Laboratoire
d'Informatique
de Robotique
et de Microélectronique
de Montpellier
LIRMM

RESEARCH REPORT

On the Complexity of Deduction in Existential Conjunctive First Order Logic with Atomic Negation
(Long Version)

Marie-Laure Mugnier mugnier@lirmm.fr

Geneviève Simonet simonet@lirmm.fr

Michaël Thomazo mthomazo@dptinfo.ens-cachan.fr

September 2009 R.R.LIRMM 09026

161, rue Ada • 34392 Montpellier Cedex 5 • France
Tel. : 33 (0) 4 67 41 85 85 • Fax : 33 (0) 4 67 41 85 00 • www.lirmm.fr

Abstract

We consider the deduction problem in the fragment of first-order logic (FOL) composed of existentially closed conjunctions of literals (without functions), denoted FOL$\{\exists, \wedge, \neg_a\}$. This problem can be recast as several fundamental problems in artificial intelligence and databases, namely query containment for conjunctive queries with negation, clause entailment for clauses without functions and query answering with incomplete information for boolean conjunctive queries with negation over a fact base. Deduction in FOL$\{\exists, \wedge, \neg_a\}$ is Π_2^P-complete, whereas it is only NP-complete when the formulas contain no negation. We investigate the role of specific literals in this complexity increase. These literals have the property of being "exchangeable", with this notion taking the structure of the formulas into account. To focus on the structure of formulas, we see them as labeled graphs. Graph homomorphism, which provides a sound and complete proof procedure for positive formulas, is at the core of this study. Let Deduction$_k$ be the following family of problems: given two formulas g and h in FOL$\{\exists, \wedge, \neg_a\}$, such that g has at most k pairs of exchangeable literals, can g be deduced from h? The main results are that Deduction$_k$ is NP-complete if $k \leq 1$, and in P^{NP} for any value of k; moreover, it is both NP-difficult and co-NP-difficult for $k \geq 3$. As a corollary of our proofs, we are able to classify exactly previous problems when g is decomposable into a tree. Finally, several complementary results and extensions are provided.

Keywords: Complexity, first-order logic, deduction, negation, graphs, homomorphism, query containment, clause implication, conceptual graphs.

Remark: *A shorter version has been submitted for publication to a journal. This shorter version does not integrate the alternative proofs of our results based on a logical approach (Sect. 5) nor the extension to a preorder on the set of predicates (Sect. 6).*

Contents

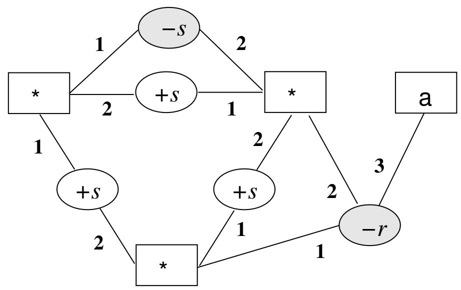
1 Introduction

In this paper, we study the complexity of deduction checking in the fragment of first-order logic (FOL), composed of existentially closed conjunctions of literals. Literals may contain constants but no other function symbols. FOL$\{\exists, \wedge, \neg_a\}$ denotes this fragment, and FOL$\{\exists, \wedge\}$ is the subfragment with positive literals only. The DEDUCTION problem in a given fragment takes two formulas g and h of this fragment as input, and asks if g can be deduced from h.

Equivalent problems. FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION can be seen as a representative of several fundamental problems in artificial intelligence and databases. It can be immediately recast as a *query containment* checking problem, which is one of the fundamental problems in databases. This problem takes two queries q_1 and q_2 as input, and asks if q_1 is contained in q_2, i.e. if the set of answers to q_1 is included in the set of answers to q_2 for all databases (e.g. [AHV95]). Algorithms based on query containment can be used to solve various problems, such as query evaluation and optimization [CM77, ASU79], rewriting queries using views [Hal01], detecting independance of queries from database updates [LS93], etc. The so-called (positive) *conjunctive queries* form a class of natural and frequently used queries and are considered as the basic database queries [CM77, Ull89]. Their expressive power is equivalent to the select-join-project queries of relational algebra and to non-recursive Datalog rules. Conjunctive queries with negation extend this class with negation on atoms. Query containment checking for conjunctive queries with negation (resp. positive conjunctive queries) is essentially the same problem as FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION (resp. FOL$\{\exists, \wedge\}$–DEDUCTION), in the sense that there are natural polynomial reductions from one to another, which preserve the structure of the objects. Another related problem in artificial intelligence is the *clause entailment* problem, a basic problem in inductive logic programming [MR94]: given two clauses C_1 and C_2, does C_1 entail C_2? If we consider first-order clauses, i.e. universally closed disjunctions of literals, without function symbols, by contraposition, we obtain an instance of FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION. Let us now look at this from a knowledge representation perspective. A key problem is *query answering*, which, generally speaking, takes a knowledge base and a query as input and asks for the set of answers to the query that can be retrieved from the knowledge base. When the query is a boolean query, i.e. with a yes/no answer, the problem can be recast as checking whether the query can be deduced from the knowledge base. In the case where the knowledge base is simply composed of a set of positive and negative facts, i.e. existentially closed conjunctions of literals, and the query is a boolean conjunctive query with negation, we obtain FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION. Finally, even if this aspect is out of the scope of the present paper, let us mention that a partial order on predicates, or more generally a preorder, can be taken into account without increasing complexity. This allows to represent a knowledge base with a light ontology and a set of facts built on this ontology. We then obtain FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION extended to preordered predicates, which is exactly the deduction problem in a fragment of conceptual graphs, called *polarized conceptual graphs* [Ker01][ML07].

Complexity and "exchangeable" literals. Whereas FOL$\{\exists, \wedge\}$-DEDUCTION is "only" NP-complete, FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION is Π_2^P-complete[1] (see Section 7). Some specific cases where FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION has a lower complexity are known but they enforce strong restrictions on the problem instances:

[1] Π_2^P is $(co\text{-}NP)^{NP}$.



$\exists x \exists y \exists z(s(x,y) \wedge s(y,z) \wedge s(z,x) \wedge \neg s(x,z) \wedge \neg r(y,z,a)))$

Figure 1: A polarized graph

briefly said, if g does not contain any pair of opposite and unifiable literals[2], then FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION becomes NP-complete (see Section 7). The aim of this paper is to investigate the complexity gap between deduction checking in FOL$\{\exists, \wedge\}$ and FOL$\{\exists, \wedge, \neg_a\}$. For that, we study the role of specific pairs of literals in the complexity increase. These literals have the property of being "exchangeable", with this notion being relative not only to the literals themselves, but also to the structure of both formulas. We show that these literals are indeed responsible for the complexity increase, in the sense that if the number of exchangeable literals in g is *bounded*, then the complexity falls into lower classes of the polynomial hierarchy. The complexity results proven in this paper generalize the results obtained in the various variants of the problem (for instance the query inclusion problem or the clause implication problem).

Graph Tools. We shall see formulas as labeled graphs to focus on their structure and rely on graph notions like paths, connectivity or cyclicity. These graphs are called polarized graphs (PGs) (name borrowed to [Ker01] in the context of conceptual graphs). More specifically, a FOL$\{\exists, \wedge, \neg_a\}$ formula is represented as a bipartite graph with two kinds of nodes: relation nodes and term nodes. Each term of the formula becomes a term node, labeled $*$ if it is a variable, otherwise by the constant itself. A positive (resp. negative) literal with predicate symbol r becomes a relation node labeled $+r$ (resp. $-r$) and it is linked to the nodes assigned to its terms. The numbers on edges correspond to the position of each term in the literal. See Figure 1 for an example. In the sequel of this section, formulas are denoted by small letters (g and h) and the associated graphs by the corresponding capital letters (G and H).

Homomorphism is a core notion in this study. Basically, a homomorphism

[2] i.e. in the form $p(u)$ and $\neg p(v)$, where $p(u)$ and $p(v)$ are unifiable.

from one algebraic structure to another maps the elements of the first structure to elements of the second structure while preserving the relations between elements. A homomorphism π from a graph G to a graph H is a mapping from nodes of G to nodes of H, which preserves edges, i.e. if xy is an edge of G then $\pi(x)\pi(y)$ is an edge of H. Since polarized graphs are labeled, there are additional conditions on labels: a relation node is mapped to a node with the same label; a term node can be mapped to any term node if it is labeled $*$, otherwise it is mapped to a node with the same constant. Numbers on edges are preserved. Let us point out that, given two formulas g and h in FOL$\{\exists, \wedge, \neg_a\}$, one can identify the notions of a *substitution* σ for variables in g, s.t. the literals of $\sigma(g)$ are contained in h, and a PG homomorphism from G to H. FOL$\{\exists, \wedge\}$-DEDUCTION can be solved by a substitution check, or equivalently by a homomorphism check on the PGs assigned to the formulas. This homomorphism check still provides a sound procedure for deduction in FOL$\{\exists, \wedge, \neg_a\}$, i.e. the existence of a homomorphism from G to H implies that g can be deduced from h, but of course it is no longer complete, i.e. g may be deducible from h even if there is no homomorphism from G to H. FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION can be recast as a problem on PGs involving a number of homomorphism checks exponential in the size of H.

Contributions of the paper. The results achieved in this paper can be summarized as follows. We first point out that if g has *no* pair of exchangeable literals, then FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION has the same complexity as in the positive fragment (indeed it can be computed by a homomorphism check, thus is NP-complete). It is then proven that the problem remains NP-complete if g has *one* pair of exchangeable literals. A natural question that arises is whether the complexity of deduction checking decreases when g has a *bounded* number of exchangeable literals. Let DEDUCTION$_k$ be the following family of problems: given two formulas g and h in FOL$\{\exists, \wedge, \neg_a\}$, such that g has at most k pairs of exchangeable literals, can g be deduced from h? It is proven that, for any k, DEDUCTION$_k$ is in P^{NP}, i.e. Δ_2^P. A complementary result is that DEDUCTION$_k$ is co-NP-difficult for $k = 3$. When g represents a query and h a base of facts, criteria that decrease the complexity and depend on g rather than h are relevant, because the query can be considered as small with respect to the fact base, and has generally a simple structure (while one cannot expect the fact base to have a special structure). In particular, when g has a structure decomposable into a tree (we will precise this point later), then homomorphism checking is polynomial; in this case, we point out that FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION is co-NP-complete; moreover, a corollary of previous results' proofs is that in general DEDUCTION$_k$ remains co-NP-complete for any $k \geq 3$ and is in P if $k \leq 1$. Table 1 summarizes these results. The recognition problem associ-

number of exchangeable pairs in g	arbitrary g	g decomposable into a tree
not bounded	Π_2^P-complete (*)	co-NP-complete
0	NP-complete	P
1 (**)	NP-complete	P
bounded by $k \geq 3$	NP-difficult co-NP-difficult and P^{NP}	co-NP-complete

(*) already known result

(**) or with an unbounded number of exchangeable pairs and a single positive (resp. negative) exchangeable literal

Table 1: Main complexity results

ated with DEDUCTION$_k$, i.e. whether g possesses at most k pairs of exchangeable literals, is co-NP-complete. Note however that all results still hold if we apply weaker criteria that bound the number of potentially exchangeable literals and can be checked in polynomial time.

Several complementary results and extensions are provided. First, we point out that a FOL$\{\exists, \wedge, \neg_a\}$ formula can be partitioned into subsets of literals called *pieces* (this notion is actually defined on PGs as it correspond to a graph decomposition notion), such that the bound on the number of pairs of exchangeable literals can be made relative to each piece of g instead of the entire g, i.e. in all results, condition "g has at most k pairs of exchangeable literals" can be relaxed into "each piece of g has at most k pairs of exchangeable literals". Secondly, we provide alternative proofs of our results based on a logical approach; as a side result, we clarify the relationships between logical and graph notions involved in this study. Finally, previous results are extended in two ways: we show that a preorder on the set of predicates can be considered without complexity increasing, which allows us to take a light ontology into account; we also refine several notions related to exchangeable literals, which allows to further decrease their number.

Paper organization. Section 2 introduces the graph framework and known results. Section 3 studies properties of exchangeable literals. Section 4 contains our main complexity results. Section 5 and Section 6 are respectively devoted to the logical approach and to extensions. Section 7 synthesizes related works and concludes on open problems.

2 Preliminaries

Without loss of generality, we assume that logical formulas are in prenex form, i.e. all quantifiers are at the beginning of the formula. Equality is not considered but all results are easily extended to it (see in particular [LM06], which shows how to include equality and inequality in the framework of polarized conceptual graphs). Since we do not consider function symbols other than constants, a *logical language* is a pair $(\mathcal{R}, \mathcal{I})$, where $\mathcal{R}$ is the set of predicates and $\mathcal{I}$ is the set of constants. The *terms* on $(\mathcal{R}, \mathcal{I})$ are thus constants in $\mathcal{I}$ or variables. An *atom* on $(\mathcal{R}, \mathcal{I})$ is of form $p(t_1, \ldots, t_k)$, where $p \in \mathcal{R}$ and, for all j in $1..k$, t_j is a term on $(\mathcal{R}, \mathcal{I})$. A *literal* is an atom (positive literal) or the negation of an atom (negative literal). A FOL$\{\exists, \wedge, \neg_a\}$ formula on $(\mathcal{R}, \mathcal{I})$ is a closed formula in the form $\exists x_1 \ldots x_q (l_1 \wedge \ldots \wedge l_p)$, where, for all i in $1..p$, l_i is a literal whose variables are in $\{x_1, \ldots, x_q\}$. Without loss of generality, we will sometimes view such a formula as the set of its literals. A FOL$\{\exists, \wedge\}$ formula has only positive literals. The set of *atoms of a formula* is the set of atoms occurring positively or negatively in its literals.

As explained in the introduction, it is convenient to see a FOL$\{\exists, \wedge, \neg_a\}$ formula as a bipartite labeled graph, that we call a polarized graph (PG). The following definitions and results about polarized graphs are mainly based on [LM07] and [ML07].

Definition 1 (polarized graph) *Let us consider a vocabulary $\mathcal{V} = (\mathcal{R}, \mathcal{I})$ where $\mathcal{R}$ is a finite set of relation names of any arity and $\mathcal{I}$ a set of individual names, or constants. A* polarized graph *(PG) is a finite undirected bipartite labeled multigraph $G = (R,\ T,\ E,\ l)$ where R and T are the (disjoint) sets of nodes, respectively called set of* relation *nodes and set of* term *nodes, E is the family of edges (there may be several edges with the same extremities, thus strictly speaking, a PG is a multigraph and not a graph) and l is the label mapping. For $x \in R$, $l(x) = +r$ (x is called a positive relation node) or $l(x) = -r$ (x is called a negative relation node) where $r \in \mathcal{R}$; the degree of x (i.e. the number of edges incident to it) must be equal to the arity of r; furthermore, the edges incident to x are totally ordered, which is represented by labeling edges from 1 to the degree of x. An edge labeled i between a relation node x and a term node t is denoted $(x,\ i,\ t)$. For $t \in T$, either $l(t) = *$ (t is called a variable node) or $l(t) \in \mathcal{I}$ (t is called a constant node).*

A PG is said to be *normal* if each constant of $\mathcal{I}$ appears at most once in it. In the following, a PG is assumed to be normal unless otherwise specified. Moreover, we assume that PGs do not have redundant relation nodes (i.e. with the same label and the same ith neighbors).

A FOL$\{\exists, \wedge, \neg_a\}$ formula g on a logical language $(\mathcal{R}, \mathcal{I})$, is translated into a PG G on a vocabulary $\mathcal{V} = (\mathcal{R}, \mathcal{I})$, with the following natural bijections: from variables in g to variable nodes in G, from constants in g to constant nodes in G (s.t. a constant a yields a node with label a), from positive (resp. negative) literals in g to positive (resp. negative) relation nodes in G (s.t. the predicate and polarity of a literal yield the label of the relation node). For each argument t_i of a literal l, there is an edge $(x,\ i,\ t)$, where x is the relation node assigned to l and t is the term node assigned to t_i. There is thus a bijection from the set of FOL$\{\exists, \wedge, \neg_a\}$ formulas on a logical language $(\mathcal{R}, \mathcal{I})$ to the set of normal PGs without isolated term nodes[3] on a vocabulary $\mathcal{V} = (\mathcal{R}, \mathcal{I})$. This bijection is within an isomorphism for graphs and within a variable renaming for formulas. In the following, since we work on the graph representation of formulas, we will consider PGs as the basic constructs, and see formulas as their logical meaning. The mapping from PGs without isolated term nodes to formulas is called Φ.

Notations. Let $+r(t_1, \ldots, t_k)$ (resp. $-r(t_1, \ldots, t_k)$) denote the subgraph induced by a positive (resp. negative) relation node with label $+r$ (resp. $-r$) and its list of neighbors $t_1, \ldots, t_k$. By analogy with its logical translation $r(t_1, \ldots, t_k)$ (resp. $\neg r(t_1, \ldots, t_k)$), in which t_i denotes the term assigned to the term node t_i, we also call it a *literal*. $\sim r$ denotes a label with relation name r, where $\sim$ can be $+$ or $-$. Given a literal (resp. a relation label) l, $\overline{l}$ denotes the *complementary* literal (resp. relation label) of l, i.e. it is obtained from l by reversing its sign. Letters u, v and w are used to denote a tuple $(t_1, \ldots, t_k)$ of terms (or term nodes). Thus $\sim r(u)$ denotes a literal of arbitrary sign and arity. The notations $l = \sim r(u)$ and $\overline{l}$ are also used for a logical literal l equal to $r(u)$ or $\neg r(u)$.

If π is a mapping from a set of terms (or term nodes) to a set of terms (or term nodes), then for $u = (t_1, \ldots, t_k)$, $\pi(u)$ denotes the tuple $(\pi(t_1), \ldots, \pi(t_k))$. A *substitution* of variables maps every variable to a term (variable or constant) and every constant to itself. Removing a literal from a graph means removing its relation node, so some term nodes of the removed literal may become isolated. If L is a set of literals of G then $G \setminus L$ is the subgraph of G obtained from G by removing the literals in L. In a similar way, if G' is a subgraph of G then $G \setminus G'$ is the subgraph of G obtained from G by removing the literals in G'.

