To appear on the Proceedings of the
6th International Workshop on Database Programming Languages (DBPL'97)
August 18-20, 1997, Estes Park, Colorado
Lecture Notes in Computer Science, Springer-Verlag

Datalog and Description Logics: Expressive Power

Marco Cadoli[1] and Luigi Palopoli[2] and Maurizio Lenzerini[1]

[1] Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
e-mail: <lastname>@dis.uniroma1.it
[2] Dipartimento di Elettronica Informatica e Sistemistica
Università della Calabria
I-87036 Rende (CS), Italy
e-mail: palopoli@unical.it

Abstract. Recently there was some attention on integration of description logics of the $\mathcal{AL}$-family with rule-based languages for querying relational databases such as Datalog, so as to achieve the best characteristics of both kinds of formalisms in a common framework. Formal analysis on such hybrid languages has been limited to computational complexity: i.e., how much time/space it is needed to answer to a specific query? This paper carries out a different formal analysis, the one dealing with expressiveness, which gives precise characterization of the concepts definable as queries. We first analyze the applicability to hybrid languages of formal tools developed for characterizing the expressive power of relational query languages. We then present some preliminary results on the expressiveness of hybrid languages. In particular, we show that relatively simple hybrid languages are able to define all finite structures expressed by skolemized universally quantified second-order formulae with some constraints on the quantified predicates.

1 Introduction

Recently there was some attention on integration of description logics of the $\mathcal{AL}$-family with rule-based languages for querying relational databases such as Datalog. The two classes of languages exhibit different properties.

- Description logics are good at structuring knowledge in terms of classes and relationships, but are not suitable for expressing complex queries.
- Datalog is good at formulating deductive queries, but uses a flat data model, namely, the relational model, for representing the knowledge base.

Therefore, it is reasonable to investigate knowledge representation systems, hereafter called hybrid languages, integrating the two paradigms so as to achieve the best characteristics of both kinds of formalisms in a common framework. Notable attempts of this sort are [DLNS91,LR96b,LR96a,DLNS97].

The formal analysis reported in such papers is limited to *computational complexity*: i.e., how much time/space is needed to answer to a specific query? This paper carries out a different formal analysis, the one dealing with *expressiveness* of such hybrid languages. Intuitively, the expressiveness, or expressive power, of a query language tells us what "properties" can be extracted from a knowledge base. Thus, the notion of expressive power complements that of computational complexity of a knowledge representation formalism, because the latter tells us how difficult it is to answer a query, while the former gives precise characterization of the concepts that it is possible to define as queries.

Formal studies of expressive power of description logics have been recently pursued. In particular, [Bor96] shows that description logics built using constructors usually considered in the literature are characterized by subsets of first-order logic allowing only three variable symbols. [Baa96] gives a methodological contribution, pointing out that expressiveness must be defined within a precise formal framework, and proposes the model-theoretic approach for the characterization of expressive power. Interestingly, he shows that the complexity of inference of two equally expressive languages may be different.

Since we are dealing with a framework comprising Datalog, specifically designed for querying relational databases, it seems natural to look at the formal tools and methods developed in the database field to investigate the expressiveness of query languages. The importance of formal analysis of expressive power of query languages is acknowledged in the database community [Kan90]. However, the exploitation of such formal tools in our context is not straightforward, as they have been devised to handle a case where, from the logical point of view, the knowledge base has a single model over a fixed domain — this not being the case for $\mathcal{AL}$-languages. In fact, description logics permit the specification (in the TBox) of incomplete information about the world, i.e., they permit the specification of a set of possible worlds. As we will see in the paper, this is a fundamental reason for the increased expressive power of hybrid languages.

The goal of this paper is twofold:

1. To carry out an analysis of the adequacy of the above mentioned formal tools.
2. To perform some considerations and give preliminary results on the expressiveness of hybrid languages.

With regard to the first goal, we observe that the expressive power of query languages has been measured in at least three different ways:

1. With respect to a specific property, such as transitive closure. For example, it is well-known that there is no fixed query in relational calculus that, for any graph G encoded as a relation $edge/2$ in the obvious way, determines whether the transitive closure of G contains a specific edge or not. Vice-versa, such a query does exist in Datalog.
2. With respect to a set of logical formulae, such as first- or second-order logic. For instance, "while queries" [AV92] can express exactly the set of second-order properties over ordered finite structures.

3. With respect to a complexity class, such as P, NP, coNP, PSPACE, etc. As an example, Datalog with stable negation can express all NP properties of finite structures [Sch95], e.g., whether a graph is 3-colorable or not.

According to the above classification, [Bor96] gives a contribution in the context of the second measure, whereas the so-called "Fagin's theorem" [Fag74], provides the basis for unifying the second and the third modalities. This theorem, which is one of the major results in this field, says that the set of NP properties coincides with the set of properties expressed by existentially quantified second-order formulae. This result has been generalized to other complexity classes and sets of logical formulae. In this work, we are going to use both the second and the third modality.

With regard to the second goal, let us discuss an example to help clarify the issue. Suppose we want to check the 3-colorability of a graph $G = \langle V, A \rangle$ encoded as a set of facts $\Delta_{\mathcal{E}} = \{edge(a,b) \mid (a,b) \in A\}$. In the hybrid languages we are considering, this can be done by means of a 2-components query:

TBox ($\Delta_{\mathcal{T}}$):

$$\begin{aligned} \mathsf{T} &\sqsubseteq red \sqcup green \sqcup blue \\ red &\sqsubseteq \neg green \\ green &\sqsubseteq \neg blue \\ red &\sqsubseteq \neg blue \end{aligned}$$

