Kiabora: an Analyzer of Existential Rule Bases

Michel Leclère and Marie-Laure Mugnier and Swan Rocher

University Montpellier 2, France

Abstract. Kiabora is a software tool dedicated to the analysis of a base of existential rules (also called Datalog± rules). It is able to check if a set of rules belongs to a known class for which entailment is decidable, either directly, by checking some syntactic properties, or by means of its Graph of Rule Dependencies, which allows to combine decidable cases. Kiabora is available online via a simple web form. It is written in Java and is easily extensible to take new decidability results into account. It comes with a textual format, called DLGP (for Datalog Plus), which can be seen as an extension to usual plain Datalog format. In this paper, we briefly introduce the existential rule framework as well as decidability results and presents the analysis performed by Kiabora. More details are available on Kiabora website.[1]

1 Introduction

Existential rules allow to assert the existence of not-yet-known individuals, a desirable feature in open world applications. These rules are of the form $body \rightarrow head$, where the body and the head are conjunctions of atoms (without functions) and variables that occur only in the head are *existentially* quantified. They are also known as Datalog± framework, an extension of plain Datalog to tuple-generating dependencies [CGL09]. Existential rules have been specifically developed in the context of Ontology-Based Query Answering (OBQA), which consists in querying data while taking into account inferences enabled by the ontology. Note that existential rules cover the core of lightweight description logics dedicated to query answering, while being more powerful and flexible.

The basic OBQA problem can be recast as entailment of a Boolean conjunctive query (an existentially closed conjunction of atoms) from a knowledge base composed of data and existential rules. This problem being undecidable, there has been a lot of work on defining specific cases of existential rules for which it is decidable, as well as safely combining such cases.

Kiabora is an open source software written in Java, whose purpose is to analyze a set of existential rules w.r.t. decidability of entailment. It is able to check if this set belongs to a known decidable class of rules (i.e., a class for which entailment is decidable), either directly by checking some syntactic properties or by means of its Graph of Rule Dependencies (GRD); in this latter case, the strongly connected components in the GRD are analyzed, and their properties are combined in order to determine the decidability of the whole set of rules, as well as the kind of paradigm (forward or backward chaining) ensuring decidability. Of course, decidable sets of rules are generally

[1] Kiabora website: `http://www2.lirmm.fr/graphik/kiabora/`

not recognizable, thus Kiabora tries to find sufficient properties ensuring membership of a known decidable class of rules.

Kiabora also provides some utilities, in particular concerning rule decompositions: it allows to decompose a rule into a set of equivalent rules with a simpler head, either restricted to a "piece" (which roughly corresponds to a unit of information added by a rule) or a single atom (which requires to introduce an auxilliary predicate). Kiabora's preferred input/output format is called DLGP (for Datalog Plus). DLGP is a textual format meant to be at once human-friendly, concise and easy to parse. It can be seen as an extension to the commonly used format for plain Datalog.

The paper is organized as follows. Fundamental notions about the existential rule framework and decidability properties are respectively introduced in Sections 2 and 3. Section 4 presents the analysis performed by Kiabora. Section 5 draws up some perspectives for the next tool version.

2 Preliminaries

An *atom* is of the form $p(e_1 \dots e_k)$ where p is a predicate, $k \geq 1$ is the arity of p, and each e_i is a variable or a constant. A *fact* F is an existentially closed conjunction of atoms $\exists \bar{X} F[\bar{X}]$, where $\bar{X}$ is the set of variables occurring in F. The classical notion of a fact as a ground atom is thus generalized by the introduction of existentially quantified variables. Hence a conjunction of facts can be seen as a single fact. An *existential rule* is a positive rule of the form $\forall \bar{X} \forall \bar{Y} (B[\bar{X}, \bar{Y}] \rightarrow \exists \bar{Z} H[\bar{X}, \bar{Z}])$, where B and H are conjunctions of atoms, respectively the body and the head of the rule, $\bar{X}$ are the variables shared by B and H, $\bar{Y}$ are the variables that occur only in B and $\bar{Z}$ are the variables that occur only in H; $\bar{Z}$ variables are existentially quantified.