Definition 2 (PG homomorphism) *A PG homomorphism π from $G = (R_G,\ T_G,\ E_G,\ l_G)$ to $H = (R_H,\ T_H,\ E_H,\ l_H)$, both built on a vocabulary $\mathcal{V} = (\mathcal{R}, \mathcal{I})$, is a mapping from $R_G \cup T_G$ to $R_H \cup T_H$, such that:*

1. *for all $r \in R_G$, $\pi(r) \in R_H$; for all $t \in T_G$, $\pi(t) \in T_H$*
   (π preserves bipartition*)*

[3]A PG may have isolated term nodes, which cannot be obtained by the previous translation of a formula, but may arise for a subgraph of a PG.

2. *for all edge* $(r,\ i,\ t)$ *in G,* $(\pi(r),\ i,\ \pi(t))$ *is in* H
   (π preserves edges and their ordering*)*

3. *for all* $r \in R_G$*,* $l_H(\pi(r)) = l_G(r)$
   (π preserves relation labels*)*

4. *for all* $t \in T_G$*, if* $l_G(t) \in \mathcal{I}$ *then* $l_H(\pi(t)) = l_G(t)$*, otherwise there is no condition on* $l_H(\pi(t))$
   (π may "instantiate" variables*).*

If there is a homomorphism π from G to H, we say that G (or a subgraph of G) is *mapped* to H by π. G is called the *source* graph and H the *target* graph. Given a literal l composed of a relation node $r \in R_G$, with label $\sim p$, and list of neighbors u, $\pi(l)$ denotes the literal composed of the relation node $\pi(r)$ with list of neighbors $\pi(u)$, i.e., since π preserves relation labels, $\pi(l)$ is the literal $\sim p(\pi(u))$ in H.

Definition 3 (inconsistent PG/set of literals) *A PG (or set of literals) is said to be* inconsistent *if it contains two complementary literals* $+r(u)$ *and* $-r(u)$*. Otherwise it is said to be* consistent*.*

It can be immediately checked that inconsistent PGs correspond to unsatisfiable formulas. Positive PGs are translated into positive formulas; for this positive fragment it has been proven that PG homomorphism is sound and complete w.r.t. logical deduction, provided that the target graph is normal (basically [CM92], considering that positive PGs are a particular case of simple conceptual graphs).

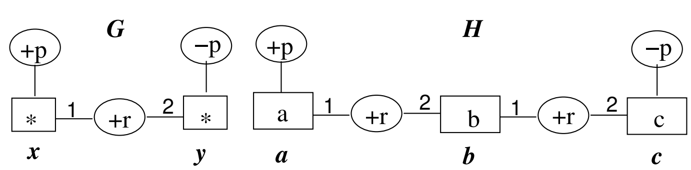


Figure 2: Non-completeness of PG homomorphism

Property 1 (Substitution / PG Homomorphism Equivalence) *Let* G *and* H *be two PGs without isolated term nodes. There is a homomorphism from* G *to* H *if and only if there is a substitution* σ *of variables in* $\Phi(G)$ *into terms in* $\Phi(H)$ *such that for each literal* $\sim p(u)$ *in* $\Phi(G)$*,* $\sim p(\sigma(u))$ *is a literal in* $\Phi(H)$*.*

For general PGs, homomorphism is still sound:

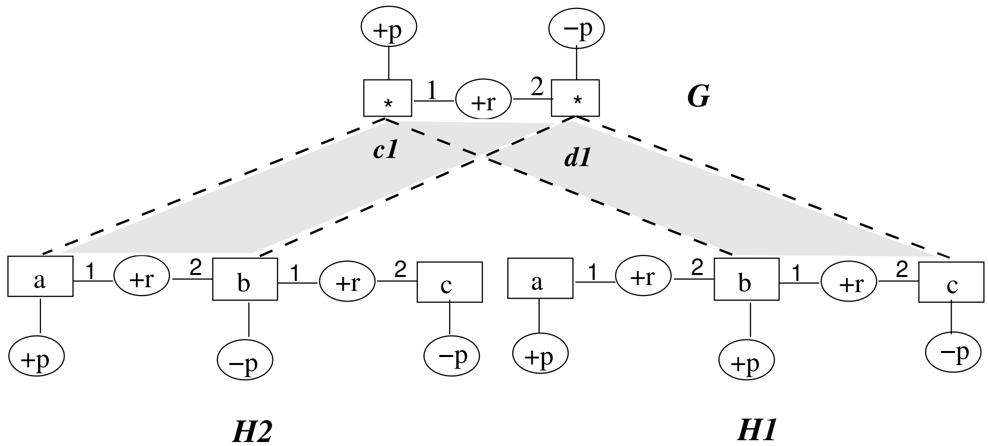


Figure 3: When the law of the excluded-middle intervenes

Property 2 *Given two PGs G and H, if there is a homomorphism from G to H then $\Phi(G)$ can be deduced from $\Phi(H)$.*

But homomorphism is no longer complete, as illustrated by Figure 2. In this figure, the formulas assigned to G and H by Φ are respectively $\Phi(G) = \exists x \exists y(p(x) \wedge \neg p(y) \wedge r(x,\ y))$ and $\Phi(H) = p(a) \wedge r(a,b) \wedge r(b,c) \wedge \neg p(c)$. $\Phi(G)$ can be deduced from $\Phi(H)$ using the tautology $p(b) \vee \neg p(b)$ (indeed, every model of $\Phi(H)$ satisfies either $p(b)$ or $\neg p(b)$; if it satisfies $p(b)$, then x and y are interpreted as b and c; in the opposite case, x and y are interpreted as a and b; thus every model of $\Phi(H)$ is a model of $\Phi(G)$). However, there is no homomorphism from G to H.

More generally, negation introduces disguised disjunctive information that cannot be taken into account by homomorphism. This disjunctive information is related to the law of the excluded-middle which holds in classical logic: given a proposition P, either P is true, or $\neg P$ is true. This leads to reasoning by cases: if a property or relation is not asserted, either it is true or its negation is true. We thus have to consider all ways of *completing* the knowledge asserted by a PG. Let us look again at the example in Figure 2. H does not say whether p holds for b. We thus have to consider two cases: either a relation node with label $+p$ or a relation node with label $-p$ can be attached to b. Let H_1 and H_2 be the graphs respectively obtained from H (see Figure 3). There is a homomorphism from G to H_1 and there is a homomorphism from G to H_2. We conclude that G can be deduced from H.

Definition 4 (Completion) *A consistent PG defined on a vocabulary $\mathcal{V} = (\mathcal{R}_\mathcal{V}, \mathcal{I}_\mathcal{V})$ is* complete *w.r.t. a set of relation names $\mathcal{R} \subseteq \mathcal{R}_\mathcal{V}$, if for each $r \in \mathcal{R}$ with arity k, for each k-tuple of not necessarily distinct term nodes $(t_1, \ldots, t_k)$, it contains*

$+r(t_1, \ldots, t_k)$ *or* $-r(t_1, \ldots, t_k)$*. If such a PG* H^c *is obtained by adding relation nodes to a PG* H*, it is called a* completion *of* H *(w.r.t.* $\mathcal{R}$*).*

If a relation node $\sim r(u)$ with $r \in \mathcal{R}$ is added to a complete PG, either this relation node is redundant or it makes the PG inconsistent. A complete PG is obtained from a consistent PG G by repeatedly adding positive and negative relation nodes as long as a relation node bringing new information and not yielding an inconsistency can be added. Since a PG is a finite graph defined over a finite set of relation names, the number of different complete PGs that can be obtained from it is finite. We can now define the deduction problem on PGs in terms of completion.

Definition 5 (PG-DEDUCTION) PG-DEDUCTION *takes two PGs* G *and* H *defined on a vocabulary* $\mathcal{V} = (\mathcal{R}_\mathcal{V}, \mathcal{I}_\mathcal{V})$ *as input, with* H *being consistent, and asks whether* G *can be PG-deduced from* H*, i.e. whether* G *can be mapped to each completion of* H *w.r.t.* $\mathcal{R}_\mathcal{V}$*.*

The following theorem expresses that PG-DEDUCTION is sound and complete with respect to the deduction in FOL.

Theorem 1 *[ML07] Let* G *and* H *be two PGs without isolated term nodes, with* H *being consistent. Then* G *can be PG-deduced from* H *if and only if* $\Phi(H) \models \Phi(G)$*.*

In the rest of the paper, we will thus not distinguish between logical deduction in the FOL$\{\exists, \wedge, \neg_a\}$ fragment and PG-deduction, and use the expression "G is deducible from H".

Let us outline a brute-force algorithm scheme for PG-DEDUCTION: all completions of H w.r.t. relation names occurring in G are generated from H, and for each of them it is checked whether G can be mapped to it. A complete graph to which G cannot be mapped can be seen as a counter-example to the assertion that G is deducible from H. Actually, not all relation names occurring in G need to be considered for completing H:

Property 3 *[LM07] The relation names that do not have both positive and negative occurrences in* G *and in* H*, are not needed in the completions of* H *(i.e.* G *is deducible from* H *if and only if* G *can be mapped to each completion of* H *w.r.t. the set of relation names that have both positive and negative occurrences in* G *and in* H*).*

From now on, completions of H are implicitly defined w.r.t. the set of relation names that have both positive and negative occurrences in G and in H, unless otherwise specified. This set of relation names will be referred to as the *completion vocabulary* w.r.t. (G, H).

3 Exchangeable literals and related properties

This section defines exchangeable literals and related notions, and provides the basic theorems underlying the complexity results in Section 4.

Two literals are said to be *opposite* if they have the same predicate and opposite polarities. Let us identify specific opposite literals in G, which likely play a role in the problem complexity, in the sense that they may lead to use the law of the excluded-middle. We say that two opposite literals of G are "exchangeable" if their arguments can have the same images by homomorphisms from G to (necessarily distinct) completions of H. More precisely:

Definition 6 (Exchangeable pair/literal w.r.t. (G, H)**)** *A pair* $\{+p(u), -p(v)\}$ *of opposite literals in G is* exchangeable *w.r.t.* (G, H) *if there are two completions of* H*, say* H_1 *and* H_2*, and two homomorphisms* π_1 *and* π_2*, respectively from G to* H_1 *and from G to* H_2*, such that* $\pi_1(u) = \pi_2(v)$*. A literal in G is* exchangeable *w.r.t.* (G, H) *if it belongs to an exchangeable pair w.r.t.* (G, H)*.*

In the following, exchangeable pairs and exchangeable literals are implicitly defined "w.r.t. (G, H)" if not otherwise specified[4].

See for instance G in Figure 2. Let us consider the pair $\{+p(x), -p(y)\}$ of opposite literals in G. This pair is exchangeable, as can be seen in Figure 3: there is a homomorphism π_1 from G to a completion H_1 of H and there is a homomorphism π_2 from G to another completion H_2 of H, such that $\pi_1(x) = \pi_2(y)$ (and is the node in H with label b).

If a pair of literals $\{l_1, l_2\}$ is exchangeable then l_1 and $\overline{l_2}$ can be unified (after a renaming of their common variables), but the reverse is not generally true because the notion of exchangeable pair takes both structures of G and H into account. See for instance Figure 4, where l_1 and $\overline{l_2}$ are unifiable, as well as l_1 and $\overline{l_3}$. $\{l_1, l_2\}$ is an exchangeable pair, which can be seen with the following two completions of H (note that the completion vocabulary is restricted to p): in one completion, say H_1, $-p(b)$ is added (and a homomorphism from G to H_1 maps l_2 to $-p(b)$; in another completion, say H_2, $+p(b)$ and $-p(d)$ are added (and a homomorphism from G to H_2 maps l_1 to $+p(b)$. It can be checked that $\{l_1, l_3\}$ is not an exchangeable pair: there are no two completions such that their argument can be mapped to the same node [5].

[4]Note that "w.r.t. H" would not be sufficient. Indeed, a subgraph G' of G may contain literals that are exchangeable w.r.t. (G', H) but not w.r.t. (G, H). In particular, the property "being without exchangeable pair of literals" is not inherited by the subgraphs.

[5]The restriction to relation names of the completion vocabulary (see Property 3) in completions of H is important; in the previous example, $\{l_1, l_3\}$ would be an exchangeable pair if the relation name r was considered in completions of H.

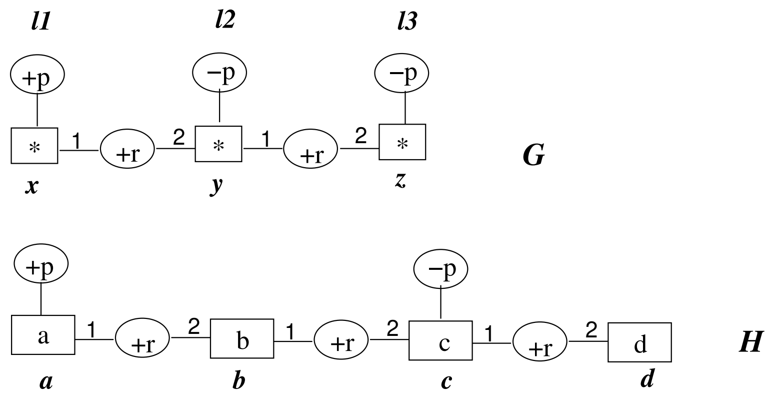


Figure 4: Exchangeable versus unifiable literals

We will now consider the subgraphs of G that do not contain any exchangeable pair w.r.t. (G, H). A subgraph of G *without exchangeable pair w.r.t.* (G, H) is a subgraph of G containing at most one literal of each exchangeable pair w.r.t. (G, H). A particular case is the *socle* of G (w.r.t. H) which contains no exchangeable literal w.r.t. (G, H) at all.

Definition 7 (Socle G_s) *Given two PGs G and H, the* socle *of G w.r.t. H, denoted G_s, is the subgraph of G obtained from G by removing all exchangeable literals.*

We recall that removing a literal means removing its relation node. Thus the socle of G contains all term nodes in G. See Figure 2: G has one exchangeable pair $\{+p(x), -p(y)\}$. The subgraphs of G without exchangeable pair are the subgraphs of G not containing $+p(x)$ *or* not containing $-p(y)$. G_s is the subgraph of G obtained by removing both relation nodes.

The following theorem is a key technical result, which underlies the main forthcoming results:

Theorem 2 *Let G and H be two PGs, with H being consistent. If G is deducible from H, then, for* each *completion H^c of H, there is a homomorphism from G to H^c that maps G_s to H.*

Proof: Assuming that G is deducible from H, let H^c be a completion of H. Let R be the set of literals l in $H^c \setminus H$ such that there is a homomorphism from G to H^c mapping a literal of G_s to l. R is consistent since it is a set of literals in H^c. Let $H^{c'}$ be the completion of H obtained from H^c by replacing every literal of R by its complementary literal, and let π be a homomorphism from G to $H^{c'}$ (such a homomorphism exists since G is deducible from H). Let us show that π is a homomorphism from G to H^c that maps G_s to H. No literal of G can be mapped by π to the complementary literal of a literal of R (otherwise this literal would be exchangeable with a literal of G_s, which contradicts the definition of G_s). Thus π is a homomorphism from G to H^c. Therefore, by definition of R, every literal of G_s is mapped by π to either H or R. However, as π is a homomorphism from G to $H^{c'}$, which contains no literal of R, no literal of G_s can be mapped to R, thus π maps G_s to H. $\square$

Let H^{c+} (resp. H^{c-}) be the positive (resp. negative) completion of H obtained by adding only positive (resp. negative) literals. As a corollary of the previous theorem, we obtain:

Property 4 *Let G and H be two PGs, with H being consistent. Let G^- (resp. G^+) be the subgraph of G defined by adding to G_s all negative (resp. positive) exchangeable literals in G. If G is deducible from H, then there is a homomorphism from G to H^{c+}, the positive completion of H (resp. to H^{c-}, the negative completion of H), that maps G^- (resp. G^+) to H.*

Proof: Let us prove the property for G^- and H^{c+} (the proof for G^+ and H^{c-} is symmetric). If G is deducible from H, Theorem 2 ensures that there is a homomorphism, say π, from G to H^{c+} that maps G_s to H. Since H^{c+} is obtained from H by adding positive literals, π maps all negative literals of G to H. Thus π maps G^- to H. $\square$

If we consider any subgraph of G without exchangeable pair (w.r.t. (G, H)), we have a weaker relationship between this subgraph and completions of H:

Theorem 3 *Let G and H be two PGs, with H being consistent. Let G' be a subgraph of G without exchangeable pair w.r.t. (G, H). If G is deducible from H, then there is a completion H^c of H and a homomorphism from G to H^c that maps G' to H.*

Proof: We suppose that G is deducible from H. Let R be the set of literals l such that there is a completion H^c of H such that l is a literal in $H^c \setminus H$ and there is a homomorphism from G to H^c mapping a literal of G' to l. R is consistent since G' contains no exchangeable pair w.r.t. (G, H). Let H^c be a completion of H containing the complementary literals of all literals of R (such a completion exists since R is consistent), and let π be a homomorphism from G to H^c (such a homomorphism exists since G is deducible from H). Let us show that π maps G' to H. By definition of R, every literal of G' is mapped by π to either H or R. However, as π is a homomorphism from G to H^c, which contains no literal of R, no literal of G' can be mapped to R, so π maps G' to H. $\square$

Theorem 3 can be rephrased as follows: if G is deducible from H, then each subgraph G' of G without exchangeable pair can be mapped to H by a homomorphism that can be extended to a homomorphism from G to a completion of H. We give the following definitions and property for this notion of extensibility.

Definition 8 (Ground subgraph of G) *A* ground *subgraph of G (w.r.t. H) is a graph obtained from G by removing some literals whose relation name belongs to the completion vocabulary (w.r.t. (G, H)).*

Note that G_s is a ground subgraph of G.

Definition 9 (Extensible homomorphism) *A homomorphism π from a ground subgraph G' of G to H is* extensible *(w.r.t. (G, H)) if it satisfies*

1. *for any literal $\sim r(u)$ in $G \setminus G'$, $\overline{\sim r}(\pi(u))$ is not in H;*
2. *for any opposite literals $+r(u)$ and $-r(v)$ in $G \setminus G'$, $\pi(u) \neq \pi(v)$.*

Note that, as G' is a ground subgraph of G, G' contains all term nodes of G, so $\pi(u)$ is defined for any literal $\sim r(u)$ in $G \setminus G'$.