Datalog rules ($\Delta_{\mathcal{R}}$):

$$\begin{aligned} non\_3\_col &\leftarrow edge(X,Y), red(X), red(Y). \\ non\_3\_col &\leftarrow edge(X,Y), blue(X), blue(Y). \\ non\_3\_col &\leftarrow edge(X,Y), green(X), green(Y). \end{aligned}$$

where the axioms in the TBox (called inclusion axioms) impose that the three colors actually partition the domain, and the Datalog rules are used to define the concept of non-3-colorability (a more formal introduction to the syntax and semantics of description logics will be presented in Section 2). Indeed, one can verify that $\Delta_{\mathcal{T}} \cup \Delta_{\mathcal{R}} \cup \Delta_{\mathcal{E}} \models non\_3\_col$ iff G is not 3-colorable (where $\models$ denotes the usual logical consequence operator, i.e., validity in all models). In the terminology of [LR96b], $\Delta_{\mathcal{T}}$ is "acyclic", and $\Delta_{\mathcal{R}}$ is "non-recursive". Moreover $\Delta_{\mathcal{T}}$ belongs to the class CARIN-MARC of "maximal (decidable) $\mathcal{ALCNR}$ recursive CARIN", which includes the constructors $\sqcup, \sqcap, (\geq\ n\ R), \exists R.C$, and negation on primitive concepts. $\Delta_{\mathcal{R}}$ is "role-safe", i.e., each of its rules is such that for every atom of the form $R(x,y)$ in the antecedent, where R is a role, then either x or y appear in an ordinary atom of the antecedent. In fact, the TBox is a set of *inclusion axioms* [BDS93], and concept constructors used in the TBox are just propositional. Actually, this is an $\mathcal{AL}$-log program [DLNS91]. From the point of view of expressiveness, we can conclude that Datalog, when augmented with inclusion axioms typical of description logics, is able to capture some coNP-complete queries.

The above example just proves that the *data complexity* (i.e., complexity considering the extensional component as the input and the intensional component –the query– $\Delta_{\mathcal{T}} \cup \Delta_{\mathcal{R}}$ not part of the input) of $\mathcal{AL}$-log is coNP-hard, but it does not imply that either $\mathcal{AL}$-log or CARIN are able to express *all* queries in coNP. Such a distinction is important since the expressive power of a language is not necessarily the same as its complexity (it is always less than or equal to). Several languages with bounding complexity class properly containing their expressiveness are known, cf. e.g., [AV92,EGM97]. In the present paper we show that relatively simple hybrid languages are able to define all finite structures expressed by skolemized universally quantified second-order formulae with some constraints on the quantified predicates.

The paper is organized as follows. In Section 2 we recall the definition of hybrid knowledge bases and of expressiveness of a query language. In Section 3 we show various options for setting up a framework for expressive power in hybrid languages. In Section 4 we give results about expressiveness of two hybrid languages. In Section 5 we discuss an extension of our framework obtained by allowing stratified negation in the bodies of rules. In Section 6 we present conclusions and some open problems.

2 Preliminaries

In this section we give some preliminary notions on description logics, hybrid systems, and queries.

2.1 Description logics

From the syntactic point of view, the language of description logics contains unary relations representing classes of objects and referred to as *concepts* and binary relations, called *roles*, by which relationships between concepts can be established. Complex knowledge is defined by means of descriptions built from a set of given constructors. In this paper, we shall deal with hybrid languages whose descriptive component is a subset of $\mathcal{ALCNR}$ descriptors, which are reported next (A represents a primitive concept, and $P_1, \ldots, P_m$ represent primitive roles; C, D represent complex concepts, and R represents a complex role).

$C, D \rightarrow$	$A \mid$	(primitive concept)
	$\top \mid$	(universal concept)
	$\bot \mid$	(empty concept)
	$C \sqcap D \mid$	(conjunction)
	$C \sqcup D \mid$	(disjunction)
	$\neg C \mid$	(complement)
	$\forall R.C \mid$	(universal quantification)
	$\exists R.C \mid$	(existential quantification)
	$(\geq n\ R) \mid (\leq n\ R)$	(number restrictions)
$R \rightarrow$	$P_1 \sqcap \cdots \sqcap P_m$	(role conjunction)

As an example, the following complex concept

$$(\exists \mathsf{friend.tall}) \ \sqcap \ \forall \mathsf{friend.}(\forall \mathsf{friend.doctor}),$$

where doctor and tall are primitive concepts and friend is a primitive role, denotes the set of the individuals having at least one tall friend and such that for each of their friends, each of his or her friends is a doctor.

Description logic knowledge bases contain two parts: a terminological component $\Delta_{\mathcal{T}}$, often called TBox, and an assertional component $\Delta_{\mathcal{A}}$, also called ABox. The TBox contains sentences of the form $C \sqsubseteq D$, telling that every instance of the concept C must be also an instance of the concept D.

As an example, the following sentence states that all parents whose children are either doctors or lawyers are happy.

$$\mathsf{happy\_parent} \sqsubseteq (\exists \mathsf{child}.\top) \ \sqcap \ \forall \mathsf{child.}(\mathsf{doctor} \sqcup \mathsf{lawyer}).$$

Concepts can be named by declaring that both $C \sqsubseteq D$ and $D \sqsubseteq C$ hold, often abbreviated $C := D$, where C is a concept name and D is an arbitrary description.

The ABox of a knowledge base includes sentences of the form $C(a)$ and $R(a, b)$, where C is a concept, R is a role and a and b are individual objects.

As an example, the sentence $(\exists \mathsf{friend.tall})(marco)$ stored in the ABox tells that the individual $marco$ has at least one tall friend. Similarly, the sentence $\mathsf{friend}(luigi, marco)$ tells that the individuals $luigi$ and $marco$ are friends.

In general, whereas the TBox contain descriptions predicating general properties of the structured domain, the ABox serves the purpose of expressing (possibly complex) properties characterizing individual domain objects.

