On the Complexity of Entailment in Existential Conjunctive First Order Logic with Atomic Negation

Marie-Laure Mugnier, Geneviève Simonet, Michaël Thomazo

On the Complexity of Entailment in Existential Conjunctive First Order Logic with Atomic Negation

Marie-Laure Mugnier* †
University Montpellier II
mugnier@lirmm.fr

Geneviève Simonet*
University Montpellier II
simonet@lirmm.fr

Michaël Thomazo* †
University Montpellier II
thomazo@lirmm.fr

March 2011

LIRMM Research Report RR-11026

Abstract

We consider the entailment 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. Entailment 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 shall 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 ENTAILMENT$_k$ 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, is g entailed by h? The main

*LIRMM (CNRS: UMR5506, University Montpellier II), France
†GraphIK (INRIA Sophia Antipolis)

results are that ENTAILMENT$_k$ is NP-complete if k is less or equal to 1, and $P_{||}^{NP}$-complete for any value of k greater or equal to 3. As a corollary of our proofs, we are able to classify exactly previous problems when g is decomposable into a tree.

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

Contents

1 Introduction

In this paper, we study the complexity of checking entailment 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 (where $\neg_a$ stands for atomic negation, i.e., negation whose scope is an atom), and FOL$(\exists, \wedge)$ is the subfragment with positive literals only.

The ENTAILMENT problem in a given fragment takes two formulas g and h of this fragment as input, and asks if g is entailed by h.

Equivalent problems. FOL$(\exists, \wedge, \neg_a)$-ENTAILMENT 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 independence 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$(\exists, \wedge, \neg_a)$-ENTAILMENT (resp. FOL$(\exists, \wedge)$–ENTAILMENT), in the sense that there is a natural bijection from the set of conjunctive queries with negation (resp. positive conjunctive queries) on a given database schema to the set of FOL$(\exists, \wedge, \neg_a)$ (resp. FOL$(\exists, \wedge)$) formulas on the logical language corresponding to this schema, such that query containment coincides with logical entailment. 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$(\exists, \wedge, \neg_a)$-ENTAILMENT. 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 is entailed by the knowledge base. In the case where the knowledge base is simply composed of a set of positive and negative facts, i.e., ground literals or existentially closed conjunctions of literals[1], and the query is a Boolean conjunctive query with negation, we obtain

[1] In the literature, a fact is usually assumed to be a ground literal. By extending this notion to existentially closed conjunctions of literals, we naturally cover languages such as the basic semantic web language RDF [W3C04], dedicated to the description of web resources, where the so-called

FOL$(\exists, \wedge, \neg_a)$-ENTAILMENT. Let us point out that this definition of the query answering problem is consistent with the so-called open-world assumption (OWA), which assumes incomplete knowledge about the represented world. This assumption is commonly made in knowledge representation and reasoning. The opposite assumption, closed-world assumption (CWA), commonly made in databases, assumes complete knowledge about the represented world. It follows that only positive facts (the data) need to be encoded, with negative facts being obtained by difference with the content of the fact base. Then, negation occurs only in queries and is interpreted as the absence of a positive fact, i.e., $\neg p(a_1 \dots a_l)$ holds if $p(a_1 \dots a_l)$ is *not* entailed by the fact base (while with OWA $\neg p(a_1 \dots a_l)$ holds if it is entailed by the fact base). Note however that the query containment problem for conjunctive queries with negation is the same regardless of the assumption made (e.g. [LM07]).

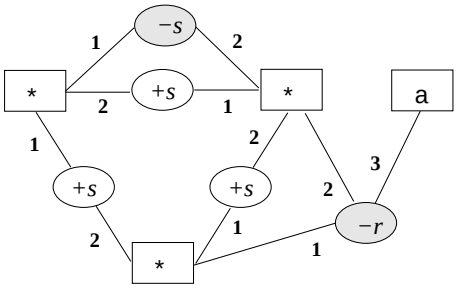
Finally, even if this aspect is out of the scope of the present paper, let us mention that a partial order on predicates, or more generally a preorder, can be taken into account without increasing complexity. This allows to represent a terminology where concepts and relations are preordered by a subsumption relation. These concepts and relations are logically translated into a set of predicates used to build facts. We then obtain FOL$(\exists, \wedge, \neg_a)$-ENTAILMENT extended to preordered predicates, which is exactly the entailment problem in a fragment of conceptual graphs, called *polarized conceptual graphs* [Ker01][ML07].