Property 5 *A homomorphism π from a ground subgraph G' of G to H is extensible (w.r.t. (G, H)) if and only if it can be extended to a homomorphism from G to a completion of H.*

Proof: Let π be a homomorphism from G' to H. Conditions 1 and 2 are obviously necessary for π to be extendable to a homomorphism from G to a completion of H. Let us show that they are sufficient. We suppose that π satisfies conditions 1 and 2. Let H' be the graph obtained from H by adding the literal $\sim r(\pi(u))$ for every literal $\sim r(u)$ in $G \setminus G'$ such that $\sim r(\pi(u))$ is not already present in H. For each added literal l, the literal $\bar{l}$ is not in H by condition 1, and is not another added literal by condition 2. Thus H' is consistent. Moreover, as G' is a ground subgraph of G, the relation name of each literal in $G \setminus G'$ belongs to the completion vocabulary. It follows that H' can be completed into a completion H^c of H and that π can be extended to a homomorphism from G to H^c. □

We obtain the following corollary of Theorem 3 and Property 5.

Corollary 1 *Let G and H be two PGs, with H being consistent. Let G' be a ground subgraph of G without exchangeable pair w.r.t. (G, H). If G is deducible from H, then there is an extensible homomorphism from G' to H.*

Previous properties provide necessary deducibility conditions, and therefore sufficient non-deducibility conditions. For instance, by Corollary 1, if we find a ground subgraph of G without exchangeable pair w.r.t. (G, H) such that there is no extensible homomorphism from G' to H then we know that G is not deducible from H.

The problem of checking whether there is an extensible homomorphism from G' to H (given PGs G and H and a ground subgraph G' of G) is NP-complete. It is in NP since an extensible homomorphism from G' to H provides a polynomial certificate, and it is complete for NP since in the case where $G' = G$, it is equivalent to the NP-complete problem of checking homomorphism from G to H.

4 Main complexity Results

We now focus on the role of exchangeable literals in the problem complexity. It follows immediately from previous properties that the problem complexity falls into NP if G has no exchangeable pair (see also Section 4.2). A natural question that arises then is whether a bounded number of exchangeable pairs affects the complexity. The answer is yes, as we will show it.

To study this question, let us define the following family of problems, where k is the maximal number of exchangeable pairs in G, and is fixed for each problem.

DEDUCTION$_k$
Input: two PGs G and H, with H being consistent and G possessing (at most) k exchangeable pairs w.r.t.(G, H).
Question: Is G deducible from H?

For any integers k and k' such that $k < k'$, DEDUCTION$_{k'}$ is at least as difficult as DEDUCTION$_k$, since any graph G possessing at most k exchangeable pairs also possesses at most k' exchangeable pairs.

Please note for the following results that we make the usual assumption that the arity of predicates is bounded by a constant.

4.1 Complexity of the recognition problem

A desirable property is that recognizing exchangeable literals is not difficult compared to PG-DEDUCTION complexity, which is indeed the case:

Property 6 *Let* EXCHANGEABLE *be the problem that takes two PGs G and H as input and asks if G possesses an exchangeable pair w.r.t.* (G, H). EXCHANGEABLE *is NP-complete.*

Proof: EXCHANGEABLE is in NP: a polynomial certificate is given by a pair of two opposite literals in G, and the proof that it is exchangeable, i.e. two completions of H and two homomorphisms from G to these completions which map the literals to the "same place". For NP-completeness, a reduction is built from positive PG-HOMOMORPHISM (given two positive PGs G_1 and G_2, is there a homomorphism from G_1 to G_2 ?). Let G_1 and G_2 be two positive PGs. "Gadgets" are added to G_1 and G_2, yielding G'_1 and G'_2 respectively, such that there is a homomorphism from G_1 to G_2 if and only if G'_1 possesses an exchangeable pair w.r.t. (G'_1, G'_2). Take, for instance, the graphs G and H in Figure 2, and choose the relation names r and p such that they do not occur in G_1 and G_2. G'_1 (resp. G'_2) is obtained by making the disjoint sum[6] of G_1 and G (resp. of G_2 and H). The only candidate exchangeable pair in G'_1 is $\{+p(x), -p(y)\}$. □

The polynomial certificate used in the previous proof can be extended in a straightforward way to a polynomial certificate for the problem of deciding whether a graph possesses "at least k exchangeable pairs" (where k is fixed). It follows that this problem is NP-complete too. Thus, the problem of deciding whether a graph possesses at most k exchangeable pairs, i.e. the recognition problem associated with DEDUCTION$_k$, is co-NP-complete.

Property 7 *The problem that takes two PGs G and H as input and asks if G possesses at most k exchangeable pairs w.r.t. (G, H) is co-NP-complete.*

The complexity of the recognition problem associated with DEDUCTION$_k$ may be seen as restricting practical use of the results in this paper. However, besides the fact that recognizing exchangeable pairs may be easier in practice than in theory, most of these results can be used in a weaker form by replacing exchangeable pairs by pairs of opposite (or opposite and unifiable) literals, which can be recognized in linear time. For instance, Theorem 2 still holds if G_s is replaced by the subgraph of G obtained from G by removing all pairs of opposite and unifiable literals, since this graph is a subgraph of G_s.

4.2 DEDUCTION$_0$ and DEDUCTION$_1$

It follows from previous results that DEDUCTION$_0$ is NP-complete. We will show that DEDUCTION$_1$ is also NP-complete.

Property 8 *Let G and H be two PGs, with G having no exchangeable pair w.r.t. (G, H), and H being consistent. G is deducible from H if and only if there is a homomorphism from G to H.*

[6]The disjoint sum of two graphs A and B is the graph obtained by making the union of two disjoint copies of A and of B.

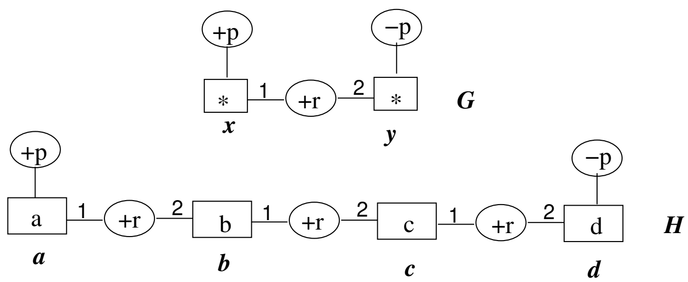


Figure 5: Illustration of Algorithm 1

Proof: If there is a homomorphism from G to H then G is deducible from H by Property 2. The converse follows from Theorem 2 since $G_s = G$ (or from Theorem 3 with $G' = G$). $\square$

Property 9 *The problem* DEDUCTION$_0$ *is NP-complete.*

It can be immediately checked that DEDUCTION$_1$ is NP-difficult: it is at least as difficult as the NP-complete problem DEDUCTION$_0$. It remains to prove that DEDUCTION$_1$ is in NP.

Let us first explain the ideas of the proof on Figure 5. G possesses one exchangeable pair $\{+p(x), -p(y)\}$. There is no homomorphism from G to H. But G can be mapped to every completion of H that contains $-p(b)$ (with x and y being respectively mapped to a and b). If a completion does not contain $-p(b)$, then it contains $+p(b)$, thus it remains to check that G is deducible from $H_1 = H + \{+p(b)\}$. The same reasoning is applied on H_1: there is no homomorphism from G to H_1, but G can be mapped to every completion of H_1 that contains $-p(c)$ (with x and y being respectively mapped to b and c); it remains to check that G is deducible from $H_2 = H_1 + \{+p(c)\}$, which is the case since there is a homomorphism from G to H_2. G can thus be seen as "sliding" on a growing H, from a place allowing to map $G \setminus \{-p(y)\}$ to a place allowing to map $G \setminus \{+p(x)\}$. Each step after the first one uses the literal added at the preceding step. We are sure that this sliding process will succeed after a finite number of steps since H cannot grow infinitely.

These ideas directly lead to Algorithm 1.

Property 10 *The algorithm* DEDUCTION$_1$ *is correct.*

Proof: We first check that the recursive call satisfies the precondition, i.e. that if there is at most one exchangeable pair w.r.t. (G, H) then there is at most one exchangeable pair w.r.t. $(G, H + \{\overline{\sim p}(\pi(u))\})$ and the precondition on $\sim p(u)$

Algorithm 1: DEDUCTION$_1$

Data: G and H two PGs; H is consistent; G possesses at most one exchangeable pair; if it has one, $\sim p(u)$ is an exchangeable literal in G otherwise $\sim p(u)$ is a literal in G such that relation name p belongs to the completion vocabulary w.r.t. (G, H).

Result: true if G is deducible from H, false otherwise

```
begin
    if there is no extensible homomorphism from G \ {~p(u)} to H then
        return false
    else
        let π be such a homomorphism
        if ~p(π(u)) is in H then
            return true
        else
            return DEDUCTION_1(G, H + {~p̄(π(u))}, ~p(u))
end
```

---

still holds. It is indeed the case, since any exchangeable pair w.r.t. $(G, H + \{\overline{\sim p}(\pi(u))\})$ is also an exchangeable pair w.r.t. (G, H), as any completion of $H + \{\overline{\sim p}(\pi(u))\}$ is also a completion of H (note that the completions of H and of $H + \{\overline{\sim p}(\pi(u))\}$ are defined w.r.t. the same set of relation names since relation name p belongs to the completion vocabulary w.r.t. (G, H)).

We also check that the number of recursive calls is finite, as the number of nodes of H is incremented at each recursive call (the added literal $\overline{\sim p}(\pi(u))$ is not already present in H since π is extensible[7]), and is bounded by the number of literals in a completion of H.

Let us show by induction on the number k of recursive calls that DEDUCTION$_1(G, H, \sim p(u))$ returns true if G is deducible from H, and false otherwise. If $k = 0$, i.e. if there is no recursive call, then either there is no extensible homomorphism from $G \setminus \{\sim p(u)\}$ to H (and then by Corollary 1 G is not deducible from H) and DEDUCTION$_1(G, H, \sim p(u))$ returns false, or $\sim p(\pi(u))$ is in H (and then π can be extended to a homomorphism from G to H, so G is deducible from H) and DEDUCTION$_1(G, H, \sim p(u))$ returns true. Thus the property is true for $k = 0$. We suppose that it is true for k recursive calls. Let us show that it is true for $k + 1$ recursive calls. As there is at least one recursive call, DEDUCTION$_1(G, H, \sim p(u))$ returns true iff DEDUCTION$_1(G, H + \{\overline{\sim p}(\pi(u))\}, \sim p(u))$ returns true, i.e., by

[7] Here, as $G \setminus G'$ is restricted to literal $\sim p(u)$, conditions 1 and 2 of extensibility are restricted to: $\overline{\sim p}(\pi(u))$ is not in H.

induction hypothesis, iff G is deducible from $H + \{\overline{\sim p}(\pi(u))\}$. It remains to show that G is deducible from H iff G is deducible from $H + \{\overline{\sim p}(\pi(u))\}$. If G is deducible from H then G is deducible from $H + \{\overline{\sim p}(\pi(u))\}$ since every completion of $H + \{\overline{\sim p}(\pi(u))\}$ is a completion of H. Conversely, we suppose that G is deducible from $H + \{\overline{\sim p}(\pi(u))\}$. As π is an extensible homomorphism from $G \setminus \{\sim p(u)\}$ to H, it can be extended to a homomorphism from G to $H + \{\sim p(\pi(u))\}$. Thus G can be mapped to every completion of $H + \{+p(\pi(u))\}$ and to every completion of $H + \{-p(\pi(u))\}$, and therefore to every completion of H (since any completion of H contains either $H + \{+p(\pi(u))\}$ or $H + \{-p(\pi(u))\}$. Hence G is deducible from H. $\square$

The following property immediately follows from Algorithm 1.

Property 11 *Let G and H be two PGs such that G has (at most) one exchangeable pair, containing literal $\sim p(u)$ and H is consistent. G is deducible from H if and only if there is a sequence $(\pi_i)_{i\in 1..m}$ such that:*

1. *π_1 is an extensible homomorphism from $G \setminus \{\sim p(u)\}$ to $H_1 = H$*
2. *$\forall i \in 2..m-1$,*
   π_i is an extensible homomorphism from $G \setminus \{\sim p(u)\}$ to $H_i = H_{i-1} + \{\overline{\sim p}(\pi_{i-1}(u))\}$
3. *π_m is a homomorphism from G to $H_m = H_{m-1} + \{\overline{\sim p}(\pi_{m-1}(u))\}$.*

We are now able to prove the NP-completeness of DEDUCTION$_1$.

Theorem 4 *The problem* DEDUCTION$_1$ *is NP-complete.*

Proof: The polynomial certificate follows directly from Property 11. Indeed, the length m of the sequence is bounded by $(n_H)^k$, where n_H is the number of term nodes in H and k is the arity of r (which is considered as bounded by a constant). $\square$

4.3 DEDUCTION$_k$

Let us now show that DEDUCTION$_k$ falls into P^{NP} for any value of parameter k. The technique used to show that Deduction$_1$ is in NP does not seem to be generalizable to $k \geq 2$. Instead, we will rely on Theorem 2. We first deduce from this theorem a necessary and sufficient deducibility condition (Property 12), which will be used in subsequent complexity proofs, and is also interesting for itself.

Let us provide an idea of this condition on examples of Figures 2 and 5. For the graphs in Figure 2, if $p(b)$ is known to be true (i.e. if literal $+p(b)$ is added to

H) then G can be deduced (i.e. G can be mapped to $H + \{+p(b)\}$), and if $p(b)$ is known to be false then G can be deduced too (i.e. G can also be mapped to $H + \{-p(b)\}$). Thus there are two extensible homomorphisms from G_s to H, which can be extended to homomorphisms from G to $H + \{+p(b)\}$ and $H + \{-p(b)\}$ respectively, with the proposition $p(b) \vee \neg p(b)$ being a tautology. Similarly, for the graphs in Figure 5, there are three extensible homomorphisms π_1, π_2 and π_3 from G_s to H mapping G_s to $+r(a,b)$, $+r(b,c)$ and $+r(c,d)$ respectively, that can be extended to homomorphisms from G to $H + \{-p(b)\}$, $H + \{+p(b), -p(c)\}$ and $H + \{+p(c)\}$ respectively, with the proposition $\neg p(b) \vee (p(b) \wedge \neg p(c)) \vee p(c)$ being a tautology. We will build from the set of extensible homomorphisms from any ground subgraph G' of G contained in G_s to H a propositional formula that is a tautology if and only if G is deducible from H.

Notations 1 *Let G and H be two PGs, with H being consistent, and let G' be a ground subgraph of G.*
P_H denotes the set of atoms of $\Phi(H^c \setminus H)$, where H^c is an arbitrary completion of H, seen as the set of atoms of a language in propositional logic.
For any extensible homomorphism π from G' to H, $L_{G'}(\pi)$ denotes the set of literals l such that $l = \sim p(\pi(u))$ for some literal $\sim p(u)$ in G and l is not in H, and $C_{G'}(\pi)$ denotes the conjunction of the literals in $L_{G'}(\pi)$ seen as a proposition on P_H.
$D_{G'}(G,H)$ denotes the disjunction of the propositions $C_{G'}(\pi)$ for all extensible homomorphisms π from G' to H.
Omission of subscript G' means that G' is equal to G_s.

For instance, in the previous example of Figure 5, with $P_H = \{p(b), p(c)\}$ and $G' = G_s$, $L(\pi_1) = \{-p(b)\}$, $L(\pi_2) = \{+p(b), -p(c)\}$, $L(\pi_3) = \{+p(c)\}$, $C(\pi_1) = \neg p(b)$, $C(\pi_2) = p(b) \wedge \neg p(c)$, $C(\pi_3) = p(c)$, $D(G,H) = \neg p(b) \vee (p(b) \wedge \neg p(c)) \vee p(c)$.
$L_{G'}(\pi)$ is the set of literals "missing" in H for π to be extendable to a homomorphism from G to H, and therefore it is the set of literals that have to be in any completion H^c of H such that π can be extended to a homomorphism from G to H^c. This is stated in following Lemma 1.

Lemma 1 *Let G and H be two PGs, let H^c be a completion of H, let G' be a ground subgraph of G, and let π be an extensible homomorphism from G' to H. π can be extended to a homomorphism from G to H^c if and only if $L_{G'}(\pi)$ is a set of literals in H^c.*

Lemma 2 expresses the straightforward correspondence between the completions of H and the truth assignments on P_H.

Lemma 2 *There is a bijection f from the set of completions of H to the set of truth assignments on P_H such that for any completion H^c of H, any ground subgraph G' of G and any extensible homomorphism π from G' to H, $L_{G'}(\pi)$ is a set of literals in H^c if and only if $f(H^c)$ satisfies $C_{G'}(\pi)$.*

Proof: Let f be the mapping from the set of completions of H to the set of truth assignments on P_H defined by: for every completion H^c of H, $f(H^c)$ assigns the value true to an atom $p(u)$ in P_H if $+p(u)$ is a literal in H^c, and false otherwise (i.e. if $-p(u)$ is a literal in H^c). f clearly satisfies the desired conditions. □

Property 12 *Let G and H be two PGs, with H being consistent, and let G' be a ground subgraph of G contained in G_s. G is deducible from H if and only if $D_{G'}(G, H)$ is a tautology.*

Proof: By Theorem 2 (since G' is contained in G_s) and Property 5 (since G' is a ground subgraph of G), G is deducible from H iff for each completion H^c of H, there is an extensible homomorphism from G' to H that can be extended to a homomorphism from G to H^c. By Lemmas 1 and 2, the latter proposition can be rephrased as: for each truth assignment v on P_H, there is an extensible homomorphism π from G' to H such that v satisfies $C_{G'}(\pi)$, i.e. $D_{G'}(G, H)$ is a tautology.
□

In order to prove that DEDUCTION$_k$ is in P^{NP}, we show how to compute $D(G, H)$ without explicitly computing all extensible homomorphisms from G_s to H, whose number may be exponential in the size of G. Let $\mathcal{E}$ be the set of exchangeable literals, and $\mathcal{T}_\mathcal{E}$ be the set of term nodes occurring in $\mathcal{E}$. The main idea is that, for any extensible homomorphism from G_s to H, the set $L(\pi)$, and therefore proposition $C(\pi)$, only depend on the restriction of π to $\mathcal{T}_\mathcal{E}$. Thus, we can define $L(\varphi)$ and $C(\varphi)$ for any mapping φ from $\mathcal{T}_\mathcal{E}$ to the set T_H of term nodes in H, and $D(G, H)$ is the disjunction of the propositions $C(\varphi)$ for every mapping φ from $\mathcal{T}_\mathcal{E}$ to T_H that can be extended[8] to an extensible homomorphism from G_s to H. Algorithm 2 computes $D(G, H)$ to determine whether G is deducible from H, using Property 12.

