically done for Horn clauses, a more complex operation is required.

3 The Graph-Based Framework

A set of atoms $\mathcal{A}$ can be seen as an ordered labeled hypergraph $\mathcal{H}_{\mathcal{A}}$, whose nodes and ordered hyperedges respectively encode the terms and the atoms from $\mathcal{A}$. One may also encode $\mathcal{A}$ as an undirected bipartite graph which is exactly the *incidence graph* of $\mathcal{H}_\mathcal{A}$ (it is a multigraph actually since there may be several edges between two nodes), where one class of nodes encodes the terms and the other the atoms (see Figure 1): for each atom $p(t_1, \ldots, t_k)$ in $\mathcal{A}$, instead of a hyperedge, there is an atom node labeled by p and this node is incident to k edges linking it to the nodes assigned to $t_1, \ldots, t_k$. Each edge is labeled by the position of the corresponding term in the atom. Therefore, all objects of the preceding logical framework can be defined as graphical objects. In particular, a fact or a query is encoded as a (hyper)graph and an existential rule can be seen as a pair of (hyper)graphs or equivalently as a bicolored (hyper)graph. Entailment between facts/queries is computed by a (hyper)graph homomorphism, which corresponds to the homomorphism notion defined on formulas; entailment using rules relies on homomorphism in the same way as in the logical framework.

Example 1. Figure 1 pictures a fact $F = siblingOf(a, b)$, a rule $R = siblingOf(X, Y) \rightarrow parentOf(Z, X) \wedge parentOf(Z, Y)$ (quantifiers are omitted) with its body in white and its head in gray, as well as the fact $F' = sibblingOf(a, b) \wedge parentOf(Z_0, a) \wedge parentOf(Z_0, b)$ obtained by applying R to F, where Z_0 is the newly created existential variable. Note that it is not necessary to label nodes representing variables, the graph structure being sufficient to encode co-occurrences of variables.

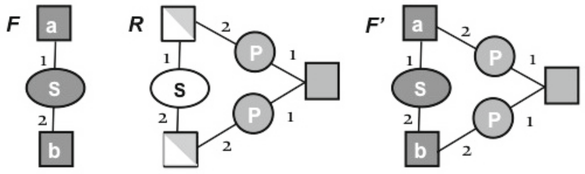

Fig. 1. Graph Representation of facts and rules

As mentioned in the introduction, this graph-based framework can be seen as a specific member of the conceptual graph fragments we have defined and developed. Conceptual graphs are defined with respect to a vocabulary, which can be seen as a very basic ontology. This vocabulary contains two finite (pre)ordered sets of concepts and of relations with any arity —and it can be further enriched by relation signatures, concept disjointness assertions, etc. The orders are interpreted as a specialization relation. Concepts and relations are logically translated into predicates and the specialization orders into formulas of the form $\forall x_1 \ldots x_k p_2(x_1 \ldots x_k) \rightarrow p_1(x_1 \ldots x_k)$ for $p_2 \leq p_1$. A basic conceptual graph is a bipartite multigraph where so-called concept nodes represent instances of concepts (i.e., terms) and relation nodes represent relations between concept instances (i.e., atoms). A concept node is labeled by a set of concepts (interpreted as a conjunction) and a marker (which can be the generic marker $\star$, referring to an unknown individual, or a constant). A relation node is labeled by a relation. Concept labels are partially ordered in a lattice obtained from the order on concepts and the order on markers ($\star$ is greater than all constants, which are pairwise incomparable). Homomorphism takes the orders on labels into account: for all concept or relation node x, one must have $label(x) \geq label(h(x))$. This allows to take the ontology into account in a very efficient way as the label comparisons can be compiled then performed in constant time. Note that the semantic web language RDFS can be encoded in the basic conceptual graph fragment. Conceptual graph rules are defined as pairs of basic conceptual graphs.

The existential rule framework can thus be seen as a conceptual graph fragment in which the vocabulary is restricted to a singleton concept set and a flat relation set. Both have the same expressivity, since the orders on concepts and relations can be encoded into the graphs —as if the rules translating these orders were applied in forward chaining.

4 Procedures for Entailement with Existential Rules

The ability to generate existential variables, associated with arbitrarily complex conjunctions of atoms, makes entailment undecidable in general. Since the birth of TGDs various conditions of decidability have been exhibited. We focus here on two abstract properties, which come with finite procedures based on forward and backward chaining respectively, and for which the graph view is particularly relevant. These properties are said to be *abstract* in the sense that they are not recognizable, i.e., deciding if a given set of rules has the property is undecidable [BLM10]. However, they provide generic algorithmic schemes that can be further customized for specific recognizable classes of rules.