A description logic knowledge base Δ is given semantics through *interpretations*. An interpretation I is associated with a domain of constants $\mathcal{D}^I$, which can be finite or infinite. Actually $\mathcal{ALCNR}$ has the so-called *finite model property* [CL94], i.e., if a knowledge base has a model, then it has a finite one. As a consequence, inference in all models is equivalent to inference in finite ones, and we can restrict to finite domains. We note that some concept description languages do not have the finite model property [Cal96]. Any interpretation is supposed to:

1. map each individual object a from Δ onto an element a^I of $\mathcal{D}^I$ and each concept C (resp., role R) onto a unary (resp., binary) relation C^I (resp., R^I) over $\mathcal{D}^I$, and
2. satisfy the following equations:

$$
\begin{array}{ll}
\top^I & = \mathcal{D}^I \\
\bot^I & = \emptyset \\
(C \sqcap D)^I & = C^I \cap D^I \\
(C \sqcup D)^I & = C^I \cup D^I \\
(\neg C)^I & = \mathcal{D}^I \setminus C^I \\
(\forall R.C)^I & = \{a \in \mathcal{D}^I \mid \forall b : (a,b) \in R^I \rightarrow b \in C^I\} \\
(\exists R.C)^I & = \{a \in \mathcal{D}^I \mid \exists b : (a,b) \in R^I \wedge b \in C^I\} \\
(\geq n\ R)^I & = \{a \in \mathcal{D}^I \mid \#\{b \mid (a,b) \in R^I\} \geq n\} \\
(\leq n\ R)^I & = \{a \in \mathcal{D}^I \mid \#\{b \mid (a,b) \in R^I\} \leq n\} \\
(P_1 \sqcap \cdots \sqcap P_m)^I & = P_1^I \cap \cdots \cap P_m^I
\end{array}
$$

where $\#S$ denotes the cardinality of the set S.

Following conventions, the Unique Name Assumption is assumed, whereby interpretations map distinct individual objects from the knowledge base into distinct elements of $\mathcal{D}^I$.

An interpretation I of a knowledge base Δ is a *model* for Δ if for each inclusion $C \sqsubseteq D$ defined in its TBox, $C^I \subseteq D^I$ and, furthermore, for each assertion $C(a)$ (resp., $R(a,b)$) $a^I \in C^I$ (resp., $(a^I, b^I) \in R^I$). Finally, Δ logically implies a sentence γ if γ is true in every model of Δ.

Note that, since the Unique Name Assumption holds, we can limit ourselves to consider interpretations mapping individual elements to themselves. In the following, we shall implicitly assume such interpretations.

2.2 Horn rules

Syntactically, a Horn rule has the form

$p(\mathbf{X}) \leftarrow p_1(\mathbf{X}_1), \ldots, p_k(\mathbf{X}_k)$

where $\mathbf{X}$, $\mathbf{X}_i$ are lists of variables or constants. We require that every variable appearing in $\mathbf{X}$ also appears in $\mathbf{X}_1 \cup \cdots \cup \mathbf{X}_n$. The part on the left of the implication sign is usually referred to as the *head* of the rule, whereas that on the right is its *body*. A rule with an empty body and where no variables appear is called a *fact*. Sets of facts can be thought of as tuples stored in relational tables separate from the Horn rules. So, in our framework, $\Delta_\mathcal{R}$ is assumed not to contain facts. Also, as most often assumed, predicates occurring in rule heads in $\Delta_\mathcal{R}$ do not occur in facts.

2.3 Hybrid knowledge bases

A hybrid knowledge base Δ is obtained by coupling a description logic knowledge base with a set of Horn rules and a relational database. For any hybrid knowledge base Δ, $\Delta_\mathcal{R}$ denotes its rule component, $\Delta_\mathcal{E}$ denotes its relational database, $\Delta_\mathcal{T}$ is its TBox and, finally $\Delta_\mathcal{A}$ is its ABox. Predicates appearing in $\Delta_\mathcal{R} \cup \Delta_\mathcal{E}$ but not belonging to the description logic component of Δ are called *ordinary* predicates. For any hybrid knowledge base Δ we require that Horn rule heads in $\Delta_\mathcal{R}$ always contain ordinary predicates. However, descriptive predicates

can occur in facts encoded in relational tables of $\Delta_{\mathcal{E}}$. The conceptual assumption here is that the terminological and assertional components of the hybrid knowledge base completely capture the structuring of the application domain objects, whereas Horn rules are used to describe inference processes to be carried out during query answering.

The semantics of a hybrid knowledge base is obtained by extending that of its description logic component to Horn rules and relational databases. Thus, models of a hybrid knowledge base Δ are models of $\Delta_{\mathcal{T}} \cup \Delta_{\mathcal{A}}$ which satisfy each rule in $\Delta_{\mathcal{R}}$ and each fact in $\Delta_{\mathcal{E}}$. A rule

$p(\mathbf{X}) \leftarrow p_1(\mathbf{X}_1), \ldots, p_k(\mathbf{X}_k)$

is satisfied by an interpretation I if for each mapping α from variables to $\mathcal{D}^I$ such that $\alpha(\mathbf{X_i}) \in p_i^I$, for each i $(1 \leq i \leq k)$, then $\alpha(\mathbf{X}) \in p^I$, where p^I, p_i^I are the extensions assigned by I to p, p_i, respectively. A fact $p(\mathbf{a})$ is satisfied by I if $\mathbf{a}^I \in p^I$. Note that we are not assuming the Closed World Assumption to hold over $\Delta_{\mathcal{E}}$.

In order to clarify what kind of queries hybrid knowledge bases allow to express, we report next an example, taken from [LR96a].

comp, am_co, and am_ass_co are symbols for concepts denoting companies, American companies, and companies having at least one associate company which is American, respectively. asso is a symbol for a role denoting association between companies. Consider the knowledge base Δ consisting of the following components:

1. The TBox $\Delta_{\mathcal{T}}$ contains the following sentences:
   $\mathsf{comp} \sqsubseteq \exists \mathsf{asso.comp}$
   $\mathsf{am\_ass\_co} \sqsubseteq \mathsf{comp} \sqcap \exists \mathsf{asso.am\_co}$
   For instance, the second sentence tells that American associate companies are those companies that have at least one American associate.
2. The ABox $\Delta_{\mathcal{A}}$ contains the assertion $\mathsf{am\_ass\_co}(b)$
3. The rule set $\Delta_{\mathcal{R}}$ consists of the following rule:

$$price(X, usa, high) \leftarrow made\_by(X, Y), \mathsf{comp}(Y), \mathsf{asso}(Y, Z), \mathsf{am\_co}(Z), monopoly(Y, X, usa)$$

4. Finally, the relational database $\Delta_{\mathcal{E}}$ contains the facts $made\_by(a, b)$ and $monopoly(b, a, usa)$.

Then, we can infer $\Delta \models price(a, usa, high)$. Indeed, from $\mathsf{am\_ass\_co}(b)$ we deduce that b has at least one American associate company. That is, there exist a c such that $\mathsf{am\_co}(c)$ which is associate with b. Therefore, from the rule, we infer that $price(a, usa, high)$ is entailed, even if the body of the rule is not "immediately" instantiated by $\Delta_{\mathcal{A}} \cup \Delta_{\mathcal{E}}$.