If the number of exchangeable pairs is bounded by a constant k, then the number of mappings from $\mathcal{T}_\mathcal{E}$ to the set of term nodes in H becomes polynomial, which makes DEDUCTION$_k$ fall into P^{NP}.

Theorem 5 *For any integer $k \geq 0$, the problem* DEDUCTION$_k$ *is in P^{NP}.*

[8] A mapping φ from $\mathcal{T}_\mathcal{E}$ to T_H can be extended to an extensible homomorphism from G_s to H iff it satisfies both following independent conditions: 1) φ can be extended to a homomorphism, say π, from G_s to H and 2) φ satisfies conditions 1 and 2 of extensibility, which only depend on the restriction of π to $\mathcal{T}_\mathcal{E}$, i.e. on φ itself.

Algorithm 2: Deduction$_k(G, H)$

Data: G and H two PGs, such that H is consistent
Result: true if G is deducible from H, false otherwise
begin
  Let $\mathcal{E}$ be the set of exchangeable literals w.r.t. (G, H)
  Let $\mathcal{T}_\mathcal{E}$ be the set of term nodes occurring in $\mathcal{E}$
  Let $G_s = G \setminus \mathcal{E}$
  $\Phi \leftarrow false$
  for *every mapping* φ *from* $\mathcal{T}_\mathcal{E}$ *to the set of term nodes in* H **do**
    if φ *can be extended to an extensible homomorphism from* G_s *to* H **then**
      $\Phi \leftarrow \Phi \vee C(\varphi)$
  return *Tautology*(Φ)
end

Proof: It is sufficient to show that if the number of exchangeable pairs is bounded by k then Algorithm 2 can be executed in polynomial time with a polynomial number of calls to a NP oracle. This is indeed the case since:
- to compute $\mathcal{E}$, it is sufficient to determine for each pair of opposite literals of G (whose number is polynomial) if it is exchangeable, which is in NP,
- $|\mathcal{T}_\mathcal{E}| \leq 2kr$, where r is the maximal arity of a relation name, so the number of mappings from $\mathcal{T}_\mathcal{E}$ to the set of term nodes in H is bounded by n_H^{2kr}, and therefore is polynomial,
- determining if such a mapping φ can be extended to an extensible homomorphism from G_s to H is in NP (such an extension provides a polynomial certificate),
- determining if a proposition is not a tautology is in NP. □

4.4 DEDUCTION$_3$

Let us now prove that DEDUCTION$_k$ is co-NP-difficult for any $k \geq 3$. As it is also NP-difficult, it is not likely in NP nor in co-NP.

Theorem 6 *The problem* DEDUCTION$_3$ *is co-NP-difficult.*

Proof: To prove that DEDUCTION$_3$ is co-NP-difficult, we define a reduction from the co-NP-complete problem 3-DNF Tautology to DEDUCTION$_3$.
3-DNF Tautology
Input: a 3-DNF propositional formula Φ, i.e. a proposition Φ in disjunctive normal form (disjunction of conjunctions of literals) such that each conjunction in Φ has

at most 3 literals.
Question: Is Φ a tautology?

The reduction uses Property 12. Let Φ be a 3-DNF proposition. By Property 12, it is sufficient to build two PGs G and H in polynomial time, with H consistent and with at most 3 exchangeable pairs, such that for some ground subgraph G' of G contained in G_s, $D_{G'}(G,H)$ is a tautology iff Φ also is.
It is rather easy to build such PGs G and H with at most 9 exchangeable pairs. To ensure that they have at most 3 exchangeable pairs, we have to refine the construction. For this, we introduce the notion of *correct* mapping w.r.t. Φ.
Let P be the set of atoms in Φ. A mapping α from P to $\{1,2,3\}$ is said to be *correct* (w.r.t. Φ) if for any conjunction C in Φ and any positive literals p and p' (resp. negative literals $\neg p$ and $\neg p'$) in C, $\alpha(p) \neq \alpha(p')$.
For instance, if $\Phi = (\neg p \wedge \neg s) \vee (s \wedge \neg q \wedge \neg r) \vee (p \wedge q \wedge r)$ then the mapping $\alpha = \{(p,1),(q,2),(r,3),(s,2)\}$ is correct. Note that there may be no correct mapping w.r.t. a given Φ. For instance, if $\Phi = (p \wedge q \wedge r) \vee (p \wedge q \wedge s) \vee (r \wedge s)$ then a correct mapping α should satisfy $\alpha(r) = \alpha(s)$ from the two first conjunctions, and $\alpha(r) \neq \alpha(s)$ from the third conjunction.
In the first step of the proof, we will describe how to build in polynomial time from a 3-DNF proposition Φ both a 3-DNF proposition Φ', such that Φ' is a tautology iff Φ is, and a correct mapping α w.r.t. Φ' (which will necessarily exist). In the second step, we will describe how to build PGs G and H with at most 3 exchangeable pairs from a 3-DNF Φ and a correct mapping w.r.t. Φ, such that for some ground subgraph G' of G contained in G_s, $D_{G'}(G,H)$ is a tautology iff Φ is.

1. Construction of Φ' and α
For each atom p in P, let h be the number of occurrences of p in Φ, these h occurrences are replaced by h new atoms p_1, p_2, ..., p_h, and the 3-DNF formula $NEQ(p_1,\ldots,p_h) = (p_1 \wedge \neg p_2) \vee (p_2 \wedge \neg p_3) \vee \ldots \vee (p_{h-1} \wedge \neg p_h) \vee (p_h \wedge \neg p_1)$ is added to the disjunction. Φ' is the obtained formula. For instance, if $\Phi = (\neg p \wedge \neg s) \vee (s \wedge \neg q \wedge \neg r) \vee (p \wedge q \wedge r)$ then $\Phi' = (\neg p_1 \wedge \neg s_1) \vee (s_2 \wedge \neg q_1 \wedge \neg r_1) \vee (p_2 \wedge q_2 \wedge r_2) \vee NEQ(p_1,p_2) \vee NEQ(q_1,q_2) \vee NEQ(r_1,r_2) \vee NEQ(s_1,s_2)$.
Note that a truth assignment satisfies $NEQ(p_1,\ldots,p_h)$ iff it does not assign the same truth value to $p_1,\ldots,p_h$. It follows that Φ' is a tautology iff it is satisfied by each truth assignment assigning the same truth value to $p_1,\ldots,p_h$. Thus Φ' is a tautology iff Φ is.
A correct mapping α w.r.t. Φ' is built as follows: for each conjunction in Φ' coming from a conjunction in Φ (considered independently from the others), atoms of positive (resp. negative) literals are mapped to consecutive integers starting from 1; α is the union of the mappings obtained for these conjunctions. For instance,

if $\Phi' = (\neg p_1 \wedge \neg s_1) \vee (s_2 \wedge \neg q_1 \wedge \neg r_1) \vee (p_2 \wedge q_2 \wedge r_2) \vee NEQ(p_1, p_2) \vee NEQ(q_1, q_2) \vee NEQ(r_1, r_2) \vee NEQ(s_1, s_2)$ then we independently define $\alpha_1 = \{(p_1, 1), (s_1, 2)\}$, $\alpha_2 = \{(s_2, 1), (q_1, 1), (r_1, 2)\}$ and $\alpha_3 = \{(p_2, 1), (q_2, 2), (r_2, 3)\}$, and $\alpha = \alpha_1 \cup \alpha_2 \cup \alpha_3$. It is easy to check that Φ' and α can be computed in polynomial time and that α is correct w.r.t. Φ'.

2. Construction of G and H

Let Φ be a 3-DNF formula and α be a correct mapping w.r.t. Φ. PGs G and H are defined as follows (see Figure 6 for an illustration).

G is independent from Φ and α. It has 6 variable nodes x_1, x_2, x_3, y_1, y_2 and y_3, and 7 literals: $+r(x_1, x_2, x_3, y_1, y_2, y_3)$ and, for all i in 1..3, $+p(x_i)$ and $-p(y_i)$.

H depends from Φ and α. Let $p_1, \ldots, p_h$ be the atoms in Φ, and let $C_1, \ldots, C_q$ be the conjunctions in Φ. H has $h + 2$ constant nodes labeled with $a_1, \ldots, a_h$, c and d, and it has $q + 2$ literals: $+p(c)$, $-p(d)$ and, for all i in $1..q$, $+r(u_i)$, with $u_i = (s_{i,1}, s_{i,2}, s_{i,3}, t_{i,1}, t_{i,2}, t_{i,3})$ being defined as follows. For all i in $1..q$ and all j in 1..3:

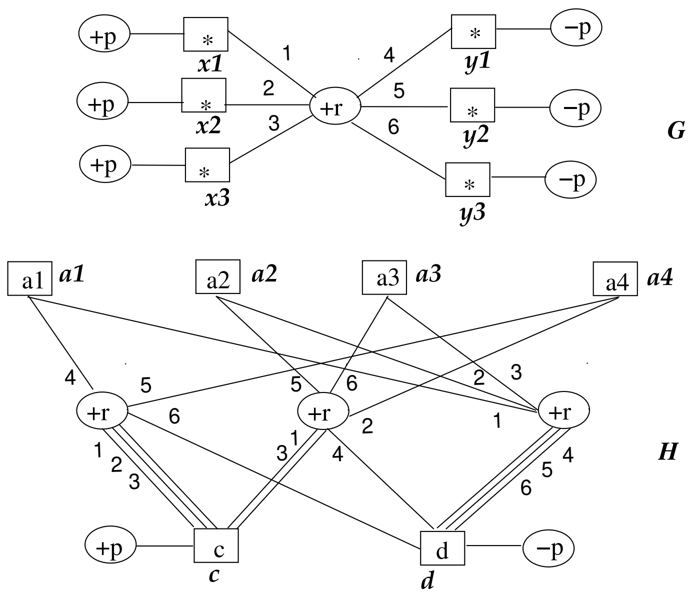
- if $j = \alpha(p_k)$ for some positive literal p_k in C_i (there is at most one such literal p_k since α is correct) then $s_{i,j} = a_k$ else $s_{i,j} = c$,

- if $j = \alpha(p_k)$ for some negative literal $\neg p_k$ in C_i (there is at most one such literal $\neg p_k$ since α is correct) then $t_{i,j} = a_k$ else $t_{i,j} = d$.

For instance, consider the formula of the previous example $(\neg p \wedge \neg s) \vee (s \wedge \neg q \wedge \neg r) \vee (p \wedge q \wedge r)$. Let us rename p, q, r and s into p_1, p_2, p_3 and p_4 respectively. We obtain $\Phi = (\neg p_1 \wedge \neg p_4) \vee (p_4 \wedge \neg p_2 \wedge \neg p_3) \vee (p_1 \wedge p_2 \wedge p_3)$. Let $\alpha = \{(p_1, 1), (p_2, 2), (p_3, 3), (p_4, 2)\}$. Then the literals of H labeled with $+r$ are $+r(c, c, c, a_1, a_4, d)$, $+r(c, a_4, c, d, a_2, a_3)$ and $+r(a_1, a_2, a_3, d, d, d)$, as pictured in Figure 6.

G and H can be constructed in polynomial time. The completion vocabulary is restricted to $\{p\}$. Let G' be the subgraph of G restricted to its literal $+r(x_1, x_2, x_3, y_1, y_2, y_3)$. G' is a ground subgraph of G contained in G_s. It is easy to check that $D_{G'}(G, H)$ is obtained from Φ by replacing each atom p_i by atom $p(a_i)$. For instance, in the example of Figure 6, there are 3 extensible homomorphisms from G' to H, and $D_{G'}(G, H) = (\neg p(a_1) \wedge \neg p(a_4)) \vee (p(a_4) \wedge \neg p(a_2) \wedge \neg p(a_3)) \vee (p(a_1) \wedge p(a_2) \wedge p(a_3))$. Thus $D_{G'}(G, H)$ is a tautology iff Φ is.

It remains to show that there are at most 3 exchangeable pairs w.r.t. (G, H). There are 9 pairs of opposite literals in G, namely the pairs $\{+p(x_i), -p(y_j)\}$ for i, j in 1..3. However, if x_i and y_j are mapped to the same node w in H by two homomorphisms from G to completions of H, then there is an integer k in $1..h$ such that w is labeled a_k, with $i = j = \alpha(p_k)$. Thus, each exchangeable pair is in the form $\{+p(x_i), -p(y_i)\}$, with i in 1..3. As announced at the beginning of this proof, using a correct mapping w.r.t. Φ to define H allows to bound the number of



$\Phi = (\neg p_1 \wedge \neg p_4) \vee (p_4 \wedge \neg p_2 \wedge \neg p_3) \vee (p_1 \wedge p_2 \wedge p_3)$
$\alpha = \{(p_1, 1), (p_2, 2), (p_3, 3), (p_4, 2)\}$

Figure 6: Reduction from 3-DNF Tautology to DEDUCTION$_3$

exchangeable pairs to 3 instead of 9. □

4.5 When homomorphism checking is polynomial

Homomorphism checking becomes polynomial in the particular case where G is decomposable into a tree, for instance if G is a graph with treewidth less than a fixed integer k (and in this case it corresponds to a formula of the k-variables fragment of FOL [KV00]); if G is seen as a hypergraph, with relation nodes becoming hyperedges, another polynomial case is obtained if G is a hypergraph with hypertreewidth at most a fixed integer k (and in this case it corresponds to a formula of the k-guarded fragment of FOL) [GLS01]. These particular cases are specially relevant in a query answering context, where G represents a query and H represents a knowledge base composed of a set of facts. Indeed, one may reasonably assume that the query has a simple structure with respect to that of the base.

Interestingly, our previous proofs allow us to completely classify the complexity of DEDUCTION and DEDUCTION$_k$ in the above special cases (except for $k = 2$ for which the complexity in the general case is unknown):

Theorem 7 *When G has a special structure that makes homomorphism checking polynomial, the following complexity results hold:*

- DEDUCTION *is co-NP-complete*
- DEDUCTION$_0$ *and* DEDUCTION$_1$ *are in P*
- DEDUCTION$_k$ *is co-NP-complete for any $k \geq 3$.*

Proof: DEDUCTION is in co-NP since a completion H^c of H to which G cannot be mapped is a polynomial certificate of the complementary problem, NON-DEDUCTION (the size of H^c is polynomial in the size of H and the absence of homomorphism from G to H^c can be checked in polynomial time by hypothesis). DEDUCTION is complete for this complexity class because the proof of Theorem 6 shows that DEDUCTION$_3$ remains co-NP-difficult when homomorphism checking from G to any graph is polynomial (in the reduction, the graph G built is a tree). Hence, DEDUCTION$_k$ is also co-NP-complete for any $k \geq 3$. That DEDUCTION$_0$ and DEDUCTION$_1$ are in P follows immediately from Property 8 and Algorithm 1 respectively. □

4.6 Pieces

We will now take advantage of some simple graph properties to extend previous results. First note that G is deducible from H if and only if each connected com-

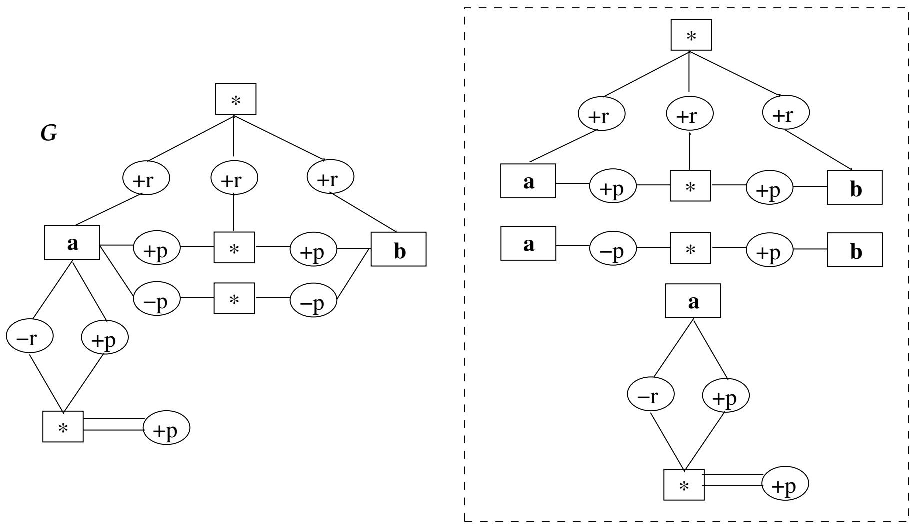


Figure 7: Pieces

ponent of G is deducible from H. Secondly, by splitting[9] constant nodes in G into several nodes (in this case G is no longer normal), we do not change the logical semantics of G and we preserve the existence or not of a homomorphism from G to any normal graph.

Let us define particular subgraphs that we call the *pieces* of G w.r.t. its constant nodes. Let $\cong$ be the following equivalence relation: given r and s two relation nodes in G, $r \cong s$ if there is a path in G between r and s that does not go through a constant node, i.e. a path $x_0(= r) \ldots x_k(= s)$ such that, for $0 < i < k$, x_i is not a constant node. The pieces of G are the subgraphs composed of the literals whose relation nodes are in the same equivalence class for $\cong$. This definition is extended to isolated term nodes by considering that each isolated node form its own piece. See Figure 7, which shows a PG on the left and its pieces on the right. The pieces of G can be computed in linear time by a traversal of G.

Property 13 *Let G and H be two PGs, with H being consistent. G is deducible from H if and only if each piece of G is deducible from H.*

The constant nodes in pieces of G can themselves be further split without any impact on the existence of a homomorphism from G to H. Some cycles in pieces

[9] Splitting a term node x into n nodes, according to a partition $\{E_1, \ldots, E_n\}$ of the edges incident to x, consists of deleting x, creating n term nodes $x_1, \ldots, x_n$ with the same label as x, and attaching to each x_i the edges in E_i, i.e. for each edge (x, j, r) in E_i, an edge (x_i, j, r) is created.

can thus be broken. Homomorphism checking becomes polynomial in the particular case where all pieces of G can be split to yield a graph decomposable into a tree (cf. Section 4.5).

See for instance Figure 7: G has 9 pairs of opposite literals, which may yield 9 pairs of exchangeable literals (depending on H and on edge labels in G, that are omitted in Figure 7); each piece of G has no opposite literals, *a fortiori* no exchangeable literals, thus to check whether G can be deduced from H, one just has to check if each piece of G can be mapped to H. Furthermore, each piece of G can be transformed into a logically equivalent tree by splitting constant nodes, thus this instance of the Deduction problem belongs to the polynomial cases.