Complexity and "exchangeable" literals. Whereas FOL$(\exists, \wedge)$-ENTAILMENT is "only" NP-complete, FOL$(\exists, \wedge, \neg_a)$-ENTAILMENT is Π_2^P-complete[2] [FNTU07] [Mug07]. Some specific cases where FOL$(\exists, \wedge, \neg_a)$-ENTAILMENT has a lower complexity are known but they enforce strong restrictions on the problem instances: briefly said, if g does not contain any pair of opposite and unifiable literals[3], then FOL$(\exists, \wedge, \neg_a)$-ENTAILMENT becomes NP-complete (see Section 6). The aim of this paper is to investigate the complexity gap between entailment 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

"blank nodes" are logically translated into existential variables, or fragments of conceptual graphs (see hereafter). This extension has no incidence on the complexity of the problems we consider.

[2] Π_2^P is *co*-(NP^{NP}).

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



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

Figure 1: A polarized graph

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 from [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 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$(\exists, \wedge)$-ENTAILMENT can be solved

by such 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 entailment in FOL$(\exists, \wedge, \neg_a)$, i.e., the existence of a homomorphism from G to H implies that g is entailed by h, but of course it is no longer complete, i.e., g may be entailed by h even if there is no homomorphism from G to H.

FOL$(\exists, \wedge, \neg_a)$-ENTAILMENT can be recast as a problem on PGs involving a number of homomorphism checks exponential in the size of H. Indeed, 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, i.e., for any formula A, $(A \vee \neg A)$ is valid. This leads to reasoning by cases: if nothing is known about $p(u)$, then either $p(u)$ or $\neg p(u)$ holds. We are thus led to consider all possible ways of "completing" H with missing relation nodes (while keeping it consistent) and to check if G can be mapped by homomorphism to all these completions of H. Intuitively, exchangeable literals are literals from G that may lead to use the law of the excluded-middle. More precisely, exchangeable literals are literals of the form $p(u)$ and $\neg p(v)$ respectively, such that u and v can be mapped "at the same place" by homomorphisms from G to (necessarily distinct) completions of H.

Finally, let us come back to query answering and the distinction between OWA and CWA. With CWA, H can be seen as implicitly completed with solely negative relation nodes; then, G is CWA-entailed by H if and only if there is a homomorphism from G to this negative completion of H (which can be checked without effectively computing this completion). It follows that, with CWA, answering a conjunctive query with negation is not more complex than answering a positive conjunctive query.

Contributions of the paper. The results achieved in this paper can be summarized as follows. Please note that we make the assumption that the arity of predicates is bounded by a constant. This assumption is commonly made in knowledge representation, but not necessarily in databases. We first point out that if g has *no* pair of exchangeable literals, then FOL$(\exists, \wedge, \neg_a)$-ENTAILMENT 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 entailment checking decreases when g has a *bounded* number of exchangeable literals. Let ENTAILMENT$_k$ 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, is g entailed by h? It is proven that, for any $k \geq 3$, ENTAILMENT$_k$ is $P_{||}^{NP}$-complete. When g represents a query and h a

Number of exchangeable pairs in g	**General g and h**	**Homomorphism check polynomial**
unbounded	Π_2^P-complete (*)	co-NP-complete
0	NP-complete	P
1 (**)	NP-complete	P
bounded by $k \geq 3$	$P_{\|\|}^{NP}$-complete	co-NP-complete

(*) already known result

(**) the same complexity holds if g has an unbounded number of exchangeable pairs that all have the same positive (resp. negative) literal

Table 1: Main complexity results

base of facts, criteria that decrease the complexity and depend on g rather than h are specially 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). Of course, these criteria are also relevant when g and h are both queries. In particular, when g has a structure decomposable into a tree (we will precise this point later), then checking if there is a homomorphism from g to h can be done in polynomial time. In this case, we point out that FOL$(\exists, \wedge, \neg_a)$-ENTAILMENT is co-NP-complete; moreover, a corollary of previous proofs is that ENTAILMENT$_k$ remains co-NP-complete for any $k \geq 3$ and is in P if $k \leq 1$.

Table 1 summarizes the complexity results. The recognition problem associated with ENTAILMENT$_k$, 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.

Finally, these results are extended in two ways. 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 corresponds 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". Second, we refine several notions related to exchangeable literals, in order to 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 is devoted to refinements. Section 6 synthe-

sizes related work and concludes this study.