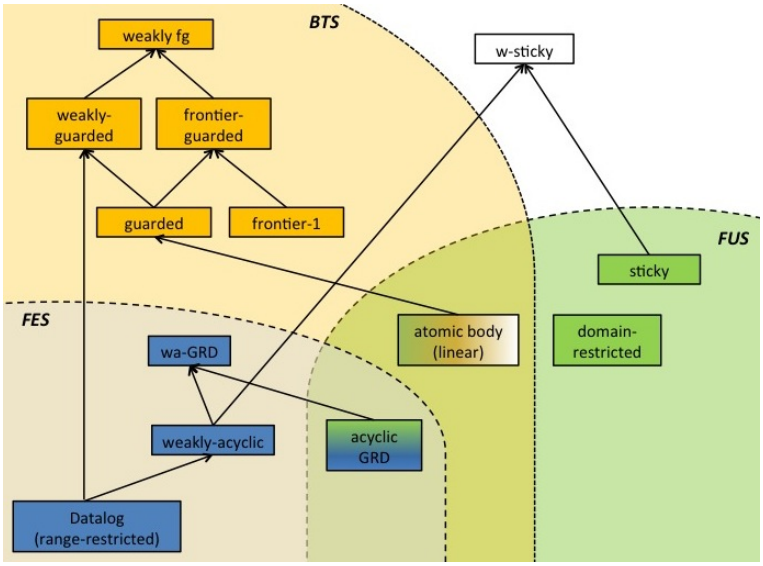
Example 1 (Facts and Rules). We consider the following predicates, with their arity mentioned after their name: area/1, project/1, researcher/1, hasExpertise/2, isMember/2, isProject/3. Unary predicates can be seen as classes of entities. Binary predicates *hasExpertise* and *isMember* respectively relate a researcher to an area and a project. The ternary predicate *isProject* links a project, the area of this project and the leader of this project. The following sentences express a fact and two rules.

- Fact: *"Researcher* a *is member of a project in* kr *area"*
  $\exists X \exists Y (researcher(a) \wedge isMember(a, X) \wedge isProject(X, kr, Y))$
- Rule 1: *"Every leader of a project is a member of this project"*
  $\forall X \forall Y \forall Z (isProject(X, Y, Z) \rightarrow isMember(Z, X))$
- Rule 2: *"Every researcher expert in an area is member of a project in this area"*
  $\forall X \forall Y (researcher(X) \wedge hasExpertise(X, Y) \rightarrow$
  $\exists Z \exists L (isProject(Z, Y, L) \wedge isMember(X, Z)))$

A *Boolean conjunctive query* Q has the same logical form as a fact. The fundamental decision problem associated with query answering can be defined as follows: given a knowledge base $\mathcal{K} = (F, \mathcal{R})$, where F is a fact (or a set of facts logically equivalent to a single fact) and $\mathcal{R}$ is a set of existential rules, and a Boolean conjunctive query Q, is Q entailed by $\mathcal{K}$ (notation $F, \mathcal{R} \models Q$)? This problem is undecidable in the general case, even if $\mathcal{R}$ is restricted to a single rule [BLMS11].

3 Decidability Properties

For reasons of brevity, in the sequel of this paper we call "decidable" a class of rules for which the fundamental entailment problem is decidable. Decidable classes found in the literature are based on various syntactic properties of existential rules. Figure 1 shows the main currently known decidable classes, partially ordered by syntactic inclusion: Datalog, weakly-acyclic [FKMP05], (weakly-) acyclic GRD [Bag04], atomic-body or linear [BLMS11,CGL09], frontier-1 and (weakly-)(frontier-)guarded [CGL09,BLMS11], domain-restricted [BLMS11], (weakly-)sticky [CGP12]. These classes are all considered by Kiabora (see Kiabora website for their definition and the synthesis paper [Mug11] for a more complete picture of decidable classes).



For readability reasons, the decidable class of "disconnected rules" was omitted in the picture. This class is included in weakly-acyclic (fes), frontier-guarded (bts) and domain-restricted (fus).