In all previous complexity results, k can be seen as representing the maximum number of exchangeable pairs in a piece of G instead of in G.

5 Logical approach through resolution trees

In this section, we follow a logical approach and prove again fundamental results of this paper, namely Theorems 2 and 3 and Property 12, using the resolution method in propositional logic. These new proofs will be used in Section 6.1 to show that these results (hence the complexity results built on them) still hold when a preorder on predicates is considered. Besides this use, the new proofs are interesting in themselves, because they establish links between the resolution method, which is one of the main proof method in logics, and our method based on homomorphism and completions. The notion of a PG-resolution tree is defined, which allows to clarify these links. In particular, all logical literals used in a resolution tree "come from" exchangeable literals in G (see property 16 and its proof for details).

Let G and H be two PGs,with H being consistent. By Theorem 1, G can be deduced from H if and only if $\Phi(H) \vDash \Phi(G)$, or equivalently, $\Phi(H) \wedge \neg\Phi(G)$ is unsatisfiable. By Herbrand Theorem, $\Phi(H) \wedge \neg\Phi(G)$ is unsatisfiable if and only if the set F of propositional formulas defined as follows is unsatisfiable. The set of atoms of the propositional language on which F is defined is the set of atoms $p(u)$ where p is a relation name in $\mathcal{R}$ and u is a tuple of terms that are terms in $\Phi(H)$. F is the set of clauses (disjunctions of literals) equal to $C_H \cup C_G$, where C_H is the set of clauses $\Phi(H)$ (each clause is restricted to a literal) and C_G is the set of all clauses in the form $\sigma(c(G))$ where $c(G)$ is the disjunction of the complementary literals of the literals in $\Phi(G)$ and σ is a substitution of the variables of $c(G)$ by terms of $\Phi(H)$. As usually done, we represent a clause by the set of its literals. For instance, if G and H are the PGs shown in Figure 5, $C_H = \{\{p(a)\}, \{r(a, b)\}, \{r(b, c)\}, \{r(c, d)\}, \{\neg p(d)\}\}$, $c(G) = \{\neg p(x), \neg r(x, y), p(y)\}$ and C_G is the set of all clauses obtained from

$c(G)$ by replacing x and y by elements of $\{a, b, c, d\}$. We recall a classical property of unsatisfiable sets of propositional clauses.

Definition 10 (Resolution tree) *A resolution tree of a set F of propositional clauses is an anti-rooted binary tree T (each internal node of T has exactly two parents in T) labeled with propositional clauses such that the anti-root of T is labeled with the empty clause, each leaf of T is labeled with a clause of F and for each internal node y of T whose parents in T have respective labels c_1 and c_2, there is a literal l in c_1 such that $\bar{l}$ is a literal in c_2 and $label(y) = Res(c_1, c_2, l) = (c_1 \setminus \{l\}) \cup (c_2 \setminus \{\bar{l}\})$.*

Property 14 (Resolution) *A set of propositional clauses is unsatisfiable if and only if it has a resolution tree.*

In the following we suppose that G is deducible from H and we consider a resolution tree of $F = C_G \cup C_H$. Note that if c_2 is restricted to $\{\bar{l}\}$ then $Res(c_1, c_2, l) = c_1 \setminus \{l\}$. Thus, clauses of C_H allow to eliminate literals from clauses not belonging to C_H (since H is consistent) without adding any literal. We may assume w.l.o.g. that resolution operations involving clauses of C_H are performed first, hence there is a resolution tree whose leaves are labeled with clauses obtained from clauses of C_G by removing some literals l such that $\{\bar{l}\}$ is a clause of C_H. Moreover, we may assume that *all* literals l such that $\{\bar{l}\}$ is a clause of C_H are removed from these clauses, since removing some literals from some clauses of an unsatisfiable set of clauses preserves its unsatisfiability, and therefore preserves the existence of a resolution tree of this set of clauses[10]. We may also assume that none of these clauses contains complementary literals since such a clause is a tautology and removing a tautology from an unsatisfiable set of formulas preserves its unsatisfiability. We obtain a resolution tree whose leaves are labeled with clauses obtained from clauses of C_G by removing all literals complementary to literals of clauses of C_H, and containing no complementary literals. For instance, if G and H are the PGs shown in Figure 5, such a resolution tree is given in Figure 8. Note that the complementary tree of a resolution tree T, i.e. the tree obtained from T by replacing in each label each literal l by $\bar{l}$, is still a resolution tree. The labels of this resolution tree contain literals of $\Phi(G)$ instead of their negation. Moreover, in order to come back to the PG point of view, we replace in labels of the obtained resolution tree logical literals by PG literals.

Definition 11 (PG-resolution tree) *A* PG-resolution tree *is a tree T whose nodes are labeled with sets of PG literals and such that the tree obtained from T by*

[10]Please note that a clause that becomes empty is kept in the set.

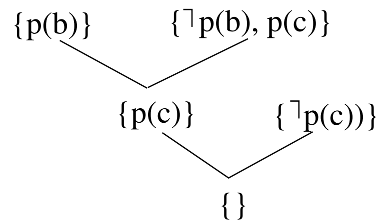


Figure 8: A resolution tree

replacing in each label each PG literal $+p(u)$ (resp. $-p(u)$) by the logical literal $p(u)$ (resp. $\neg p(u)$) is a resolution tree. Resolution operation $Res(c_1, c_2, l)$ is renamed into PG-resolution operation $PG\text{-}Res(c_1, c_2, l)$.

For instance, the PG-resolution tree obtained from the complementary tree of the resolution tree shown in Figure 8 is given in Figure 9 The anti-root of a PG-

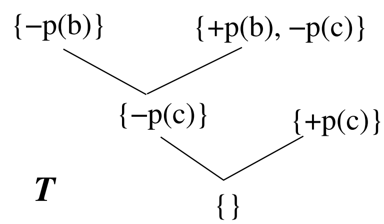


Figure 9: A PG-resolution tree

resolution tree T is denoted by $ar(T)$, and the set of its leaves is denoted by $L(T)$. We will use the following property of (PG-)resolution trees.

Lemma 3 *In any PG-resolution tree T, for any literal l in the label of a node of $L(T)$, $\bar{l}$ is also a literal in the label of a node of $L(T)$.*

Proof: Let x be a node of $L(T)$ and let l be a literal in $label(x)$. Let μ be the path in T from x to $ar(T)$, let y be the first node of μ from x such that l is not a literal in $label(y)$, and let $c = label(y)$ (y exists since $ar(T)$ is labeled with the empty clause, and $y \neq x$ since l is a literal in $label(x)$). Let y_1 and y_2 be the parents of y in T, with y_1 on μ, labeled with c_1 and c_2 respectively. As l is a literal in c_1 but not in c, $c = PG\text{-}Res(c_1, c_2, l)$ and $\bar{l}$ is a literal in c_2, and therefore in the label of some node of $L(T)$. $\square$

In the same way as we identify term nodes of a PG G with the terms associated with this node in $\Phi(G)$, we identify for each clause $\sigma(c(G))$ of C_G the substitution

σ of variables of $\Phi(G)$ by terms of $\Phi(H)$ with a mapping from the set T_G of term nodes of G to the set T_H of term nodes of H (mapping each constant node of G to the constant node of H having the same label). Thus, we associate with each node x of $L(T)$ a mapping π_x from T_G to T_H such that $label(x)$ can be defined from π_x as follows.

Definition 12 (PG-resolution tree of (G, H)) *Let G and H be two PGs. A* PG-resolution tree of (G, H) *is a structure $(T, (\pi_x)_{x \in L(T)})$ where T is a PG-resolution tree such that for each node x of $L(T)$, $label(x)$ is consistent and π_x is a mapping from T_G to T_H such that $label(x) = \{\sim p(\pi_x(u)) \mid \ \sim p(u)$ is a literal in G and $\sim p(\pi_x(u))$ is not a literal in $H\}$.*

For instance the tree T shown in Figure 9 is a PG-resolution tree of (G, H), where G and H are the PGs shown in Figure 5: if z is the node of $L(T)$ labeled with $\{-p(b)\}$ (resp. $\{+p(b), -p(c)\}$, $\{+p(c)\}$) then π_z is the mapping from T_G to T_H mapping term nodes x and y to a and b (resp. b and c, c and d).

Property 15 (PG-resolution) *Let G and H be two PGs, with H being consistent. G is deducible from H if and only if there is a PG-resolution tree of (G, H).*

Proof: This follows from the discussion above: G is deducible from H iff the set F equal to $C_G \cup C_H$ is unsatisfiable; by Property 14, F is unsatisfiable iff it has a resolution tree, and there is a resolution tree of F if and only if there is a PG-resolution tree of (G, H). □

Property 16 *Let G and H be two PGs, and let $(T, (\pi_x)_{x \in L(T)})$ be a PG-resolution tree of (G, H). For any node x of $L(T)$, π_x can be extended to a homomorphism from G to a completion of H, and any such homomorphism maps G_s to H.*

Proof: Let x be a node of $L(T)$. Let us show that π_x can be extended to a homomorphism from G to a completion of H, i.e that there is a completion H^c of H such that for any literal $\sim p(u)$ in G such that $\sim p(\pi_x(u))$ is not a literal in H, $\sim p(\pi_x(u))$ is a literal in H^c. This is still equivalent to: there is a completion H^c of H such that $label(x)$ is a set of literals in H^c. To prove this, it is is sufficient to show that the following propositions a) and b) hold:
a) $H + label(x)$ is consistent,
b) each relation name in $label(x)$ is in the completion vocabulary w.r.t. (G, H).
Let us show Proposition a). As H and $label(x)$ are consistent, it is sufficient to show that for any literal l in $label(x)$, $\bar{l}$ is not a literal in H. Let l be a literal in $label(x)$. By Lemma 3, $\bar{l}$ is in the label of a node of $L(T)$, and therefore is not a literal in H.
Let us show Proposition b). Let p be a relation name in $label(x)$. Let us show

that $+p$ and $-p$ have occurrences in G and in H. By Lemma 3, $+p$ and $-p$ have occurrences in labels of nodes of $L(T)$, and therefore in G. Since by Property 15 G is deducible from H, every node of G labeled with $+p$ (resp. $-p$) is mapped by a homomorphism from G to H^{c-} (resp. H^{c+}) to a node of H labeled with $+p$ (resp. $-p$). Hence Proposition b) holds, which completes the proof that π_x can be extended to a homomorphism from G to a completion of H.

Let π'_x be a homomorphism from G to a completion of H extending π_x. Let us show that π'_x maps G_s to H, i.e that each literal $\sim p(u)$ in G such that $\sim p(\pi'_x(u))$ is not a literal in H is exchangeable. Let $\sim p(u)$ be a literal in G such that $\sim p(\pi'_x(u))$ is not a literal in H. Then $\sim p(\pi'_x(u))$ is a literal in $label(x)$. By Lemma 3, there is a node y of $L(T)$ such that $\overline{\sim p}(\pi'_x(u))$ is a literal in $label(y)$. So there is a literal $\overline{\sim p}(v)$ in G and a homomorphism π'_y (extending π_y) from G to a completion of H such that $\pi'_x(u) = \pi'_y(v)$. It follows that $\{\sim p(u), \overline{\sim p}(v)\}$ is an exchangeable pair, hence $\sim p(u)$ is exchangeable. □

We are now ready to give new proofs of Theorems 2 and 3.

Lemma 4 *Let G and H be two PGs. If there is a PG-resolution tree of (G, H) then, for* each *completion H^c of H, there is a homomorphism from G to H^c that maps G_s to H.*

Proof: We suppose that there is a PG-resolution tree $(T, (\pi_x)_{x\in L(T)})$ of (G, H). Let H^c be a completion of H. Let us show that there is a homomorphism from G to H^c that maps G_s to H. By Property 16, it is sufficient to show that there is a node x of $L(T)$ such that π_x can be extended to a homomorphism from G to H^c, i.e. such that $label(x)$ is a set of literals in H^c. For any node y of T, let $P(y)$ denote the property:

$P(y)$: $label(y)$ is a set of literals in H^c.

In order to prove that there is a node x of $L(T)$ such that $P(x)$ holds, we build a path μ from $ar(T)$ to a leaf x of T, $\mu = (ar(T) = y_0, y_1, ..., y_p = x)$ such that for each i from 0 to p, $P(y_i)$ holds. We define y_i and prove $P(y_i)$ by induction on i. For $i = 0$, $y_0 = ar(T)$ and $P(y_0)$ trivially holds. We suppose that $(ar(T) = y_0, y_1, ..., y_i)$ is a path in T from $ar(T)$ towards a leaf of T such that $P(y_i)$ holds and y_i is not a leaf of T. Let z and z' be the parents of y_i labeled with c and c' resp., and $+p(w)$ such that $label(y_i) = PG\text{-}Res(c, c', +p(w))$. Thus $label(z) \subseteq label(y_i) \cup \{+p(w)\}$ and $label(z') \subseteq label(y_i) \cup \{-p(w)\}$. Moreover, either $+p(w)$ or $-p(w)$ is a literal in H^c since by Property 16, $+p(w)$ is a literal in $H' \setminus H$ for some completion H' of H. We define y_{i+1} as z if $+p(w)$ is a literal in H^c, and z' otherwise. It follows from $P(y_i)$ and the definition of y_{i+1} that $P(y_{i+1})$ also holds. Hence there is a node x of $L(T)$ such that $P(x)$ holds. □

Theorem 2 *Let G and H be two PGs, with H being consistent. If G is deducible from H, then, for* each *completion H^c of H, there is a homomorphism from G to H^c that maps G_s to H.*

Proof: This follows immediately from Property 15 and Lemma 4. □

Lemma 5 *Let G and H be two PGs. Let G' be a subgraph of G without exchangeable pair w.r.t. (G, H). If there is a PG-resolution tree of (G, H) then there are a completion H^c of H and a homomorphism from G to H^c that maps G' to H.*

Proof: We suppose that there is a PG-resolution tree $(T, (\pi_x)_{x \in L(T)})$ of (G, H). Let us show that there is a completion H^c of H and a homomorphism from G to H^c that maps G' to H. By Property 16, it is sufficient to show that there is a node x of $L(T)$ such that for any literal $\sim p(u)$ in G', $\sim p(\pi_x(u))$ is a literal in H (since in that case any homomorphism from G to a completion of H extending π_x maps G' to H), i.e. such that for any literal $\sim p(u)$ in G', $\sim p(\pi_x(u))$ is not a literal in $label(x)$. For this, it is sufficient to show that there is a node x of $L(T)$ such that a stronger property $P(x)$ holds, where $P(y)$ is defined for any node y of T by:
$P(y)$: for any literal $\sim p(u)$ in G' and any homomorphism π from G to a completion of H, $\sim p(\pi(u))$ is not a literal in $label(y)$.
In order to prove that there is a node x of $L(T)$ such that $P(x)$ holds, we build a path μ from $ar(T)$ to a leaf x of T, $\mu = (ar(T) = y_0, y_1, ..., y_p = x)$ such that for each i from 0 to p, $P(y_i)$ holds. We define y_i and prove $P(y_i)$ by induction on i. For $i = 0$, $y_0 = ar(T)$ and $P(y_0)$ trivially holds. We suppose that $(ar(T) = y_0, y_1, ..., y_i)$ is a path in T from $ar(T)$ towards a leaf of T such that $P(y_i)$ holds and y_i is not a leaf of T. Let z and z' be the parents of y_i labeled with c and c' resp., and $+p(w)$ such that $label(y_i) = PG\text{-}Res(c, c', +p(w))$. Thus $label(z) \subseteq label(y_i) \cup \{+p(w)\}$ and $label(z') \subseteq label(y_i) \cup \{-p(w)\}$. Moreover as G' is without exchangeable pair, if there are a literal $+p(u)$ in G' and a homomorphism π from G to a completion of H such that $\pi(u) = w$ then for any literal $-p(u)$ in G' and any homomorphism π from G to a completion of H, $\pi(u) \neq w$, and therefore $-p(\pi(u)) \neq -p(w)$. We define y_{i+1} as z' if there are a literal $+p(u)$ in G' and a homomorphism π from G to a completion of H such that $\pi(u) = w$, and as z otherwise. It follows from $P(y_i)$ and the definition of y_{i+1} that $P(y_{i+1})$ also holds. Hence, there is a node x of $L(T)$ such that $P(x)$ holds. □

Theorem 3 *Let G and H be two PGs, with H being consistent. Let G' be a subgraph of G without exchangeable pair w.r.t. (G, H). If G is deducible from*

H, then there are a completion H^c of H and a homomorphism from G to H^c that maps G' to H.

Proof: This follows immediately from Property 15 and Lemma 5. □

Note. The new proof of Theorem 2 (resp. Theorem 3) provides an algorithm to find from a PG-resolution tree T of (G, H) a homomorphism from G to a given completion of H that maps G_s to H (resp. a completion of H and a homomorphism from G to this completion that maps a given subgraph of G without exchangeable pair to H) by construction of a path in T from its anti-root to one of its leaves. The algorithm induced by the proof of Theorem 2 is simple since we only have to decide at each step if a given literal is in the given completion or not. The algorithm induced by the proof of Theorem 3 is more delicate since we must decide at each step if there are a literal l in G' and a homomorphism from G to a completion of H mapping l to a given literal, which is an NP-complete problem. But computing a PG-resolution tree of (G, H) is itself difficult, as the size of the set C_G of clauses is exponential in the number of variable nodes in G.

In Section 4, we deduced Property 12 from Theorem 2, using Lemmas 1 and 2 to translate PG-deduction into conditions on propositional formulas. We present a new proof of Property 12 from Properties 15 and 16, using the natural correspondence between a PG-resolution tree and the associated resolution tree to translate PG-deduction into conditions on propositional formulas.