2 Preliminaries

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. 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). An *atom* on $(\mathcal{R}, \mathcal{I})$ is of form $p(t_1, \ldots, t_n)$, $n \geq 1$, where $p \in \mathcal{R}$ and, for all j in $1, \ldots, n$, t_j is a term on $(\mathcal{R}, \mathcal{I})$. Note that nullary predicates are not considered because their processing is trivial; the tools developed here would therefore be unnecessarily complicated for dealing with them. A *literal* on $(\mathcal{R}, \mathcal{I})$ is an atom (positive literal) or the negation of an atom (negative literal) on $(\mathcal{R}, \mathcal{I})$. A FOL$(\exists, \wedge, \neg_a)$ formula on $(\mathcal{R}, \mathcal{I})$ is an existentially closed conjunction of literals on $(\mathcal{R}, \mathcal{I})$. Without loss of generality, we consider that it is of the form $\exists x_1 \ldots x_q (l_1 \wedge \ldots \wedge l_p)$, where, for all i in $1 \ldots p$, l_i is a literal whose variables are in $\{x_1, \ldots, x_q\}$. A FOL$(\exists, \wedge)$ formula has only positive literals. The set of *atoms occurring in 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 $\mathcal{V} = (\mathcal{R}, \mathcal{I})$ be a vocabulary 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 $G = (R,\ T,\ E,\ \lambda)$ 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 λ is a labeling mapping of nodes and edges. For $x \in R$, $\lambda(x) = +r$ (x is called a positive relation node) or $\lambda(x) = -r$ (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 $\lambda(t) = *$ (t is called a variable node) or $\lambda(t) \in \mathcal{I}$ (t is called a constant node).*

Each PG can be put into a *normal* form, such that each constant of $\mathcal{I}$ appears at

most once in it. In the following, a PG is assumed to be in this normal form unless otherwise specified.

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[4] on a vocabulary $\mathcal{V} = (\mathcal{R}, \mathcal{I})$. This bijection is up to isomorphism for graphs and up to 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. Moreover, we will assume that PGs do not have redundant relation nodes (i.e., with the same label and the same ith neighbors), thus the associated formulas can be seen as sets of atoms.

Notations. Let $+r(t_1, \ldots, t_q)$ (resp. $-r(t_1, \ldots, t_q)$) denote the subgraph induced by a positive (resp. negative) relation node with label $+r$ (resp. $-r$) and its list of neighbors $t_1, \ldots, t_q$. By analogy with its logical translation $r(t_1, \ldots, t_q)$ (resp. $\neg r(t_1, \ldots, t_q)$), in which t_i denotes the term assigned to the term node t_i, we also call it a *literal*. Let $\sim r$ denote a label with relation name r, where $\sim$ can be $+$ or $-$. Given a literal (resp. a relation label) l, $\bar{l}$ 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 $(t_1, \ldots, t_q)$ of terms (or term nodes). Thus $\sim r(u)$ denotes a literal of arbitrary sign and arity. If π is a mapping from a set of terms (or term nodes) to a set of terms (or term nodes), then for $u = (t_1, \ldots, t_q)$, $\pi(u)$ denotes the tuple $(\pi(t_1), \ldots, \pi(t_q))$. 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 and the edges incident to it, 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 (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*

[4] 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.

it is said to be consistent.

It can be immediately checked that inconsistent PGs correspond to unsatisfiable formulas.

Definition 3 (PG homomorphism) *A PG homomorphism from $G = (R_G,\ T_G,\ E_G,\ l_G)$ to $H = (R_H,\ T_H,\ E_H,\ l_H)$, over the same vocabulary $\mathcal{V} = (\mathcal{R}, \mathcal{I})$, is a mapping π 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*)*
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. We call G 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.

Proposition 1 (Substitution / PG Homomorphism Equivalence) *Let G and H be two PGs without isolated term nodes (with H being normal). 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 $\Phi(H)$.*

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 entailment, provided that the target graph is normal (basically [CM92], considering that positive PGs are a particular case of simple conceptual graphs). For general PGs, homomorphism is still sound:

Proposition 2 *Given two PGs G and H, if there is a homomorphism from G to H then $\Phi(G)$ is entailed by $\Phi(H)$.*