Fig. 1. Main decidable classes

There are two main paradigms for processing rules, namely forward chaining (FC) and backward chaining (BC). FC uses the rules to enrich the initial fact F into a fact F' and $F, \mathcal{R} \models Q$ holds if $F' \models Q$. BC uses the rules to rewrite Q into a query Q' and $F, \mathcal{R} \models Q$ holds if $F \models Q'$. More complex mechanisms rooted in these paradigms have then been developed. In order to classify the decidable classes found in the literature, the following abstract properties related to the behavior of FC and BC mechanisms are considered in [BLMS11]: (1) FC halts in finite time; (2) FC may not halt but the generated facts have a tree-like structure; (3) BC halts in finite time. These properties yield three *abstract* classes of rules, respectively called *finite expansion sets* (fes), *bounded treewidth sets* (bts) and *finite unification sets* (fus). The fes class is included in the bts

class, and they are both incomparable with the fus class. These classes are said to be abstract in the sense that they do not come with a syntactic property that can be checked on rules or sets of rules. As a matter of fact, they are not recognizable in general. Note that, whereas fes and fus classes are provided with halting mechanisms, no halting procedure is known for bts; however, a subclass of bts was defined, namely gbts, which comes with a halting procedure [BMRT11].

In contrast, the classes defined by a syntactic property are called *concrete*. Syntactic properties have to be fulfilled either by each rule (*unit property*) or by the set of rules as a whole (*global property*). Almost all known concrete classes fulfill one of the above abstract properties (see Figure 1). Some of them fulfill several abstract properties, moreover a given set of rules may itself belong to several concrete classes.

The rough union of two sets of rules belonging to decidable classes almost always leads to undecidability. The notion of rule dependency can be used to define constraints on the interactions between rules and safely combine decidable classes. We say that a rule R_2 *depends* on a rule R_1 if R_1 may lead to trigger R_2, i.e., there exists a fact F such that R_1 is applicable to F, which yields a fact F', and a *new* application of R_2 is possible on F' (see [BLMS11] for a formalization and computation of this notion).[2]

The Graph of Rule Dependencies (GRD) is a directed graph with the set of rules as set of nodes and an edge from a rule R_1 to a rule R_2 if R_2 depends on R_1. If the GRD is acyclic then the associated set of rules is both fes and fus. If all its strongly connected components (SCC) are fes (resp. fus) then the set of rules is fes (resp. fus). Moreover, fes, bts and fus SCC can be safely combined when the set of rules can be partitioned while enforcing specific constraints. If all SCC are fes or bts, and no rule processed as fes depends on a rule processed as bts, then the set of rules is bts; if no rule processed as bts depends on a rule processed as bts or fus then entailment is decidable. Schematically, fes/bts rules can be applied to the initial fact and used to build a finite saturated fact (fes case) or finite structure encoding a potentially infinite fact (gbts case), while fus rules can be used to rewrite the initial query into a query evaluated against the transformed fact.

4 Kiabora

Kiabora can be run via a simple web form, which allows to load a rule set file (preferably in DLGP format) and to set the analysis parameters. Following Example 2 is a DLGP file translating the fact and rules in Example 1; note that objects may be named (here the rules are named `R1` and `R2`) and quantifiers are implicit.

Example 2 (DLGP format).

```
researcher(a), isMember(a, X), isProject(X, kr, Y). % fact
[R1] ismember(Z,X) :- isProject(X, Y, Z). % plain Datalog rule
[R2] isProject(Z, Y, L), isMember(X, Z) :-
researcher(X), hasExpertise(X, Y). % extended Datalog rule
```

[2] Kiabora allows to refine this notion by enforcing that the fact obtained after the application of R_2 is not equivalent to F' [Lam12], however this option may significantly increase the computation time.

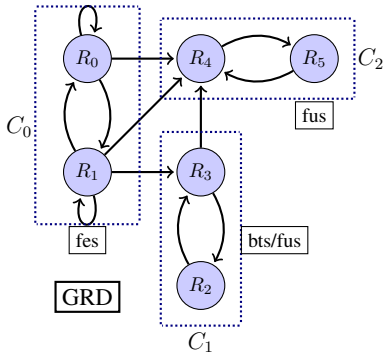
DLGP format allows to encode negative constraints (rules with an head restricted to $\bot$) and non-Boolean conjunctive queries as well, but these objects have no specific role in this paper.

The output of the analysis is composed of the following elements (in textual format):

- the GRD, the content of each SCC and the graph of the SCC (i.e., the reduced GRD, in which each SCC is a node),
- the unit properties fulfilled by each rule,
- the unit and global properties fulfilled by each SCC and by the whole set of rules,
- one or several safe combinations of abstract classes based on the SCC of the GRD,
- the conclusion of the analysis (proven or not decidable w.r.t. the selected parameters).