Lemma 6 *Let G and H be two PGs, and let G' be a ground subgraph of G contained in G_s. There is a PG-resolution tree of (G, H) iff $D_{G'}(G, H)$ is a tautology.*

Proof: We suppose that there is a PG-resolution tree $(T, (\pi_x)_{x \in L(T)})$ of (G, H). Let us show that $D_{G'}(G, H)$ is a tautology. By Properties 5 and 16 for any node x of $L(T)$, π_x can be extended to an extensible homomorphism π'_x from G' to H, and therefore $label(x) = L_{G'}(\pi'_x)$. It follows that the set F of labels of the leaves of the complementary tree T' of the resolution tree associated with T is the set of clauses $\neg C_{G'}(\pi'_x)$ for all nodes x of $L(T)$. As T' is a resolution tree, by Property 14 F is unsatisfiable, so the negation of the conjunction of the clauses $\neg C_{G'}(\pi'_x)$ in F, i.e. the disjunctions of propositions $C_{G'}(\pi'_x)$ for all nodes x of $L(T)$, is a tautology. Hence $D_{G'}(G, H)$ is a tautology.
Conversely, we suppose that $D_{G'}(G, H)$ is a tautology. Let us show that there is a PG-resolution tree of (G, H). Let F be the set of clauses $\neg C_{G'}(\pi)$ for all extensible homomorphisms π from G' to H. As $D_{G'}(G, H)$ is a tautology, F is unsatisfiable, and therefore has a resolution tree T by Property 14. Let T' be the PG-resolution

tree associated with the complementary tree of T. For each node x of $L(T')$ there is an extensible homomorphism π from G' to H such that $label(x) = L_{G'}(\pi)$, and therefore $label(x)$ satisfies the condition required on labels of a PG-resolution tree of (G, H), with π_x equal to the restriction of π to T_G. Hence $(T', (\pi_x)_{x \in L(T')})$ is a PG-resolution tree of (G, H). □

Property 12 *Let G and H be two PGs, with H being consistent, and let G' be a ground subgraph of G contained in G_s. G is deducible from H iff $D_{G'}(G, H)$ is a tautology.*

Proof: This follows immediately from Property 15 and Lemma 6. □

6 Extensions

This section presents two extensions of previous results, on one hand by integrating a preorder on relation names, which allows to take a light ontology into account, and on the other hand by refining the notion of exchangeable literals in order to reduce their number.

6.1 Preordered predicates

In knowledge-based systems, an ontology describes the categories (or classes of objects) of an application domain, called concepts, and the possible relations between instances of these concepts. The set of concepts is usually provided with a so-called subsumption or generalization/specialization relation, which is a partial order, or a preorder in the case where several concepts can be equivalent. The set of relations can also be structured in the same way.

We show in this section that previous results can be extended to take a light ontology $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \mathcal{I})$ into account, where $\mathcal{C}$ and $\mathcal{R}$ are preordered sets of concepts and relations, respectively, and $\mathcal{I}$ is a set of individual names; only relations with the same arity are comparable according to this preorder. This ontology is said to be light, because concepts and relations are atomic (in the sense that they do not have a definition) and the only relationship among them is the preorder (noted $\leq$). $t_2 \leq t_1$ means that t_2 is a specialization of t_1, or t_2 is subsumed by t_1. A light ontology can be seen as the vocabulary (also called *support*) in conceptual graphs, and as a *TBox* composed of inclusions between atomic concepts and binary relations, called roles, in description logics.

Concepts are logically translated into unary predicates and relations of arity k into k-ary predicates. For simplicity, we use the same name for a concept or relation t and its predicate, and keep the symbol $\leq$ for the induced preorder on predicates. The set of logical formulas $\Phi(\mathcal{O})$ assigned to a light ontology $\mathcal{O}$ is as follows: for all predicates t_1 and t_2 with the same arity k, if $t_2 \leq t_1$, one has the formula $\forall x_1 \ldots x_k(t_2(x_1 \ldots x_k) \rightarrow t_1(x_1 \ldots x_k))$.

All notions presented in Section 2 can be extended to a light ontology [ML07]. Let $\mathcal{L}_{\mathcal{O}} = (\mathcal{C} \cup \mathcal{R}, \mathcal{I})$ be the logical language associated with a light ontology $\mathcal{O}$. There are several ways of extending PGs to encode FOL$\{\exists, \wedge, \neg_a\}$ formulas on $\mathcal{L}_{\mathcal{O}}$. A literal of form $\sim t(e)$, where t is a concept predicate, can be represented in the label of the term node assigned to e; in this case a term node is labeled by a pair $(\{\sim t_1 \ldots \sim t_p\}, m)$, where $t_1 \ldots t_p$ is a set of concepts and $m \in \{*\} \cup \mathcal{I}$. For this translation, we must extend the label comparison and adapt the piece notion. A simpler translation involves considering concept predicates as unary relations and leaving the labels of term nodes unchanged. For simplicity we choose this second representation in this paper, and we call $\mathcal{R}_{\mathcal{O}}$ the set of relation names available in PG labels. Thus, if $\mathcal{O} = (\mathcal{C}, \mathcal{R}, \mathcal{I})$ then $\mathcal{R}_{\mathcal{O}} = \mathcal{C} \cup \mathcal{R}$.

Relation node labels are preordered as follows: $+r_1 \leq +r_2$ (resp. $-r_2 \leq -r_1$) if $r_1 \leq r_2$, and $+r_1$ and $-r_2$ are incomparable. The definition of a homomorphism takes this order into account. In a PG homomorphism π, we now have $l_H(\pi(r)) \leq l_G(r)$. Let us point out that the preorder can be compiled, so that labels can be compared in constant (or almost constant) time; thus the preorder does not introduce overhead complexity. Property 1 still holds, with "$\sim p(\sigma(u))$ is a literal in $\Phi(H)$" being replaced with "there is a literal $\sim q(\sigma(u))$ in $\Phi(H)$ with $\sim q \leq \sim p$". A PG is consistent if it does not contain contradictory literals, i.e. $+r(u)$ and $-s(u)$ with $r \leq s$. Given this extended definition of consistency, the definitions of a complete PG and of a completion of a consistent PG w.r.t. a set of relation names $\mathcal{R}$ as well as the definition of PG-deduction are unchanged. Theorem 1 still holds with $\Phi(H) \models \Phi(G)$ being replaced with $\Phi(\mathcal{O}), \Phi(H) \models \Phi(G)$. We have to adapt the definitions of opposite literals and of the completion vocabulary w.r.t. (G, H). We first give the following definition and Lemmas.

Definition 13 ($H^{\mathcal{O}}$) *Let H be a consistent PG on a light ontology $\mathcal{O}$. $H^{\mathcal{O}}$ denotes the PG obtained from H by adding each literal $\sim q(u)$ not already present in H such that there is a literal $\sim p(u)$ in H with $\sim p \leq \sim q$.*

Lemma 7 *Let H be a consistent PG on a light ontology $\mathcal{O}$, and let L be a consistent set of literals such that for any literal l in L, $\overline{l}$ is not a literal in $H^{\mathcal{O}}$. Then $H^{\mathcal{O}} + L$ is consistent.*

Proof: We assume for contradiction that $H^{\mathcal{O}} + L$ is inconsistent. Let l and l' be contradictory literals in $H^{\mathcal{O}} + L$. As L is consistent, at least one of l and l', say l, is in $H^{\mathcal{O}}$. Let $l = \sim p(u)$, then $l' = \overline{\sim q}(u)$ for some q such that $\sim p \leq \sim q$. Hence $\overline{l'}$ is in $H^{\mathcal{O}}$ and therefore l' in not in L. It follows that l' is also in $H^{\mathcal{O}}$. As l and l' are in $H^{\mathcal{O}}$, there are literals $\sim p_1(u)$ and $\overline{\sim q_1}(u)$ in H with $\sim p_1 \leq \sim p \leq \sim q \leq \sim q_1$, so $\sim p_1(u)$ and $\overline{\sim q_1}(u)$ are contradictory literals in H, which contradicts the consistency of H. □

Lemma 8 *Let H be a consistent PG on a light ontology $\mathcal{O}$, and let $\mathcal{R}$ be a subset of $\mathcal{R}_{\mathcal{O}}$. Let $H^{c+}_{\mathcal{R}}$ (resp. $H^{c-}_{\mathcal{R}}$) be the PG obtained from H by adding for each atom $p(u)$ not already occurring in H such that p is in $\mathcal{R}$ and u is an $arity(p)$-tuple in H:*
$+p(u)$ if $+p(u)$ is a literal in $H^{\mathcal{O}}$,
$-p(u)$ if $-p(u)$ is a literal in $H^{\mathcal{O}}$,
$+p(u)$ (resp. $-p(u)$) if neither $+p(u)$ nor $-p(u)$ is a literal in $H^{\mathcal{O}}$.
Then $H^{c+}_{\mathcal{R}}$ (resp. $H^{c-}_{\mathcal{R}}$) is a completion of H w.r.t. $\mathcal{R}$.

Proof: Let $H^c = H^{c+}_{\mathcal{R}}$ (resp. $H^{c-}_{\mathcal{R}}$). It is sufficient to show that H^c is consistent, which immediately follows from Lemma 7 with L being the set of literals in $H^c \setminus H^{\mathcal{O}}$ (L is consistent since it contains only positive (resp. negative) literals). □

It follows from Lemma 8 that for any subset $\mathcal{R}$ of $\mathcal{R}_{\mathcal{O}}$, a consistent PG H has at least one completion w.r.t. $\mathcal{R}$.

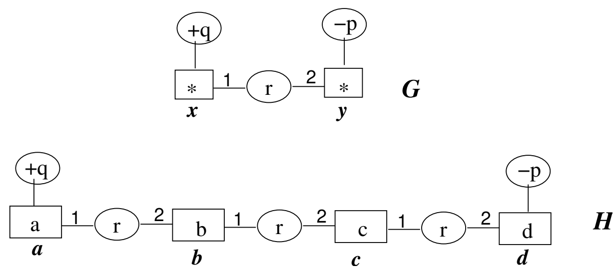
Lemma 9 *Let G and H be two PGs on a light ontology $\mathcal{O}$, with H being consistent. If G is deducible from H then each relation node label in G is also a label in $H^{\mathcal{O}}$.*

Proof: Let $\sim p$ be a label in G, let $H^c = H^{c-}_{\mathcal{R}_{\mathcal{O}}}$ if $\sim p = +p$ and $H^{c+}_{\mathcal{R}_{\mathcal{O}}}$ otherwise. By Lemma 8, H^c is a completion of H w.r.t. $\mathcal{R}_{\mathcal{O}}$. Let π be a homomorphism from G to H^c. As the polarity of $\sim p$ is opposite to that of the literals in $H^c \setminus H^{\mathcal{O}}$, each node of G with label $\sim p$ is mapped to a node of $H^{\mathcal{O}}$, and therefore $\sim p$ is a label in $H^{\mathcal{O}}$. □

Definition 14 (Weakly and strongly opposite literals) *Let r and s be relation names with $r \leq s$. Labels $-r$ and $+s$ (resp. literals $-r(u)$ and $+s(v)$) are said* weakly opposite*, and labels $+r$ and $-s$ (resp. literals $+r(u)$ and $-s(v)$) are said* strongly opposite.

Note that if $u = v$ then strongly opposite literals $+r(u)$ and $-s(u)$ are contradictory, which is not the case for weakly opposite literals $-r(u)$ and $+s(u)$. We can now give the definition of the completion vocabulary w.r.t. $(G, H, \mathcal{O})$.

Definition 15 (Completion vocabulary w.r.t. $(G, H, \mathcal{O})$**)** *Let G and H be two PGs on a light ontology $\mathcal{O}$. The* completion vocabulary w.r.t. $(G, H, \mathcal{O})$ *is the set of relation names appearing in a pair of weakly opposite labels $\{-r, +s\}$ such that both $-r$ and $+s$ have occurrences in G and in $H^{\mathcal{O}}$.*



$p \leq q$

Figure 10: PGs with preordered relation names

For instance, if G and H are the PGs shown in Figure 10 with $p \leq q$ then the completion vocabulary w.r.t. $(G, H, \mathcal{O})$ is the set $\mathcal{R} = \{p, q\}$. G can be mapped to each completion H^c of H w.r.t. $\mathcal{R}$: if H^c contains the literal $-p(b)$ then x and y can be mapped to a and b, otherwise if H^c contains the literal $-p(c)$ then x and y can be mapped to b and c, and otherwise to c and d. We will show that this holds if and only if G is deducible from H, which justifies the definition of the completion vocabulary w.r.t. $(G, H, \mathcal{O})$. Note that if we had the order $q \leq p$, then $\{+q, -p\}$ would be a pair of strongly opposite labels and the completion vocabulary would be empty, which is in accordance with the fact that G would not be deducible from H.

Property 17 *Let G and H be two PGs on a light ontology $\mathcal{O}$, with H being consistent, and let $\mathcal{R}$ be the completion vocabulary w.r.t. $(G, H, \mathcal{O})$. G is deducible from H if and only if G can be mapped to each completion of H w.r.t. $\mathcal{R}$.*

Proof: We have to show that G can be mapped to each completion of H w.r.t. $\mathcal{R}_{\mathcal{O}}$ iff G can be mapped to each completion of H w.r.t. $\mathcal{R}$. The implication from right to left holds since $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{O}}$. We suppose that G can be mapped to each completion of H w.r.t. $\mathcal{R}_{\mathcal{O}}$. Let H^c be a completion of H w.r.t. $\mathcal{R}$. Let us show that G can be mapped to H^c. For this we define from H^c a completion H' of H w.r.t. $\mathcal{R}_{\mathcal{O}}$, and we will define from a homomorphism π' from G to H' a homomorphism π from G to H^c.
Let H' be the PG obtained from H^c by adding for each atom $p(u)$ not already present in H^c such that p is in $\mathcal{R}_{\mathcal{O}}$ and u is an $arity(p)$-tuple in H:

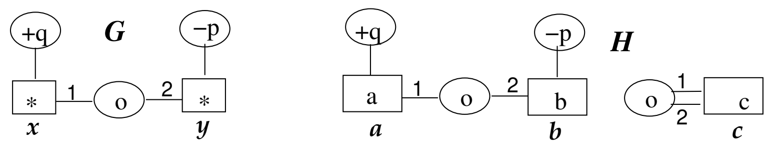
$+p(u)$ if $+p(u)$ is a literal in $(H^c)^{\mathcal{O}}$,
$-p(u)$ if $-p(u)$ is a literal in $(H^c)^{\mathcal{O}}$,
otherwise, $+p(u)$ if there is $r \in \mathcal{R}_{\mathcal{O}}$ such that $r \leq p$ and $-r$ is a label in G, and $-p(u)$ if there is no such r.
Let us show that H' is a completion of H w.r.t. $\mathcal{R}_{\mathcal{O}}$. It is sufficient to show that H' is consistent. H^c is consistent and for any literal l in $H' \setminus (H^c)^{\mathcal{O}}$, $\bar{l}$ is not in $(H^c)^{\mathcal{O}}$, so by Lemma 7 it is sufficient to prove that $H' \setminus (H^c)^{\mathcal{O}}$ is consistent. We suppose for contradiction that $H' \setminus (H^c)^{\mathcal{O}}$ is inconsistent. Let $+p(u)$ and $-q(u)$ be contradictory literals in $H' \setminus (H^c)^{\mathcal{O}}$, i.e. with $p \leq q$. As $+p(u)$ is in $H' \setminus (H^c)^{\mathcal{O}}$ there is $r \in \mathcal{R}_{\mathcal{O}}$ such that $r \leq p$ and $-r$ is a label in G, so $r \leq q$ and therefore $-q(u)$ is not in $H' \setminus (H^c)^{\mathcal{O}}$, a contradiction. Thus H' is a completion of H w.r.t. $\mathcal{R}_{\mathcal{O}}$.
Let π' be a homomorphism from G to H'. We define from π' a homomorphism π from G to H^c as follows. For any term node t of G, $\pi(t) = \pi'(t)$. Let l be a literal in G, and $l' = \pi(l) =\sim p(v)$. If l' is in $(H^c)^{\mathcal{O}}$ then there is a literal $\sim q(v)$ in H^c with $\sim q \leq \sim p$, and we define $\pi(l) = \sim q(v)$. We suppose now that l' is not in $(H^c)^{\mathcal{O}}$. $l' \neq -p(v)$, otherwise l would be in the form $-r(u)$ with $r \leq p$, so $-r$ would be a label in G and $-p(v)$ would not be a literal in $H' \setminus (H^c)^{\mathcal{O}}$. So $l' = +p(v)$, and l is in the form $+s(u)$ with $p \leq s$. As $+p(v)$ is a literal in $H' \setminus (H^c)^{\mathcal{O}}$, there is $r \in \mathcal{R}_{\mathcal{O}}$ such that $r \leq p$ and $-r$ is a label in G. By Lemma 9, $-r$ and $+s$ are also labels in $H^{\mathcal{O}}$. It follows that $\{-r, +s\}$ is a pair of weakly opposite labels such that both $-r$ and $+s$ have occurrences in G and in $H^{\mathcal{O}}$, so s is in $\mathcal{R}$, and therefore either $+s(v)$ or $-s(v)$ is a literal in H^c. $-s(v)$ is not a literal in H^c, otherwise $+p(v)$ and $-s(v)$ would be contradictory literals in H'. So $+s(v)$ is a literal in H^c, and we define $\pi(l) = +s(v)$. Thus we have defined a homomorphism π from G to H^c. □

From now on, completions consider implicitly the completion vocabulary w.r.t. $(G, H, \mathcal{O})$. The definition of an exchangeable pair is extended as follows.

Definition 16 (Exchangeable pair w.r.t. $(G, H, \mathcal{O})$) *An* exchangeable *pair w.r.t. $(G, H, \mathcal{O})$ is a pair $\{-r(u), +s(v)\}$ of weakly opposite literals in G, such that there are two completions of H, say H_1 and H_2, and two homomorphisms π_1 and π_2, respectively from G to H_1 and from G to H_2, with $\pi_1(u) = \pi_2(v)$, $-r(\pi_1(u))$ is a literal in H_1 and $+r(\pi_2(v))$ is a literal in H_2.*

For instance, if G and H are the PGs in Figure 10 with $p \leq q$, then $\{+q(x), -p(y)\}$ is an exchangeable pair, since x and y can be mapped to b by homomorphisms from G to completions of H containing $+p(b)$ and $-p(b)$ respectively.