2.4 Measuring expressive power

Given a language L for transforming information from a designated input knowledge repository (e.g., a relational database or a description logic knowledge base)

to a designated output information, by the expressive power of L we mean the set of all transformations between its input and its output that L can express. For instance, if L is the language of Turing machines, we know that L can express any computable transformation from an input string into an output string. An important body of work done on this subject has been developed in the realm of relational database query languages. Next, we survey the main formal tools used in that realm.

As in most of the papers discussing the expressive power of query languages, in this paper we focus on Boolean queries, which are Boolean transformations defined on relational databases. D is a finite set of domain constants.

Definition 1 (From [CH80]). A *(Boolean) query* is a mapping from relational databases onto $\{True, False\}$ satisfying the following constraints:

1. Q is partial recursive;
2. Q is generic, i.e., for each bijection ρ over D, $\rho(\mathrm{Q(D)}) = \mathrm{Q}(\rho(\mathrm{D}))$.

Thus a query is a computable, generic (in other words, constants are uninterpreted), and well-typed mapping that takes a relational database as the input and returns a Boolean value as the output.

As we have pointed out in Section 1, expressive power of query languages has been measured in at least three different ways. For the sake of completeness, with reference to the third measure, we explain how a query language expressing boolean queries can be related to a complexity class. Each query Q defines a family of databases DB_{Q} as follows:

$$DB_{\mathrm{Q}} = \{\mathrm{D} \mid \mathrm{Q(D)} = True\}.$$

Let $\mathbf{D}$ be a collection of databases. Let $\mathcal{C}$ be a Turing complexity class (e.g., NP). Then the family $\mathbf{D}$ is said to be $\mathcal{C}$-*recognizable* if the problem of deciding if a given database D belongs to $\mathbf{D}$ is in $\mathcal{C}$. Then we say that a query language L *expresses* a database complexity class $\mathcal{C}$ if for each $\mathcal{C}$-recognizable database family $\mathbf{D}$ there is an expression e of L which defines $\mathbf{D}$.

Fagin's theorem [Fag74], cited in Section 1, states that a query language expresses NP iff it has the same expressive power of the language of existentially quantified second-order formulae $(\exists \mathbf{S})\phi$, where $\mathbf{S}$ is a tuple of predicate names and ϕ is a first-order formula. Subsequently, many other similar results relative to other complexity classes have been proved. In our framework, since TBoxes are assumed to contain predicates (concepts, roles) with fixed arity, we will refer in the following to fragments of universal second-order logic with fixed arity [Fag75]. The class NP_k, $k \geq 1$, includes all collections which can be expressed by second-order formulae $(\exists \mathbf{S})\phi$ where each predicate in $\mathbf{S}$ has arity less than or equal to k. Classes NP_k form a proper hierarchy within NP, in that each class NP_k is strictly contained in NP_{k+1} [Fag93]. This hierarchy does not immediately relate to computational complexity: In fact, the smallest class in the hierarchy (NP_1) contains NP complete collections (e.g., the collection of

3-colorable graphs). Nevertheless there are polynomial collections of databases (e.g., the collection of dyadic relations with even number of tuples) that are not in NP_1 [Fag93]. The class NP_1 has interesting properties: as an example, in [Cos93] it is proven that NP_1 differs from coNP_1, while this is a long-standing open question for unbounded NP and coNP.

In the following, we shall show some expressive power results for hybrid languages by relating them to subsets of NP_k, $k \geq 1$, including only those collections expressed by skolemized NP_k formulae $\exists \mathbf{S} \forall \mathbf{X} \exists \mathbf{Y} \phi(\mathbf{X}, \mathbf{Y})$ where $\phi(\mathbf{X}, \mathbf{Y})$ is a quantifier-free first-order formula[1].

It is not known whether imposing skolemization actually excludes some collections from NP_k or not.

3 Which definition of expressiveness for hybrid languages?

Most of the work on expressiveness of query languages assume that the input to the query answering process is a relational database. From the logical point of view, a relational database is a first-order finite interpretation that satisfies the properties expressed in the relational schema (the database is indeed seen as a model of the schema, see [Rei84]). Thus, a query is actually a function over the collection of all possible finite models of the relational schema.

In hybrid languages, the situation is more complicated. Indeed, we have four components involved in query processing, namely, the TBox, the ABox, the set of Horn rules, and the relational database, and we have to decide which are the components that form the input to a query, as opposed to the ones that form the query itself. Moreover, knowledge bases expressed in hybrid languages generally admit several models, in particular when the description logic employs disjunction and/or existential quantification. It follows that a query cannot in general be considered as a computation over a single first-order structure.

The above observations show that we have different options for setting up a formal framework for expressiveness in hybrid languages. The goal of this section is to present some basic considerations about such options.

3.1 Query structure

Let Δ be a hybrid knowledge base, and let $\Delta_{\mathcal{R}}$, $\Delta_{\mathcal{E}}$, $\Delta_{\mathcal{T}}$, and $\Delta_{\mathcal{A}}$ be defined as in Section 2. The components $\Delta_{\mathcal{A}}$ and $\Delta_{\mathcal{E}}$ express properties of individuals, in particular, about their membership in concepts, and their mutual relationships (roles and relations). In other words, $\Delta_{\mathcal{A}}$ and $\Delta_{\mathcal{E}}$ deal with the extensional level of the knowledge base Δ, and, as such, will be considered as parts of the input to the queries.

On the other hand, $\Delta_{\mathcal{T}}$ and $\Delta_{\mathcal{R}}$ deal with the intensional level of the knowledge base. The former expresses general knowledge about the concepts of the

[1] Formulae of this kind are said to be in *doubly prenex normal form* [MP96].

description logic part, whereas the latter is used to define intensional predicates in terms of concepts, roles, and relations.