The following example illustrates the use of the GRD.

```
[R0] p(X,Z):-p(X,Y),p(Y,Z).
[R1] p(Y,X),q(X,Y,X):-p(X,Y).
[R2] q(X,Z,T):-r(X,Y).
[R3] r(Z,T):-q(X,Y,Z).
[R4] s(Z,Z,Z):-p(X,X),r(Y,U),s(T,Y,X).
[R5] s(Z,X,Y):-s(X,X,X),s(Y,Y,Y).
```



$\mathcal{R} = \{R_0 \dots R_5\}$ does not belong to any concrete decidable class. As shown in the above figure, the GRD of $\mathcal{R}$ contains three SCC. Each SCC satisfies some concrete properties (not mentioned in the figure), which themselves allow to prove the membership to abstract rule classes : C_0 is fes, C_1 is bts (actually gbts) and fus, and C_2 is fus. There are two safe combinations of these SCC, which differ in how C_1 is considered. If C_1 is considered as a bts, then $C_0 \cup C_1$ is identified as a bts, and $\mathcal{R}$ is found decidable since there is no edge from C_2 to $C_0 \cup C_1$. Otherwise, C_1 is considered as a fus, then $C_1 \cup C_2$ is a fus, and $\mathcal{R}$ is decidable, since there is no edge from $C_1 \cup C_2$ to C_0.

User preferences regarding abstract classes can be taken into account in the combining step. For instance, the user may prefer backward chaining, hence the fus class, in which case Kiabora will give priority to safe combinations that maximise the number of rule sets processed as fus; or she may prefer to saturate the facts, hence the fes class. Furthermore, since no algorithm is known for bts, and the algorithms available for gbts are rather complex to implement, one can also ask Kiabora to minimize the number of bts components.

All concrete properties can be checked in linear or quadratic time, except those based on the GRD. Indeed, checking if a rule R_2 depends on a rule R_1 is an NP-complete problem. Note however that it becomes linear in time if R_1 has an atomic head, which is often the case. The combining step can be performed by a simple breadth-first search in the graph of the SCC.

First experiments have been led with available sets of existential rules: for sets of about five thousands rules (e.g., Galen-Lite [KLT+10]) the whole process, including file

loading and result display, takes less that one minute on a standard labtop. Nevertheless, we have to point out that available rule sets are very simple since they are translations from lightweight description logics; experiments with more complex rule sets would be needed to test the efficiency of the GRD construction.

Finally, we have to point out a current limitation of Kiabora: for now, the equality predicate in rules, which is allowed in DLGP format, is considered like any other predicate (it follows that some rule dependencies may be missed in presence of equalities).

Since new decidable rule classes are regularly discovered, Kiabora has been designed to allow for easy inclusion of new decidability properties. Actually, adding a new class membership check is very simple as it consists of (1) implementing a Java interface and (2) inserting an instance into the graph of known decidable classes (see Figure 1). For more details, the reader is referred to the *RuleClass* interface documentation.

5 Perspectives

This paper briefly presents Kiabora, a tool for analyzing the structure of existential rule bases. Its main purpose is to determine whether the rule base belongs to a known decidable class of rules and propose paradigms to process the rules. It can also be used to compute the graph of rule depencencies independently of decidability issues. In addition it provides some utilities concerning format translation or rule decompositions. Kabiora has been designed to allow easy inclusion of new decidability results.

Further developments are planned, such as exploiting negative constraints, taking into account equalities in rules or providing a finer analysis of relevant processing algorithms.

References

Bag04. J.-F. Baget. Improving the forward chaining algorithm for conceptual graphs rules. In *KR'04*, pages 407–414. AAAI Press, 2004.

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

BMRT11. J.-F. Baget, M.-L. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In *IJCAI'11*, pages 712–717, 2011.

CGL09. A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *PODS'09*, pages 77–86, 2009.

CGP12. A. Calì, G. Gottlob, and A. Pieris. Towards more expressive ontology languages: The query answering problem. *Artif. Intell.*, 193:87–128, 2012.

FKMP05. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.

KLT+10. R. Kontchakov, C. Lutz, D. Toman, F.Wolter, and M. Zakharyaschev. The combined approach to query answering in dl-lite. In *KR*, 2010.

Lam12. B. Lamare. Optimisation de la notion de dépendance. Internship report, ENS Cachan and LIRMM/ INRIA, Sept. 2012.

Mug11. M.-L. Mugnier. Ontological Query Answering with Existential Rules. In *RR'11*, pages 2–23, 2011.