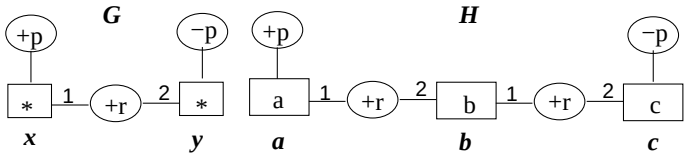


Figure 2: Non-completeness of PG homomorphism

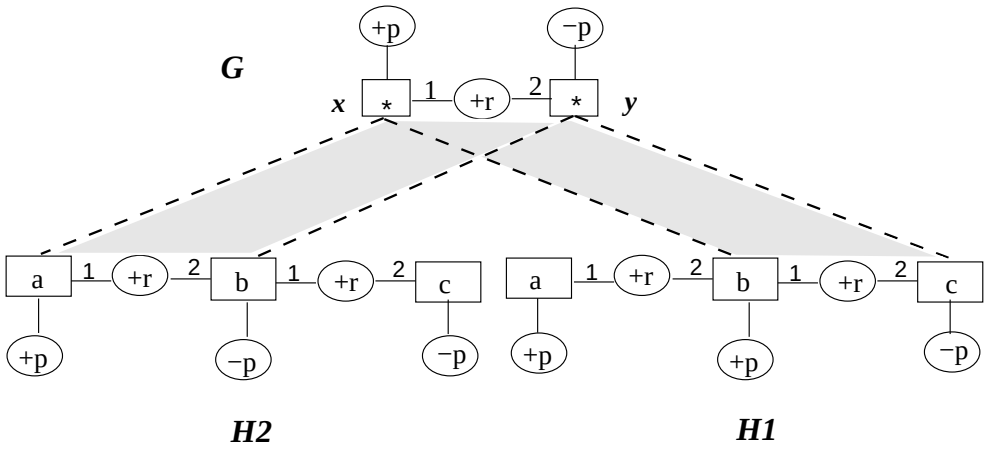


Figure 3: When the law of the excluded-middle intervenes

However, it 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 $\Phi(H) = p(a) \wedge r(a,b) \wedge r(b,c) \wedge \neg p(c)$. One can check that $\Phi(G)$ is entailed by $\Phi(H)$, 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 $\Phi(G)$).

As explained in the introduction, the law of the excluded-middle leads 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 is entailed by 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*

q, for each q-tuple of not necessarily distinct term nodes $(t_1, \ldots, t_q)$, it contains $+r(t_1, \ldots, t_q)$ or $-r(t_1, \ldots, t_q)$. 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 entailment problem on PGs in terms of completion.

Definition 5 (PG-ENTAILMENT) PG-ENTAILMENT *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 is PG-entailed by H, i.e., whether G can be mapped via homomorphism to each completion of H w.r.t. $\mathcal{R}_\mathcal{V}$.*

The following theorem expresses that PG-ENTAILMENT is sound and complete with respect to FOL entailment.

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

In the rest of the paper, we will thus not distinguish between logical entailment in the FOL$(\exists, \wedge, \neg_a)$ fragment and PG-entailment, and use the expression "G is entailed by H".

Let us outline a brute-force algorithm scheme for PG-ENTAILMENT: 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 entailed by H. Actually, not all relation names occurring in G need to be considered for completing H:

Proposition 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 entailed by 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).

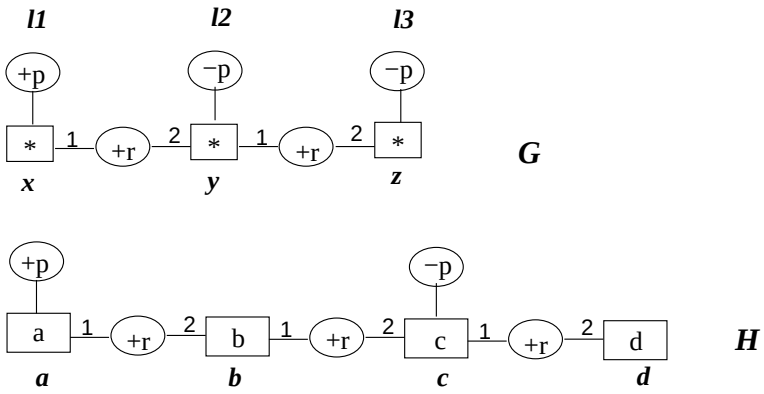


Figure 4: Exchangeable versus unifiable literals

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 (regardless of their arguments). Two opposite literals of G are said to be "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[5].

See for instance G in Figure 2. The pair $\{+p(x), -p(y)\}$ of opposite literals in G 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

[5]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.

the notion of exchangeable pair takes both the structure of G and and the one of 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 x and z can be mapped to the same node [6].

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^H (and simply G_s if not ambiguous), is the subgraph of G obtained from G by removing all exchangeable literals.*

We recall that removing a literal means removing its relation node and its incident edges. 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 entailed by 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 entailed by 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 some 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 entailed by H). Let us show that π is a

[6] The restriction to relation names of the completion vocabulary (see Prop. 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.