Following the spirit of deductive databases, it is reasonable to conceive $\Delta_{\mathcal{R}}$ as specifying the computation needed to extract information from the knowledge base. This leads to consider $\Delta_{\mathcal{R}}$ as part of the query. As for $\Delta_{\mathcal{T}}$, we have two possibilities:

- To consider it as a set of integrity constraints over the extensional part of the knowledge base. In this case, $\Delta_{\mathcal{T}}$ is used to specify which are the interpretation structures that are legal with respect to the application, and, therefore, is considered as part of the input to query processing.
- To consider it as part of the specification of the computation to be carried out in the process of query answering, and therefore as part of the query. We note that this option was implicitly used in the example of Section 1, where trying the color of the nodes of the graph was done during query evaluation. An interesting generalization of this idea to other coNP queries leads to viewing $\Delta_{\mathcal{T}}$ as playing its role in the guessing stage, and $\Delta_{\mathcal{R}}$ as specifying the computation to be performed in the checking stage.

We observe that the framework for expressiveness in databases corresponds to the special case where both $\Delta_{\mathcal{A}}$ and $\Delta_{\mathcal{T}}$ are empty, and, therefore, $\Delta_{\mathcal{R}}$ is the query, and $\Delta_{\mathcal{E}}$ is the input.

3.2 Semantical issues

As pointed out before, a query in relational databases is considered as a function applied to a single first-order model in order to compute the desired result. This idea is not directly applicable for devising a formal setting for expressiveness of hybrid knowledge bases. Indeed, we can single out at least three interesting cases of increasing difficulty:

1. The query extracts information from a single finite first-order structure.
2. The output of the query depends on a bounded number of finite structures over a single finite domain.
3. The output of the query depends on an arbitrary collection of structures over a (finite or infinite) domain.

Case (a) occurs when the TBox is seen as part of the query and the ABox is empty, or when both the TBox and the ABox are empty, and is the one where the formal tools developed in relational theory are satisfactory. We note that the example of Section 1 falls under this category.

Case (b) occurs when the ABox and the TBox have a finite Herbrand Universe, which can be considered as the domain of interpretation. The multiplicity of models in this setting is due to the use of disjunction or existential quantification in the description logic component. More generally, whenever the ABox and the TBox use only propositional connectives, i.e., neither quantifications nor number restrictions, we are in case (b).

Another way to achieve the conditions of case (b) is to impose the Unique Domain Assumption, for example by adding a special axiom to the knowledge base, called the Domain Closure Axiom (DCA). The DCA is the formula

$$\forall X \ (X = t_1 \vee \cdots \vee X = t_n),$$

where $t_1, \ldots, t_n$ are all constant symbols occurring in the knowledge base. To see the role of this assumption in query answering, consider the following example:

TBox ($\Delta_\mathcal{T}$): $\top \sqsubseteq \exists \mathtt{q}.\top$
Datalog rules ($\Delta_\mathcal{R}$): $r_1(X,Y) \leftarrow \mathtt{q}(X,Y), r_2(a).$
Relational database ($\Delta_\mathcal{E}$): $r_2(a)$

It is easy to see that, under the DCA, $r_1(a,a)$ is implied by the knowledge base, whereas this is not the case without such an assumption.

Case (c) is the one when no special assumption holds. It is therefore very complex, and seems to be basically unexplored in the literature. Indeed, a precise definition of expressiveness of queries posed to knowledge bases with arbitrary models is missing. Preliminary attempts to deal with this problem are [Var86,BE96]. However, these works adopt various simplifying assumptions, that are not suited to this framework.

4 The expressive power of CARIN knowledge bases

In this section, we present some results about the expressive power of hybrid languages. The framework used in the analysis can be described as follows:

- We assume that the relational database is the input to the query, whereas the TBox and the Horn rules form the query expression.
- We assume that the ABox is empty, and that the TBox uses only propositional connectives. In other words, we are in case (a) according to the classification illustrated in Section 3.2.

In [LR96b] it is proved that the data complexity (the input being the ABox and the relational database) of logical inference in both CARIN-MARC and ROLE-SAFE CARIN is coNP-complete, and we showed in Section 1 that coNP-complete problems are indeed expressed by very simple CARIN knowledge bases, where the Horn component is non-recursive. In this section we prove two results:

1. That ROLE-SAFE CARIN-MARC$^{=,\neq}$ with a non-recursive Horn component expresses all queries that are defined by formulae of the kind $\neg\exists\mathbf{S}\forall\mathbf{X}\exists\mathbf{Y}\phi(\mathbf{X},\mathbf{Y})$, where $\mathbf{S}$ is a list of monadic predicates, ϕ is a quantifier-free first-order formula, and $\mathbf{X}$, $\mathbf{Y}$ are lists of variables, provided that the input finite structure is given in suitable form, as specified below. The superscript $^{=,\neq}$ denotes availability of pre-interpreted symbols for equality and inequality to be used in Datalog rules. Formulae $\neg\exists\mathbf{S}\forall\mathbf{X}\exists\mathbf{Y}\phi(\mathbf{X},\mathbf{Y})$ define queries that form a subset of monadic coNP queries (hereafter, called coNP$_1$ cf.

[Fag75,Fag93,Cos93]), which contains several coNP complete queries (e.g., the complement of 3-colorability of a graph). At the moment, we do not know whether this result can be generalized to the entire set of monadic coNP queries.

2. That CARIN-MARC$^=$ with a non-recursive Horn component enriched with propositional inclusion axioms on primitive roles expresses all queries that are defined by formulae of the kind $\neg\exists\mathbf{S}'\forall\mathbf{X}\exists\mathbf{Y}\phi'(\mathbf{X},\mathbf{Y})$, where $\mathbf{S}'$ is a list of predicates with arity at most 2 and ϕ' is a quantifier-free first-order formula, provided that the input finite structure is given in suitable form, as specified below. Analogously to the previous case, such formulae define queries that form a subset of dyadic coNP (coNP$_2$), and we do not know whether this result can be generalized to the entire set of dyadic coNP queries.

The coNP-completeness of querying the classes of CARIN-MARC and ROLE-SAFE CARIN knowledge bases, whose definition has been recalled in Section 1, established in [LR96b] serves also as an upper bound to the expressiveness when the ABox is empty. In other words we know that no CARIN-MARC or ROLE-SAFE CARIN knowledge base can express queries which are not in coNP.