This definition calls for a few comments. First, condition "$-r(\pi_1(u))$ is a literal in H_1 and $+r(\pi_1(u))$ is a literal in H_2" could be replaced by "$-s(\pi_1(u))$ is a literal in H_1 and $+s(\pi_1(u))$ is a literal in H_2" or, for a symmetrical definition, by the disjunction of these two conditions. As we aim at reducing the number of exchangeable pairs as much as possible, we choose this non-symmetrical but more restrictive definition. Secondly, a weaker definition of an exchangeable pair could have been considered: it is obtained from the flat case (Definition 6) by simply replacing "opposite" with "weakly opposite", i.e. without adding the condition that $-r(\pi_1(u))$ is a literal in H_1 and $+r(\pi_1(u))$ is a literal in H_2. This weaker definition is not equivalent to the above Definition 16, even if we add the condition that neither $-r(\pi_1(u))$ nor $+r(\pi_1(u))$ is a literal in H. For instance, if G and H are the PGs shown in Figure 11 with $p \leq q$, then the pair $\{+q(x), -p(y)\}$ is



$p \leq q$

Figure 11: Weak exchangeability

"weakly" exchangeable but is not exchangeable. It is weakly exchangeable since x and y can be mapped to c by a homomorphism from G to a completion of H obtained by adding $-p(c)$ and $+q(c)$ (in this case, $H_1 = H_2$). It is not exchangeable because x and y are necessarily both mapped to c by π_1 (or by π_2), thus H_2 necessarily contains $+p(c)$ and the only way of mapping $-p(y)$ is to have $-p(c)$ or $-q(c)$in H_2, which makes it inconsistent. Intuitively, the weaker definition of an exchangeable pair seems to be insufficient since it does not necessarily involve the law of the excluded-middle, which was the motivation for introducing exchangeable pairs. This intuition is confirmed by the logical resolution approach, in which exchangeable pairs are represented by pairs of complementary literals involved in resolution operations.

Given the extended definition of an exchangeable pair, the definitions of an exchangeable literal (w.r.t. $(G, H, \mathcal{O})$) and of the socle G_s of G (w.r.t. $(H, \mathcal{O})$) are unchanged. Property 4 still holds, with H^{c+} (resp. H^{c-}) becoming $H^{c+}_{\mathcal{R}}$ (resp. $H^{c-}_{\mathcal{R}}$) defined in Lemma 8, where $\mathcal{R}$ is the completion vocabulary w.r.t. $(G, H, \mathcal{O})$. Given the extended definitions of the completion vocabulary (w.r.t. $(G, H, \mathcal{O})$), the definition of a ground subgraph of G (w.r.t. $(H, \mathcal{O})$) is unchanged, and the definition of an extensible homomorphism is extended as follows.

Definition 17 (Extensible homomorphism w.r.t. $(G, H, \mathcal{O})$**)** *A homomorphism π from a ground subgraph G' of G to H is* extensible *(w.r.t. $(G, H, \mathcal{O})$) if it satisfies*

1. for any literal $\sim r(u)$ *in* $G \setminus G'$, $\overline{\sim r}(\pi(u))$ *is not in* $H^{\mathcal{O}}$*;*

2. for all strongly opposite literals $+r(u)$ *and* $-s(v)$ *in* $G \setminus G'$, $\pi(u) \neq \pi(v)$*.*

Property 5 still holds, with the following modification of the proof that conditions 1 and 2 are sufficient for π to be extendable to a homomorphism from G to a completion of H. We suppose that π satisfies conditions 1 and 2. Let L be the set of literals $\sim r(\pi(u))$ for every literal $\sim r(u)$ in $G \setminus G'$ such that $\sim r(\pi(u))$ is not already present in H. By condition 1, for any literal in L, $\bar{l}$ is not a literal in $H^{\mathcal{O}}$, and by condition 2 L is consistent. It follows by Lemma 7 that $H^{\mathcal{O}} + L$ is consistent, and therefore $H + L$ is consistent. Hence, by Lemma 8, $H + L$ has a completion H^c, which is also a completion of H since G' is a ground subgraph of G, and π can be extended to a homomorphism from G to H^c.

Let us now consider Theorems 2 and 3, which are fundamental for the complexity results. The proofs of these theorems given in Section 3 extend to preordered relation names with the weak definition of exchangeable pairs, but they do not with the above Definition 16. The reason is that the set of complementary literals of the literals in R (with R being consistent) is no longer necessarily consistent, since R may contain weakly opposite literals in the form $-r(u)$ and $+s(u)$, whose complementary literals are contradictory. However, these theorems still hold. To show it, we extend the proofs given in Section 5, which use a PG-resolution tree of (G, H), as follows. Since $\Phi(H) \models \Phi(G)$ is replaced with $\Phi(\mathcal{O}), \Phi(H) \models \Phi(G)$, the set $F = C_G \cup C_H$ is replaced with $C_G \cup C_H \cup C_{\mathcal{O}}$, where $C_{\mathcal{O}}$ is the set of all clauses in the form $\{\neg r(u), s(u)\}$ with $r \leq s$. In a resolution tree, clauses of $C_{\mathcal{O}}$ allow to increase literals, i.e. to replace a literal $\sim p(u)$ by $\sim q(u)$ with $\sim p \leq \sim q$, in clauses not belonging to $C_{\mathcal{O}}$ (such an operation on a clause of $C_{\mathcal{O}}$ would result into a clause of $C_{\mathcal{O}}$ and therefore would be useless). We may assume w.l.o.g. that resolution operations involving clauses of C_H or $C_{\mathcal{O}}$ are performed first, so there is a resolution tree whose leaves are labeled with clauses obtained from clauses of C_G by removing all literals l such that $\bar{l}$ is a literal in $\Phi(H^{\mathcal{O}})$ and increasing the remaining literals. Thus, for any resolution operation $Res(c, c', p(w))$ performed in this resolution tree, c contains the literal $p(w)$ obtained by increasing some literal $r(w)$ with $r \leq p$ and c' contains the literal $\neg p(w)$ obtained by increasing some literal $\neg s(w)$ with $p \leq s$. Instead of increasing $r(w)$ to $p(w)$ and $\neg s(w)$ to $\neg p(w)$, we can leave $r(w)$ unchanged and increase $\neg s(w)$ to $\neg r(w)$, and do the resolution operation w.r.t. the literal $r(w)$. In other words, we can increase only negative literals. Let us show that moreover, there is such a resolution tree T such that none of the clauses labeling its leaves contains a clause of $C_{\mathcal{O}}$ (which is needed to assure label consistency of the leaves in the PG-resolution tree associated with the complementary tree of T). Let T be a resolution tree whose leaves are labeled with clauses obtained from clauses of C_G by removing all literals l such that $\bar{l}$ is a literal in $\Phi(H^{\mathcal{O}})$ and increasing the remaining negative literals, with the minimum number of clauses labeling the leaves, let F be the set of clauses labeling the leaves of T, and let c be a clause in F. Let us show that c does not contain any clause of $C_{\mathcal{O}}$. We suppose for contradiction that c contains a clause c' of $C_{\mathcal{O}}$. Then the set $(F \setminus \{c\}) \cup \{c'\}$ is also unsatisfiable, and therefore has a resolution tree T'. If c' labels some leaves of T', we can remove these leaves and increase some negative literals for T' to remain a resolution tree. We obtain a resolution tree of the desired form whose leaves are labeled with clauses of $F \setminus \{c\}$, which contradicts the definition of T. Thus, the definition of a PG-resolution tree of (G, H) is extended as follows.

Definition 18 (PG-resolution tree of $(G, H, \mathcal{O})$) *Let G and H be two PGs. A* PG-resolution tree of $(G, H, \mathcal{O})$ *is a structure $(T, (\pi_x)_{x \in L(T)})$ where T is a PG-resolution tree such that for each node x of $L(T)$, $label(x)$ is consistent and π_x is a mapping from T_G to T_H such that $label(x)$ is obtained from the set $\{\sim p(\pi_x(u)) \mid \sim p(u)$ is a literal in G and $\sim p(\pi_x(u))$ is not a literal in $H^{\mathcal{O}}\}$ by replacing each positive literal $+p(\pi_x(u))$ with a literal in the form $+r(\pi_x(u))$ with $r \leq p$.*

For instance, if G and H are the PGs shown in Figure 10 with $p \leq q$ then the tree given in Figure 9 is a PG-resolution tree of $(G, H, \mathcal{O})$. Property 15 still holds, and Property 16 is extended as follows.

Property 18 *Let G and H be two PGs on a light ontology $\mathcal{O}$, and let $(T, (\pi_x)_{x \in L(T)})$ be a PG-resolution tree of $(G, H, \mathcal{O})$. For any node x of $L(T)$, π_x can be extended to a homomorphism from G to a completion of H containing each literal in $label(x)$, and any such homomorphism maps G_s to H.*

Proof: Let x be a node of $L(T)$. Let us show that π_x can be extended to a homomorphism from G to a completion of H containing $label(x)$. For this, it is sufficient to show that there is a completion of H containing $label(x)$. To prove this, it is sufficient to show that the following Propositions a) and b) hold:
a) $H + label(x)$ is consistent,
b) each relation name in $label(x)$ is in the completion vocabulary w.r.t. $(G, H, \mathcal{O})$,
since it follows from Proposition a) and Lemma 8 that $H + label(x)$ has a completion, which is also a completion of H by Proposition b).
Let us show Proposition a). H and $label(x)$ are consistent and by Lemma 3, for any literal l in $label(x)$, $\bar{l}$ is in the label of a node of $L(T)$, and therefore is not a literal in $H^{\mathcal{O}}$, so by Lemma 7 $H + label(x)$ is consistent.

Let us show Proposition b). Let r be a relation name in $label(x)$. By Lemma 3, $+r$ and $-r$ have occurrences in labels of nodes of $L(T)$, and therefore there is a relation name p with $r \leq p$ such that $+p$ and $-r$ have occurrences in G. By Property 15 and Lemma 9, $+p$ and $-r$ also have occurrences in $H^{\mathcal{O}}$. Hence Proposition b) holds, which completes the proof that π_x can be extended to a homomorphism from G to a completion of H containing $label(x)$.
Let π'_x be a homomorphism extending π_x from G to a completion H'_x of H containing $label(x)$. Let us show that π'_x maps G_s to H, i.e that each literal $\sim p(u)$ in G such that $\sim p(\pi'_x(u))$ is not a literal in $H^{\mathcal{O}}$ is exchangeable. Let $\sim p(u)$ be a literal in G such that $\sim p(\pi'_x(u))$ is not a literal in $H^{\mathcal{O}}$. Then there is a literal $\sim r(\pi'_x(u))$ in $label(x)$ with $\sim r \leq \sim p$ (and $r = p$ if $\sim p = -p$). By Lemma 3, there is a node y of $L(T)$ such that $\overline{\sim r}(\pi'_x(u))$ is a literal in $label(y)$. So there is a literal $\overline{\sim q}(v)$ in G with $\sim q \leq \sim r$ (and $r = q$ if $\sim p = +p$) and a homomorphism π'_y (extending π_y) from G to a completion H'_y of H containing $label(y)$ such that $\pi'_x(u) = \pi'_y(v)$. As $\sim q \leq \sim r$ and $\sim r \leq \sim p$, $\sim q \leq \sim p$, so $\overline{\sim q}(v)$ and $\sim p(u)$ are weakly opposite literals in G, and as H'_x contains $label(x)$ and H'_y contains $label(y)$, $\sim r(\pi'_x(u))$ is a literal in H'_x and $\overline{\sim r}(\pi'_x(u))$ is a literal in H'_y, with $-r$ being the negative label in $\{\overline{\sim q}, \sim p\}$. It follows that $\{\overline{\sim q}(v), \sim p(u)\}$ is an exchangeable pair, hence $\sim p(u)$ is exchangeable. $\square$

Property 18 is indeed an extension of Property 16 since the added condition that the considered completion of H contains each literal in $label(x)$ is implicitly satisfied in absence of preorder on relation names. This condition is necessary to prove that an extension of π_x maps G_s to H because of the condition in the definition of an exchangeable pair that $-r(\pi_1(u))$ is a literal in H_1 and $+r(\pi_1(u))$ is a literal in H_2, which is also implicitly satisfied in absence of preorder on relation names.

The proofs of Theorems 2 and 3 using PG-resolution trees still hold, if we replace in the proof of Lemma 5 the definition of $P(y)$ by:
$P(y)$: for any literal $\sim p(w)$ in $label(y)$, any literal $\sim q(u)$ in G' with $p \leq q$ if $\sim p = +p$ and $p = q$ otherwise, and any homomorphism π from G to a completion of H containing the literal $\sim p(w)$, $\pi(u) \neq w$.
and the definition of y_{i+1} by:
y_{i+1} is defined as z' if there are a literal $+q(u)$ in G' with $p \leq q$ and a homomorphism π from G to a completion of H containing $+p(w)$ such that $\pi(u) = w$, and z otherwise.

Property 4 still holds, with H^{c+} (resp. H^{c-}) being the completion $H^{c+}_{\mathcal{R}}$ (resp. $H^{c-}_{\mathcal{R}}$) defined in Lemma 8, where $\mathcal{R}$ is the completion vocabulary w.r.t. $(G, H, \mathcal{O})$.

Property 12 is extended by replacing formula $D_{G'}(G, H)$ with $D_{G'}(G, H, \mathcal{O})$.

Notations 2 *Let G and H be two PGs on a light ontology $\mathcal{O}$, with H being consistent, and let G' be a ground subgraph of G.*
$P_{H,\mathcal{O}}$ denotes the set of atoms of $\Phi(H^c \setminus H^{\mathcal{O}})$, where H^c is an arbitrary completion of H, seen as the set of atoms of a language in propositional logic.
For any extensible homomorphism π from G' to H, $L_{G'}(\pi)$ denotes the set of literals l such that $l = \sim p(\pi(u))$ for some literal $\sim p(u)$ in G and l is not in $H^{\mathcal{O}}$, and $C_{G'}(\pi)$ denotes the conjunction of the literals in $L_{G'}(\pi)$ seen as a proposition on $P_{H,\mathcal{O}}$.
$D_{G'}(G, H, \mathcal{O}) = D_{G'}(G, H) \vee D(\mathcal{O})$, where $D_{G'}(G, H)$ denotes the disjunction of the propositions $C_{G'}(\pi)$ for all extensible homomorphisms π from G' to H and $D(\mathcal{O})$ denotes the disjunction of the conjunctions $r(u) \wedge \neg s(u)$ for all atoms $r(u)$ and $s(u)$ in $P_{H,\mathcal{O}}$ such that $r \leq s$.
Omission of subscript G' means that G' is equal to G_s.

For instance, if G and H are the PGs shown in Figure 10 with $p \leq q$ then $D(G, H, \mathcal{O})$ is a disjunction in the form $\neg p(b) \vee (q(b) \wedge \neg p(c)) \vee q(c) \vee (p(b) \wedge \neg q(b)) \vee (p(c) \wedge \neg q(c)) \vee D'$, and therefore is a tautology.

Lemma 10 *Let G and H be two PGs on a light ontology $\mathcal{O}$, and let G' be a ground subgraph of G contained in G_s. There is a PG-resolution tree of $(G, H, \mathcal{O})$ iff $D_{G'}(G, H, \mathcal{O})$ is a tautology.*

Proof: Assuming that there is a PG-resolution tree $(T, (\pi_x)_{x \in L(T)})$ of $(G, H, \mathcal{O})$, let us show that $D_{G'}(G, H, \mathcal{O})$ is a tautology. For any node x of $L(T)$, π_x can be extended to an extensible homomorphism π'_x from G' to H, such that $label(x)$ is obtained from $L_{G'}(\pi'_x)$ by decreasing positive labels. Let T' be the complementary tree of the resolution tree associated with T. Each leaf of T' is labeled with a clause obtained from a clause in the form $\neg C_{G'}(\pi'_x)$ by increasing negative predicates. Then there is a resolution tree T'' whose leaves are labeled with clauses in the form $\neg C_{G'}(\pi'_x)$ and clauses of $C_{\mathcal{O}}$, that are negations of conjunctions in $D(\mathcal{O})$. As the set of labels of the leaves of T'' is unsatisfiable by Property 14, $D_{G'}(G, H, \mathcal{O})$ is a tautology.
Conversely, we assume that $D_{G'}(G, H, \mathcal{O})$ is a tautology. Let us show that there is a PG-resolution tree of $(G, H, \mathcal{O})$. Let F be the unsatisfiable set of clauses such that $\neg D_{G'}(G, H, \mathcal{O})$ is the conjunction of the clauses in F. As discussed in the paragraph preceding Definition 18, we can build from a resolution tree of F a PG-resolution tree of $(G, H, \mathcal{O})$. $\square$

Property 19 *Let G and H be two PGs on a light ontology $\mathcal{O}$, with H being consistent, and let G' be a ground subgraph of G contained in G_s. G is deducible from H iff $D_{G'}(G, H, \mathcal{O})$ is a tautology.*

Complexity results of Section 4 are preserved since they follow from Theorems 2 and 3 and Property 12, from the NP-completeness of PG-homomorphism, which is also preserved when relation names are preordered, and from the fact that any problem is at least as difficult as in absence of preorder on relation names (in particular DEDUCTION$_3$ is still co-NP-difficult).

6.2 Refining Completions and Exchangeability

In this section we see how to reduce the set of literals added to H to obtain a completion of H, which in turn reduces the number of exchangeable pairs. We already restricted the set of literals added by defining the completion vocabulary w.r.t. (G, H). The idea is that the obtained completions of H must satisfy the following fundamental property, denoted by *Completion Property*: G is deducible from H if and only if G can be mapped to each completion of H. By Theorem 2, it is sufficient to add to H literals l such that at least one exchangeable literal in G can potentially be mapped to l. It follows that any literal l in a completion of H that is not in H and such that no exchangeable literal in G can be mapped to l can be removed from this completion. This restriction on completions of H induces a reduction of the set of homomorphisms from G to completions of H, and therefore of the set of exchangeable pairs, so that new literals in completions of H become useless and can be removed. This operation can be repeated, reducing both the set of literals added in completions of H and the set of exchangeable pairs until stability is obtained. We first refine the notion of completion vocabulary, then we introduce exchangeable triples.

6.2.1 Completion Vocabulary

We defined the completion vocabulary w.r.t. (G, H) as the set of relation names with positive and negative occurrences in G and in H, with an extension of this definition and the proof of Completion Property (Property 17) in the case of preordered predicates. We will give a general process leading to an inclusion-smaller completion vocabulary (and therefore an inclusion-smaller set of exchangeable pairs) with a more general and simpler proof of Completion Property.