A set of rules $\mathcal{R}$ is said to have the *bounded treewidth set (bts)* property if for any initial fact F, there is an integer b such that the treewidth of any fact derived from F with $\mathcal{R}$ is bounded by b (property essentially introduced in [CGK08]). The treewidth is defined with respect to a graph (the “primal graph”) associated with the hypergraph encoding a fact. The decidability proof of entailment with *bts* rules does not provide a halting algorithm (at least not directly). A subclass of *bts* has been defined recently, namely *greedy bts (gbts)*, which is equipped with a forward-chaining-like halting algorithm [BMRT11, TBMR12]. For this class of rules a bounded width tree decomposition of any derived fact can be built in a greedy way. The set of all possibly derived facts can be encoded in such tree, however this tree may be infinite. An appropriate equivalence relation on the nodes of this tree allows to build only a finite part of it. The *gbts* class is very expressive, as it includes plain Datalog, (weakly) guarded rules [CGK08], *frontier-one (fr1)* rules [BLMS09], and their generalizations (weakly / jointly) frontier-guarded rules [BLM10, KR11]. The algorithm provided in [TBMR12] can be customized to run in the “good” complexity class for these subclasses, which is important since some of them have polynomial data complexity.

A set of rules is said to have the *finite unification set (fus)* property if the set of rewritten queries restricted to its most general elements is finite. This class includes for instance rules with atomic body (also called linear Datalog+/-), domain-restricted rules and sticky(-join) rules [BLMS09, CGL09, CGP10a, CGP10b]. We propose a (sound and complete) backward chaining mechanism which computes such minimal set of rewritings when the rules are *fus*. Its originality lies in the rewriting step which is based on a graph notion, that of a *piece*. Briefly, a piece is a subgraph (i.e., subset of atoms) of the query that must be erased as a whole during a rewriting step (see [SM96] for the first piece-based backward chaining on conceptual graph rules, [BLMS11] for a revised version dedicated to existential rules and [KLMT12] for an effective implementation). We point out that the unification operation can take a preorder on predicates into account, similarly to conceptual graph operations, which can save an exponential number of rewritings.

5 Conclusion

The existential framework in the context of OBDA is rather young and challenging issues need to be solved before systems effectively able to deal with large amounts data can be built. We believe that having a double view —logical and graphical— of this framework is likely be fruitful. The logical view allows to connect to Datalog, description logics and other logic-based knowledge representation and reasoning languages. The graph view allows to connect to studies in graph theory, or in other areas in which (hyper)graphs structures have been particularly well-studied from an algorithmic viewpoint, such as constraint programming. It also allows to relate directly to graph representations of data, such as RDF/S triplestores, and other emerging graph-based paradigms for data management.

References

[AHV95] Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)

[BBL05] Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: IJCAI 2005, pp. 364–369 (2005)

[BLM10] Baget, J.-F., Leclère, M., Mugnier, M.-L.: Walking the decidability line for rules with existential variables. In: KR 2010, pp. 466–476. AAAI Press (2010)

[BLMS09] Baget, J.-F., Leclère, M., Mugnier, M.-L., Salvat, E.: Extending decidable cases for rules with existential variables. In: IJCAI 2009, pp. 677–682 (2009)

[BLMS11] Baget, J.-F., Leclère, M., Mugnier, M.-L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artificial Intelligence 175(9-10), 1620–1654 (2011)

[BMRT11] Baget, J.-F., Mugnier, M.-L., Rudolph, S., Thomazo, M.: Walking the complexity lines for generalized guarded existential rules. In: IJCAI 2011, pp. 712–717 (2011)

[BV84] Beeri, C., Vardi, M.Y.: A proof procedure for data dependencies. Journal of the ACM 31(4), 718–741 (1984)

[CGK08] Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: KR 2008, pp. 70–80 (2008)

[CGL$^+$07] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. Autom. Reasoning 39(3), 385–429 (2007)

[CGL09] Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: PODS 2009, pp. 77–86 (2009)

[CGP10a] Calì, A., Gottlob, G., Pieris, A.: Advanced processing for ontological queries. PVLDB 3(1), 554–565 (2010)

[CGP10b] Calì, A., Gottlob, G., Pieris, A.: Query answering under non-guarded rules in datalog+/-. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 1–17. Springer, Heidelberg (2010)

[CM09] Chein, M., Mugnier, M.-L.: Graph-based Knowledge Representation and Reasoning—Computational Foundations of Conceptual Graphs. Advanced Information and Knowledge Processing. Springer (2009)

[KLMT12] König, M., Leclère, M., Mugnier, M.-L., Thomazo, M.: A sound and complete backward chaining algorithm for existential rules. In: RR 2012 (to appear, 2012)

[KR11] Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: IJCAI 2011, pp. 963–968 (2011)

[SM96] Salvat, E., Mugnier, M.-L.: Sound and Complete Forward and Backward Chainings of Graph Rules. In: Eklund, P., Mann, G.A., Ellis, G. (eds.) ICCS 1996. LNCS (LNAI), vol. 1115, pp. 248–262. Springer, Heidelberg (1996)

[Sow84] Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley (1984)

[TBMR12] Thomazo, M., Baget, J.-F., Mugnier, M.-L., Rudolph, S.: A generic querying algorithm for greedy sets of existential rules. In: KR 2012 (2012)