In the following, σ denotes a fixed set of relational symbols not including equality "=" and $\mathbf{S}$ denotes a list of variables ranging over monadic relational symbols distinct from those in σ. By Fagin's theorem [Fag74], any NP-recognizable collection $\mathbf{D}$ of finite structures over σ is defined by a second-order existentially quantified formula. In particular, NP-recognizable collections $\mathbf{D}$ of finite structures, defined by formulas where all existentially quantified relational symbols are 1-ary, form the set of NP$_1$ collections.

In the following, we deal with skolemized second-order formulae of the following kind:

$$\phi = (\exists\mathbf{S})(\forall\mathbf{X})(\exists\mathbf{Y})(\theta_1(\mathbf{X},\mathbf{Y}) \vee \cdots \vee \theta_k(\mathbf{X},\mathbf{Y})), \tag{1}$$

where $\theta_1,\ldots,\theta_k$ are conjunctions of literals involving relational symbols in σ and $\mathbf{S}$, plus relational symbol "=", and all relational symbols in $\mathbf{S}$ are constrained to be monadic. The set of variable occurrences in θ_i is contained in $\mathbf{X}\cup\mathbf{Y}$. As usual, "=" is always interpreted as "equality". The set of uninterpreted relational symbols occurring in formula (1) –i.e., $\sigma\cup\mathbf{S}$– will be denoted either by $\mathcal{L}$ or by $\{a_1,\ldots,a_l\}$. In the following art(a) denotes the arity of a predicate a.

We illustrate a method that transforms a formula ϕ of the kind (1) and a finite structure D into a ROLE-SAFE CARIN-MARC knowledge base $\Delta(\phi,D)$ and a query γ. Both $\Delta(\phi,D)$ and γ use an enlarged set of relational symbols $\mathcal{L}'$ which is built as follows:

1. each relational symbol $a\in\mathcal{L}$ is in $\mathcal{L}'$;
2. for each relational symbol $a\in\mathcal{L}$ there is one relational symbol $\overline{a}$ with the same arity as a in $\mathcal{L}'$;
3. there are relational symbols t and U' with the same arity as $\mathbf{X}$, U'' with the same arity as $\mathbf{Y}$, and U with arity 1;
4. there are two binary "built-in" relational symbols eq and $\overline{eq}$ denoting equality and inequality, respectively.

The ROLE-SAFE CARIN-MARC knowledge base $\Delta(\phi, D) = \Delta_{\mathcal{T}}(\phi) \cup \Delta_{\mathcal{R}}(\phi) \cup \Delta_{\mathcal{E}}(\phi, D)$ is built as follows:

1. For each relational symbol $s \in \mathbf{S}$, the following axioms are in $\Delta_{\mathcal{T}}(\phi)$:
$$\top \sqsubseteq s \sqcup \overline{s}, \qquad s \sqsubseteq \neg \overline{s}$$
2. For each relational symbol $a \in \sigma$, the following $\mathrm{art}(a)$ rules are in $\Delta_{\mathcal{R}}(\phi)$;
$$U(X) \leftarrow a(X, Y_1, \ldots, Y_{\mathrm{art}(a)-1})$$
$$\ldots$$
$$U(X) \leftarrow a(Y_1, \ldots, Y_{\mathrm{art}(a)-1}, X)$$
3. the following rules are also in $\Delta_{\mathcal{R}}(\phi)$:
$$U'(X_1, \ldots, X_n) \leftarrow U(X_1), \ldots, U(X_n)$$
$$U''(Y_1, \ldots, Y_m) \leftarrow U(Y_1), \ldots, U(Y_m)$$
where n and m are the number of variables occurring in $\mathbf{X}$ and $\mathbf{Y}$, respectively.
4. For each conjunct
$$\theta_i(\mathbf{X}, \mathbf{Y}) = \neg w_1(\mathbf{X}, \mathbf{Y}) \wedge \cdots \wedge \neg w_n(\mathbf{X}, \mathbf{Y}) \wedge w_{n+1}(\mathbf{X}, \mathbf{Y}) \wedge \cdots \wedge w_{n+m}(\mathbf{X}, \mathbf{Y})$$
$(1 \leq i \leq k)$ in ϕ, the rule
$$t(\mathbf{X}) \leftarrow \overline{v_1}(\mathbf{X}, \mathbf{Y}), \ldots, \overline{v_n}(\mathbf{X}, \mathbf{Y}), v_{n+1}(\mathbf{X}, \mathbf{Y}), \ldots, v_{n+m}(\mathbf{X}, \mathbf{Y}), U'(\mathbf{X}), U''(\mathbf{Y})$$
is in $\Delta_{\mathcal{R}}(\phi)$, where:
   - $\overline{v_i}$ $(1 \leq i \leq n)$ is:
     - $\overline{eq}$, if w_i is "=" (this is just used here to make the syntax used for equality uniform to that used for predicates in $\mathbf{S}$),
     - $\overline{w_i}$, otherwise;
   - v_{n+i} $(1 \leq i \leq m)$ is:
     - eq, if w_{n+i} is "=",
     - w_{n+i}, otherwise.

Furthermore, the query γ is
$$(\exists X_1, \ldots, X_{art(t)})\ \neg t(X_1, \ldots, X_{art(t)}), U'(X_1, \ldots, X_{art(t)}).$$

We remark that $\Delta_{\mathcal{T}}(\phi) \cup \Delta_{\mathcal{R}}(\phi)$ is a ROLE-SAFE CARIN-MARC knowledge base.

Now, given a finite structure D, we define the complementary structure $\overline{D}$ as follows. For each relational symbol $r \in D$ there is a relational symbol $\overline{r}$ in $\overline{D}$ with the same arity as r. Then, for each relational symbol $\overline{r}$ in $\overline{D}$ and for each tuple $\mathbf{t}$ of constants from D, it holds that $\overline{D} \models \overline{r}(\mathbf{t})$ iff $D \not\models r(\mathbf{t})$. Thus, finally, let $\Delta_{\mathcal{E}}(\phi, D) = D \cup \overline{D}$, and $\Delta(\phi, D) = \Delta_{\mathcal{T}}(\phi) \cup \Delta_{\mathcal{R}}(\phi) \cup \Delta_{\mathcal{E}}(\phi, D)$. We are now ready for the first main result about expressive power of ROLE-SAFE CARIN-MARC.