The idea is that if a relation name in the completion vocabulary does not appear in any exchangeable literal then it can be removed from the completion vocabulary $\mathcal{R}$, which in turn will reduce the set of exchangeable literals w.r.t. $(G, H, \mathcal{R})$, i.e. defined with completions of H w.r.t. $\mathcal{R}$. Thus, we can successively restrict the completion vocabulary until it only contains relation names of exchangeable literals w.r.t. $(G, H, \mathcal{R})$. The refined completion vocabulary, denoted by $\mathcal{R}(G, H)$, is defined by Algorithm 3.

Algorithm 3: $\mathcal{R}(G, H)$

Data: G and H two PGs, with H being consistent.
Result: the refined completion vocabulary $\mathcal{R}(G, H)$.
begin
  Let $\mathcal{R}$ be the set of relation names that have both positive and negative occurrences in G and in H
  repeat
    $\mathcal{R}_1 \leftarrow \mathcal{R}$
    Let $\mathcal{R}$ be the set of relation names in exchangeable literals w.r.t. $(G, H, \mathcal{R})$
  until $\mathcal{R} = \mathcal{R}_1$;
  return $\mathcal{R}$
end

For instance, if G and H are the PGs shown in Figure 4, $\mathcal{R}$ is initialized with $\{p\}$ and is unchanged after one iteration of the repeat loop, thus $\{p\}$ is the returned value; in that case $\mathcal{R}(G, H)$ is equal to the completion vocabulary as previously defined (the refinement will be effective at the second step described in Section 6.2.2). In the general case, $\mathcal{R}$ is initialized with the completion vocabulary w.r.t. (G, H) and strictly decreases at each iteration of the repeat loop, except the last one where $\mathcal{R}$ is unchanged.

Let us show that all results of this paper still hold with this new definition of the completion vocabulary. It is sufficient to show that the proofs given in Section 5 still hold (remember that they extend to preordered predicates). For this, we need the following definition.

Definition 19 (PG-resolution tree of (G, H) on $\mathcal{R}$) *Let G and H be two PGs, and let $\mathcal{R}$ be a set of relation names. A* PG-resolution tree of (G, H) on $\mathcal{R}$ *is a PG-resolution tree of (G, H) such that each relation name appearing in labels of nodes of $L(T)$ is in $\mathcal{R}$.*

Property 16 and Lemmas 4, 5 and 6 still hold if we replace PG-resolution tree of (G, H) by PG-resolution tree of (G, H) on $\mathcal{R}$ and if completions, exchangeable pairs, G_s and ground subgraphs are defined w.r.t. $\mathcal{R}$ instead of the previously defined completion vocabulary, where $\mathcal{R}$ is a arbitrary set of relation names. In the proof of Property 16, Proposition b) becomes: "each relation name in $label(x)$ is in $\mathcal{R}$", which immediately follows from the definition of a PG-resolution tree of (G, H) on $\mathcal{R}$. The rest of the proofs is unchanged. Thus, to show that all results of this paper still hold with the refined completion vocabulary $\mathcal{R}(G, H)$ (as well as the Completion Property, which is a consequence of Theorem 2), it only remains to prove that Property 15 also extends, i.e. that the following Property holds.

Property 20 (PG-resolution on $\mathcal{R}(G, H)$) *Let G and H be two PGs, with H being consistent. G is deducible from H if and only if there is a PG-resolution tree of (G, H) on $\mathcal{R}(G, H)$.*

Proof: By Property 15 it is sufficient to show that there is a PG-resolution tree of (G, H) if and only if there is a PG-resolution tree of (G, H) on $\mathcal{R}(G, H)$. The implication from right to left is evident. Let $(T, (\pi_x)_{x \in L(T)})$ be a PG-resolution tree of (G, H). Let us show that it is a PG-resolution tree of (G, H) on $\mathcal{R}(G, H)$, i.e. that each relation name in labels of nodes of $L(T)$ is in $\mathcal{R}(G, H)$. Let $P(\mathcal{R})$ be the property defined by:
$P(\mathcal{R})$: each relation name in labels of nodes of $L(T)$ is in $\mathcal{R}$.
Let us show that $P(\mathcal{R})$ is an invariant of the repeat loop in Algorithm 3. $P(\mathcal{R})$ trivially holds at the initialization of the loop. We suppose that $P(\mathcal{R})$ holds. Let $\mathcal{R}'$ be the set of relation names in exchangeable literals w.r.t. $(G, H, \mathcal{R})$. Let us show that $P(\mathcal{R}')$ holds. As $P(\mathcal{R})$ holds, $(T, (\pi_x)_{x \in L(T)})$ is a PG-resolution tree of (G, H) on $\mathcal{R}$, so by the proof of extended Lemma 4 each relation name in labels of nodes of $L(T)$ is a relation name in some exchangeable literal w.r.t. $(G, H, \mathcal{R})$, and therefore is in $\mathcal{R}'$. Hence $P(\mathcal{R})$ is an invariant of the loop, and $(T, (\pi_x)_{x \in L(T)})$ be a PG-resolution tree of (G, H) on $\mathcal{R}(G, H)$. $\square$

It follows that all results of this paper still hold with $\mathcal{R}(G, H)$ as completion vocabulary. The definition of $\mathcal{R}(G, H)$ is unchanged in case of preordered predicates. Note that the preceding proofs still hold if we replace $\mathcal{R}(G, H)$ by one of its supersets, and in particular by the completion vocabulary as previously defined. Thus they provide a new and simpler proof of Property 17.

In practice, computing $\mathcal{R}(G, H)$ may be too costly (remember that deciding whether G has an exchangeable pair is NP-complete), but it may be possible to identify some relation names that cannot be in any exchangeable literal. For instance, if the literal $-r(e, e)$ is added to G and to H in the example of Figure 4, r becomes an element of the initial set $\mathcal{R}$ in Algorithm 3, but it is easy to see that it is not the relation name of an exchangeable literal and can be removed from $\mathcal{R}$. Thus the repeat loop can be replaced by a while loop in the form:
while a relation name r that is in no exchangeable literal w.r.t. $(G, H, \mathcal{R})$ can be "found" **do**
 remove r from $\mathcal{R}$

The while loop stops when no such relation name r can be detected, which does not mean that there is none. Hence, the obtained completion vocabulary may be only partially refined, but is in any case at least as good as the initial completion vocabulary.

6.2.2 Exchangeable triples

So far we have restricted the relation names of literals added in completions of H, but not their arguments. We will now take these arguments into account in order to further reduce the set of added literals.

Definition 20 (Triple w.r.t. (G, H)) *A* triple w.r.t. (G, H) *is a set* $\{+p(u), -p(v), w\}$ *where* $+p(u)$ *and* $-p(v)$ *are opposite literals in* G *and* w *is an* $arity(p)$*-tuple of term nodes of* H *such that neither* $+p(w)$ *nor* $-p(w)$ *is a literal in* H.

Definition 21 (completion w.r.t. $\mathcal{T}$) *Let* G *and* H *be two PGs, with* H *being consistent, and let* $\mathcal{T}$ *be a set of triples w.r.t.* (G, H)*. A* completion of H w.r.t. $\mathcal{T}$ *is a consistent PG obtained from* H *by adding, for each triple* $\{+p(u), -p(v), w\}$ *in* $\mathcal{T}$*, either the literal* $+p(w)$ *or* $-p(w)$.

Definition 22 (Exchangeable triple/pair w.r.t. $(G, H, \mathcal{T})$) *Let* G *and* H *be two PGs, with* H *being consistent, and let* $\mathcal{T}$ *be a set of triples w.r.t.* (G, H)*. An* exchangeable triple w.r.t. $(G, H, \mathcal{T})$ *is a triple* $\{+p(u), -p(v), w\}$ *w.r.t.* (G, H) *such that there are two completions of* H *w.r.t.* $\mathcal{T}$*, say* H_1 *and* H_2*, and two homomorphisms* π_1 *and* π_2*, respectively from* G *to* H_1 *and from* G *to* H_2 *such that* $\pi_1(u) = \pi_2(v) = w$*. An* exchangeable pair w.r.t. $(G, H, \mathcal{T})$ *is a pair* $\{+p(u), -p(v)\}$ *that is a subset of an exchangeable triple w.r.t.* $(G, H, \mathcal{T})$.

The set $\mathcal{T}(G, H)$, which is at the same time the set $\mathcal{T}$ of triples such that completions of H are defined w.r.t. $\mathcal{T}$ and the set of exchangeable triples w.r.t. $(G, H, \mathcal{T})$, is defined by Algorithm 4.

Let us illustrate Algorithm 4 on the PGs G and H pictured in Figure 4. $\mathcal{T}$ is initialized with $\{\{l_1, l_2, b\}, \{l_1, l_2, d\}\}$. It becomes $\{\{l_1, l_2, b\}\}$ after the first iteration of the repeat loop, and becomes empty after the second one, since l_1 can no longer be mapped to $+p(b)$ by a homomorphism from G to a completion of H w.r.t. $\mathcal{T}$ since no such completion of H contains the literal $-p(d)$. Hence, there is no exchangeable pair w.r.t. $(G, H, \mathcal{T}(G, H))$, and since there is no homomorphism from G to H, it follows that G is not deducible from H (provided that Property 8 still holds, which is checked below).

We prove that all results of this paper still hold in a similar way as for $\mathcal{R}(G, H)$, replacing $\mathcal{R}(G, H)$ by $\mathcal{T}(G, H)$ and defining a PG-resolution tree of (G, H) on $\mathcal{T}$ as follows.

Algorithm 4: $\mathcal{T}(G, H)$

Data: G and H two PGs, with H being consistent.
Result: the set $\mathcal{T}(G, H)$.
begin
  Let $\mathcal{T}$ be the set of triples $\{+p(u), -p(v), w\}$ w.r.t. (G, H) such that $\{+p(u), -p(v)\}$ is an exchangeable pair w.r.t. $(G, H, \mathcal{R}(G, H))$
  repeat
    $\mathcal{T}_1 \leftarrow \mathcal{T}$
    Let $\mathcal{T}$ be the set of exchangeable triples w.r.t. $(G, H, \mathcal{T})$
  until $\mathcal{T} = \mathcal{T}_1$;
  return $\mathcal{T}$
end

Definition 23 (PG-resolution tree of (G, H) on $\mathcal{T}$) *Let G and H be two PGs, and let $\mathcal{T}$ be a set of triples w.r.t. (G, H). A* PG-resolution tree of (G, H) on $\mathcal{T}$ *is a PG-resolution tree of (G, H) such that for each literal l in labels of nodes of $L(T)$, there is a triple $\{+p(u), -p(v), w\}$ in $\mathcal{T}$ such that l is either equal to $+p(w)$ or to $-p(w)$.*

These results extend to preordered predicates, where a triple w.r.t. (G, H) is in the form $\{-r(u), +s(v), w\}$ with $-r(u)$ and $+s(v)$ being weakly opposite literals in G, and the definition of an exchangeable triple is obtained from that of an exchangeable pair as above.

Note that, in Algorithm 4, $\mathcal{T}$ can be initialized with any superset of the given initialization set. In practice, we obtain a partially refined set of exchangeable triples by successively removing triples that can be recognized as non exchangeable. For instance, in the example of Figure 4, $\{l_1, l_2, d\}$ is clearly non exchangeable, and removing it makes $\{l_1, l_2, b\}$ clearly non exchangeable.

7 Related Works and Conclusion

Let us now relate the present complexity results to previous results obtained on the various forms of FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION.

Clause entailment. When the logical language includes function symbols, clause entailment is undecidable [SS88], even if both clauses are Horn-clauses (i.e. with at most one positive literal) [MP92]. In [Got87], a sufficient condition under which a "subsumption test" (which can be identified with a homomorphism check) is complete is exhibited. Translated into DEDUCTION, it says that if (1) h does not contain opposite literals, or (2) h is consistent and g does not contain opposite unifiable literals, then g can be deduced from h if and only if g can be mapped to h. On one hand, functions are allowed in this result, on the other hand if we exclude functions, we obtain particular cases of DEDUCTION$_0$. To the best of our knowledge, the Π_2^P-completeness of clause entailment for clauses without functions had not been pointed out.

Query containment. In database query languages, function symbols are naturally excluded. The undecidability of query containment for several kinds of Datalog programs/queries has long been shown (see [Shm87] for the first results). Concerning the specific case of conjunctive queries with negation, the Π_2^P-completeness of the containment problem is claimed in several papers and proven in [FNTU07][11], with a reduction from the validity problem of quantified boolean formulas in the form $\forall^*\exists^* conj$, where $conj$ is a conjunction of 3-clauses. It was also proven in the framework of polarized graphs by Bagan (2004), with a reduction from a graph problem called Generalized Ramsey Number [SU02] and this proof is reported in [Mug07] [CM08]. In [LM07], it is proven that homomorphism checking is sufficient when g has no *dependent* literals, i.e. opposite literals l_1 and l_2 s.t. l_1 and $\overline{l_2}$ can be unified after a renaming of their common variables. We obtain again a particular case of DEDUCTION$_0$. Notions close to our extensible homomorphism were used in algorithms for query containment checking in [WL03] and defined in [LM07].

As far as we know, the notion of exchangeable literals generalize all particular cases exhibited so far. As already mentioned, weaker criteria that yield an upper bound for the number of exchangeable pairs and can be checked in polynomial time can be used instead of exchangeability. In previous results, if the notion of an "exchangeable pair" is replaced by a "pair of opposite and unifiable literals", these results are weaker but on the other hand any pair of term nodes can be checked in constant time. With this weaker condition, all complexity results are still new, except for DEDUCTION$_0$.

Finally, let us mention that exchangeable literals can be exploited in algorithms solving DEDUCTION for general FOL$\{\exists, \wedge, \neg_a\}$ formulas. In [LM07] an algorithm is proposed for deciding inclusion of conjunctive queries with negation. Since queries are seen as PGs, this algorithm can be used without change for deciding on

[11]Bibliographical note: several database papers wrongly mention that [LS93] proves the Π_2^P-completeness of the query inclusion problem for conjunctive queries with negation. More precisely, the Π_2^P-completeness result reported in [LS93] is for "conjunctive queries with order constraints" (and this result is due to van der Meyden). However, there is no straightforward proof that would translate this result into one for conjunctive queries with negation.

deduction in FOL$\{\exists, \wedge, \neg_a\}$. It explores a space of graphs leading from H to its completions. This space is ordered as follows: given two graphs H_1 and H_2 in this space, $H_2 \leq H_1$ if H_1 is a subgraph of H_2. The question "is there a homomorphism from G to each completion H^c" is reformulated as "is there a *covering set* of completions, that is a subset of incomparable graphs of this space $\{H_1, \dots, H_k\}$ such that (1) there is a homomorphism from G to each H_i ; (2) for each H^c there is a H_i with $H^c \leq H_i$". Some special subgraphs of G, that are necessarily mapped to H if G is deducible from H, are used both in a filtering step (if one of these subgraphs cannot be mapped to H, then it can be concluded that G is not deducible from H) and to guide the space exploration. These subgraphs are without opposite literals. They can be replaced by subgraphs without exchangeable pairs (see Theorem 3). Moreover, the set of relation names considered in completions is restricted to relation names occurring both positively and negatively in G and H (see Property 3): this set can be further restricted to relation names occurring in exchangeable literals of G (Property 20), and the notion of completion can be further refined, using exchangeable triples.

In this paper, we have solved the main issues concerning the role of exchangeable literals in the complexity of FOL$\{\exists, \wedge, \neg_a\}$-DEDUCTION. We have shown that, as soon as the number of exchangeable pairs is bounded, the complexity falls into P^{NP}, and becomes even NP-complete if the bound is 1. However, to complete the picture, some open issues remain to be solved: Is DEDUCTION$_k$ complete for P^{NP}? What is the complexity of DEDUCTION$_2$?

References

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[ASU79] A. V. Aho, Y. Sagiv, and J. D. Ullman. Equivalences among relational expressions. *SIAM J. Comput.*, 8(2):218–246, 1979.

[CM77] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, pages 77–90, 1977.

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

[CM08] M. Chein and M.-L. Mugnier. *Graph-based Knowledge Representation and Reasoning—Computational Foundations of Conceptual Graphs*. Advanced Information and Knowledge Processing. Springer, 2008.

[FNTU07] C. Farré, W. Nutt, E. Teniente, and T. Urpí. Containment of conjunctive queries over databases with null values. In *ICDT*, pages 389–403, 2007.

[GLS01] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions: A survey. *MFCS'01*, 2136:37–57, 2001.

[Got87] G. Gottlob. Subsumption and implication. *Inf. Process. Lett.*, 24(2):109–111, 1987.

[Hal01] A. Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

[Ker01] G. Kerdiles. *Saying it with Pictures: a logical landscape of conceptual graphs*. PhD thesis, Univ. Montpellier II / Amsterdam, Nov. 2001.

[KV00] P. G. Kolaitis and M. Y. Vardi. Conjunctive-Query Containment and Constraint Satisfaction. *Journal of Computer and System Sciences*, 61:302–332, 2000.

[LM06] M. Leclère and M.-L. Mugnier. Simple conceptual graphs with atomic negation and difference. In *ICCS*, pages 331–345, 2006.

[LM07] M. Leclère and M.-L. Mugnier. Some algorithmic improvements for the containment problem of conjunctive queries with negation. In *ICDT*, pages 404–418, 2007.

[LS93] A. Y. Levy and Y. Sagiv. Queries independent of updates. In *VLDB*, pages 171–181, 1993.

[ML07] M.-L. Mugnier and M. Leclère. On querying simple conceptual graphs with negation. *Data Knowl. Eng.*, 60(3):468–493, 2007.

[MP92] J. Marcinkowski and L. Pacholski. Undecidability of the horn-clause implication problem. In *FOCS*, pages 354–362, 1992.

[MR94] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *J. Log. Program.*, 19/20:629–679, 1994.

[Mug07] M.-L. Mugnier. On the π_p^2-completeness of the containment problem of conjunctive queries with negation and other problems. Research Report 07004, LIRMM, 2007.

[Shm87] O. Shmueli. Decidability and expressiveness of logic queries. In *PODS*, pages 237–249, 1987.

[SS88] M. Schmidt-Schauß. Implication of clauses is undecidable. *Theor. Comput. Sci.*, 59:287–296, 1988.

[SU02] M. Schaefer and C. Umans. Completeness in the polynomial-time hierarchy: A compendium. Sigact News. Available on M. Schaefer's homepage, 2002.

[Ull89] J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume II*. Computer Science Press, 1989.

[WL03] F. Wei and G. Lausen. Containment of conjunctive queries with safe negation. In *ICDT*, pages 343–357, 2003.