Theorem 2. *For any skolemized NP_1 collection* $\mathbf{D}$ *of finite structures over* σ *–characterized by a formula* ϕ *of the kind (1)– and for any finite structure* D*, the* ROLE-SAFE CARIN-MARC *knowledge base* $\Delta(\phi, D)$ *built according to the above rules is such that* D *is in* $\mathbf{D}$ *if and only if* $\Delta(\phi, D) \not\models \gamma$.

In other words, the theorem says that each collection of finite structures in skolemized coNP_1 is definable by a ROLE-SAFE CARIN-MARC knowledge base $\Delta(\phi, D)$ and query γ.

To allow role axioms to occur in $\Delta_{\mathcal{T}}(\phi)$ enhances the expressive power of the language. Indeed, consider formulae of the form:

$$\phi' = (\exists \mathbf{S}')(\forall \mathbf{X})(\exists \mathbf{Y})(\theta_1(\mathbf{X}, \mathbf{Y}) \vee \cdots \vee \theta_k(\mathbf{X}, \mathbf{Y})), \tag{2}$$

where $\theta_1, \ldots, \theta_k$ are conjunctions of literals involving relational symbols in σ and $\mathbf{S}'$, plus relational symbol "=", and all relational symbols in $\mathbf{S}'$ are constrained to be either monadic or dyadic. Such formulae define a subset of NP_2 that contains NP-complete problems. We note that there are collections of polynomial-time recognizable databases (e.g., the collection of ternary relations with even number of tuples) that are not in NP_2 [Fag93], and that NP_2 strictly contains NP_1 (e.g., the collection of dyadic relations with even number of tuples is in NP_2).

In this case, we do assume the language includes built-in predicate for equality only (denoted also *eq* by analogy to the previous theorem). Inequality will be derived. To illustrate the transformation of a formula ϕ' of the kind (2) into a CARIN-MARC$^{=}$ knowledge base $\Delta'(\phi', D) = \Delta'_{\mathcal{T}}(\phi') \cup \Delta_{\mathcal{R}}(\phi') \cup \Delta_{\mathcal{E}}(\phi', D)$ with axioms on roles and a query $\gamma' = \gamma$, we modify the translation provided for the monadic case. $\Delta'(\phi', D)$ is equal to $\Delta(\phi, D)$ except that for each dyadic relational symbol $s' \in \mathbf{S}'$, the following role axioms are in $\Delta'_{\mathcal{T}}(\phi')$:

$$\top \times \top \sqsubseteq s' \sqcup \overline{s'} \qquad s' \sqsubseteq \neg\overline{s'}$$
$$\top \times \top \sqsubseteq eq \sqcup \overline{eq} \qquad eq \sqsubseteq \neg\overline{eq}$$

We are now ready for our second result about expressive power of CARIN languages.

Theorem 3. *For any skolemized NP_2 collection* $\mathbf{D}$ *of finite structures over* σ *–characterized by a formula* ϕ' *of the kind (2)– and for any finite structure* D*, the* CARIN-MARC$^{=}$ *knowledge base* $\Delta'(\phi', D)$ *built according to the above rules is such that* D *is in* $\mathbf{D}$ *if and only if* $\Delta'(\phi', D) \not\models \gamma'$.

5 Current work on negation

In this section, we overview some current research we are conducting which focuses on extending our hybrid languages by adding stratified negation in its rule component. In particular, in the following we shall define the semantics of such extended languages and show that their data complexity is still within coNP.

We begin with syntax. A stratified Datalog$^\neg$ program is a set of rules of the form:

$$p(\mathbf{X}) \leftarrow p_1(\mathbf{X_1}), \ldots p_k(\mathbf{X_k}), \neg q_1(\mathbf{Y_1}), \ldots, \neg q_n(\mathbf{Y_n})$$

such that it is possible to assign a level $lev(p)$ to each predicate symbol p and for each rule of the form above, $lev(p) > lev(q_i)$, $1 \leq i \leq n$, and $lev(p) \geq lev(p_i)$, $1 \leq i \leq k$. Then, the program can be partitioned into strata $\{S_i\}$, each stratum S_i containing all the rules whose head predicate symbol has been assigned the level i.

The semantics of a Datalog$^\neg$ program on their own is given by its *stratified* or *perfect* model, which is constructed by evaluating the program a stratum at a time, beginning from the lowest stratum. Because of stratification, each time a stratum is evaluated, the negative literals in the bodies of its rules have already been computed, so that they can be treated as base predicates.

Define now a stratified hybrid knowledge base Δ to be a knowledge base having a stratified Datalog$^\neg$ program as its rule component $\Delta_\mathcal{R}$.

To define the semantics of stratified hybrid knowledge bases, we proceed as follows. Consider $\Delta_{nR} = \Delta \setminus \Delta_\mathcal{R}$. Δ_{nR} has in general a set of models $\{M_i\}$. In our assumptions, each of these models can be looked at as an extensional relational database. Then, the semantics of Δ is given by the set of perfect models obtained by union of $\Delta_\mathcal{R}$ with each one such M_i. More formally, let γ be a query and Δ be a stratified hybrid knowledge base. Then, $\Delta \models \gamma$ iff for each perfect model M of $\Delta_\mathcal{R} \cup N$, it holds that $M \models \gamma$, where N is a model of Δ_{nR}.

Using stratified rule components probably enhances the expressive power of hybrid knowledge bases, even if we do not have a formal proof yet. However, it clearly improves their amenability at expressing interesting queries in a simple way, as the one presented below.

Datalog rules ($\Delta'_\mathcal{R}$):

$$\begin{aligned} non\_red\_path(X,Y) &\leftarrow edge(X,Y), \neg red(X), \neg red(Y). \\ non\_red\_path(X,Y) &\leftarrow non\_red\_path(X,Z), edge(Z,Y), \neg red(Y). \\ non\_red &\leftarrow \neg non\_red\_path(a,b). \\ non\_red &\leftarrow non\_3\_col. \end{aligned}$$

The rules above, when added to the 3-coloring example given in Section 1, with the query $\gamma = non\_red$ solve the following problem.

Does there exists a 3-coloring of G such that there is at least one path between nodes a and b which does not use red-colored nodes?

But what is the computational cost of adding stratified negation to hybrid knowledge bases? We can prove that the data complexity (i.e., the complexity of query answering measured as the size of $\Delta_\mathcal{E} \cup \Delta_\mathcal{A}$ varies while the size of the rest of the knowledge base remains fixed) for hybrid knowledge bases remains unchanged when stratified negation is added.

Indeed, to show that a query γ is not entailed by a knowledge base Δ it is sufficient to show a perfect model M of Δ which does not entail γ. To this end, we do the following: (1) guess an interpretation M of Δ; (2) verify that M is a perfect model of Δ; (3) verify that M does not entail γ. Clearly, step (3) can be done in polynomial time. As for step (2), it can be carried out into two substeps: (2.1) verify that M is a model of Δ_{nR}; (2.2) verify that it is a perfect model of $\Delta_{\mathcal{R}} \cup M_{nR}$, where M_{nR} is the projection of M on predicate symbols occurring in Δ_{nR} (recall that such a set is disjoint from the set of predicate symbols occurring in the rule component of the knowledge base). Both step (2.1) and step (2.2) are carried out in polynomial time. The result follows considering that the size of M is polynomial in the size of $\Delta_{\mathcal{E}} \cup \Delta_{\mathcal{A}}$

Current research we are conducting focuses on establishing the expressive power of stratified hybrid knowledge bases as well as of other variants, where more involved form of negation is allowed (e.g., stable negation).

6 Conclusions

In the present paper we discussed the possibility of exploiting the formal analysis tools developed in the database field within the context of description logics augmented with rules. Also, we proved some results on the expressive power of two hybrid languages, in particular obtaining two lower bounds on their expressiveness (Theorems 2 and 3). Upper bound for the expressiveness of the former language follows from the results of [LR96b]. We still have to investigate the upper bound of expressiveness of the latter language.

In this work we assumed empty ABoxes. Nevertheless the results we presented are valid also if the ABox contains positive atomic assertions such as $red(node1)$. Furthermore, it is important to stress that by allowing predicates with any arity to appear in a propositional TBox, we obtain languages capturing all coNP properties.

Several questions are still open. Regarding the two languages we have analyzed in this paper, (1) determine whether there are queries in coNP_2 which cannot be expressed by the former language, and (2) determine whether the languages express all (even non-skolemized) queries in coNP_1, resp. coNP_2. More generally, how to define the expressive power over finite formulae possibly denoting infinitely many models?

Acknowledgements

The authors would like to thank Alon Levy and Rick Hull for their useful comments. Thanks are also due to the anonymous referees for their precious suggestions.

This work has been supported by ASI (Italian Space Agency), MURST (Italian Ministry for University and Scientific and Technological Research), CNR (Italian Research Council), and the European Community under the Esprit Project 22469 - DWQ.

References

[AV92] S. Abiteboul and V. Vianu. Expressive power of query languages. In J. D. Ullman, editor, *Theoretical Studies in Computer Science*. Academic Press, 1992.

[Baa96] Franz Baader. A formal definition for the expressive power of terminological knowledge representation languages. *Journal of Logic and Computation*, 6:33–54, 1996.

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

[BE96] P. Bonatti and T. Eiter. Querying disjunctive databases through nonmonotonic logics. *Theoretical Computer Science*, 160:321–363, 1996.

[Bor96] Alexander Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence Journal*, 82:353–367, 1996.

[Cal96] Diego Calvanese. Finite model reasoning in description logics. In Luigia C. Aiello, John Doyle, and Stuart C. Shapiro, editors, *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 292–303. Morgan Kaufmann, Los Altos, 1996.

[CH80] A. Chandra and D. Harel. Computable queries for relational databases. *Journal of Computer and System Sciences*, 21:156–178, 1980.

[CL94] Diego Calvanese and Maurizio Lenzerini. Making object-oriented schemas more expressive. In *Proceedings of the Thirteenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS-94)*, pages 243–254, Minneapolis, 1994. ACM Press and Addison Wesley.

[Cos93] S. S. Cosmadakis. Logical reducibility and monadic NP. In *Proceedings of the Thirtyfourth Annual Symposium on the Foundations of Computer Science (FOCS-93)*, 1993.

[DLNS91] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. A hybrid system integrating Datalog and concept languages. In *Proceedings of the Second Conference of the Italian Association for Artificial Intelligence (AI*IA-91)*, number 549 in Lecture Notes In Artificial Intelligence. Springer-Verlag, 1991.

[DLNS97] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. $\mathcal{AL}$-log: Integrating Datalog and description logics. *Journal of Intelligent Information Systems*, 1997. To appear.

[EGM97] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM Transactions on Database Systems*, 22:364–418, 1997.

[Fag74] R. Fagin. Generalized First-Order Spectra and Polynomial-Time Recognizable Sets. In R. M. Karp, editor, *Complexity of Computation*, pages 43–74. AMS, 1974.

[Fag75] Ronald Fagin. Monadic generalized spectra. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 21:89–96, 1975.

[Fag93] Ronald Fagin. Finite-model theory – a personal perspective. *Theoretical Computer Science*, 116:3–31, 1993.

[Kan90] P. Kanellakis. Elements of relational database theory. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 17. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[LR96a] Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining Horn rules and description logics. In *Proceedings of the*

Twelfth European Conference on Artificial Intelligence (ECAI-96), pages 323–327, 1996.

[LR96b] Alon Y. Levy and Marie-Christine Rousset. The limits on combining recursive Horn rules with description logics. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 577–584, 1996.

[MP96] J.A. Makowsky and Y.B. Pnueli. Arity and alternation in second-order logic. *Annals of Pure and Applied Logics*, 78:189–202, 1996.

[Rei84] Raymond Reiter. Towards a logical reconstruction of relational database theory. In M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, editors, *On Conceptual Modelling*. Springer-Verlag, 1984.

[Sch95] J. S. Schlipf. The expressive powers of the logic programming semantics. *Journal of Computer and System Sciences*, 51:64–86, 1995.

[Var86] M. Y. Vardi. Querying logical databases. *Journal of Computer and System Sciences*, 33:142–160, 1986.