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. □

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:

Proposition 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 entailed by 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 proposition for G^- and H^{c+} (the proof for G^+ and H^{c-} is symmetric). If G is entailed by H, Th. 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. □

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 entailed by 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 entailed by 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 some 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 entailed by 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. □

Th. 3 can be rephrased as follows: if G is entailed by 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 will now define this notion of "extensible homomorphism" from a subgraph of G to H (Def. 9). We first restrict the subgraphs of interest to "completion subgraphs":

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

In the following, we will consider completion subgraphs of G without exchangeable pairs. Note that G_s is such a subgraph; it is not necessarily the smallest with this property as it may still contain literals with relation names from the completion vocabulary.

Definition 9 (Extensible homomorphism) *A homomorphism π from a completion 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 completion subgraph of G, G' contains all term nodes of G, so $\pi(u)$ is defined for any literal $\sim r(u)$ in $G \setminus G'$. Conditions 1 and 2 are obviously necessary for π to be extendable to a homomorphism from G to a completion of H. The next proposition shows that they are also sufficient.

Proposition 5 *A homomorphism π from a completion 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. $\Leftarrow$: Obvious. $\Rightarrow$: 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 completion 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 Th. 3 and Prop. 5.

Corollary 1 *Let G and H be two PGs, with H being consistent. Let G' be a completion subgraph of G possessing no exchangeable pair w.r.t. (G, H). If G is entailed by H, then there is an extensible homomorphism from G' to H.*

The previous properties provide necessary entailment conditions, and therefore sufficient non-entailment conditions. For instance, by Corollary 1, if we find a completion 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 entailed by H.

The problem of checking whether there is an extensible homomorphism from G' to H (given PGs G and H and a completion 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[7] 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.

ENTAILMENT$_k$
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 entailed by H?

For any integers k and k' such that $k < k'$, ENTAILMENT$_{k'}$ is at least as difficult as ENTAILMENT$_k$, since any graph G possessing at most k exchangeable pairs also possesses at most k' exchangeable pairs. For the following results, we recall that we assume that the arity of predicates is bounded by a constant.

[7]The NP-hardness of this problem can be easily checked, for instance with a straightforward reduction from the Clique problem [GJ79]; indeed, a classical undirected graph (which can be turned into a special PG) contains a k-clique if and only if there is a homomorphism from the k-clique to it.

4.1 Complexity of the Recognition Problem

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

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

Proof: EXCHANGEABLE is in NP: a polynomial certificate is given by a pair $\{+p(u), -p(v)\}$ of literals in G, and the proof that it is exchangeable, i.e., two completions H_1 and H_2 of H with homomorphisms π_1 from G to H_1 and π_2 from G to H_2 such that $\pi_1(u) = \pi_2(v)$. 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). Consider, for instance, the graphs G and H in Figure 2, and choose relation names r and p, as well as the constants a, b and c, such that they do not occur in G_1 and G_2. G'_1 (resp. G'_2) is obtained by making the disjoint sum[8] 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 ENTAILMENT$_k$, is co-NP-complete.

Proposition 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 for any $k \geq 0$.*

The complexity of the recognition problem associated with ENTAILMENT$_k$ may be seen as restricting practical use of the results in this paper. However, 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, Th. 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.

[8] 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.

4.2 ENTAILMENT$_0$ and ENTAILMENT$_1$

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

Proposition 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 entailed by H if and only if there is a homomorphism from G to H.*

Proof: If there is a homomorphism from G to H then G is entailed by H by Prop. 2. The converse follows from Th. 2 since $G_s = G$ (or from Th. 3 with $G' = G$). □

Proposition 9 *The problem* ENTAILMENT$_0$ *is NP-complete.*

It follows that ENTAILMENT$_1$ is NP-hard. We will now prove that ENTAILMENT$_1$ 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 entailed by $H_1 = H + \{+p(b)\}$. 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 entailed by $H_2 = H_1 + \{+p(c)\}$, 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 $G \setminus \{+p(x)\}$. We are sure that this sliding process will either succeed or stop by lack of homomorphism after a finite number of steps since H cannot grow infinitely.

These ideas directly lead to Algorithm 1. Note that $G \setminus \{\sim p(u)\}$ is a completion subgraph without exchangeable pair. Thus, if there is no extensible homomorphism from $G \setminus \{\sim p(u)\}$ to H, then G is not entailed by H; otherwise, let π be such a homomorphism: either $\sim p(\pi(u))$ is in H and there is a homomorphism from G to H, or, noticing that G is entailed by $H + \{\sim p(\pi(u))\}$, it remains to check that G is entailed by $H + \{\overline{\sim p}(\pi(u))\}$, hence the recursive call.

Proposition 10 *The algorithm* ENTAILMENT$_1$ *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 $\sim p(u)$ still holds. It is indeed the case, since any exchangeable pair w.r.t. $(G, H +$

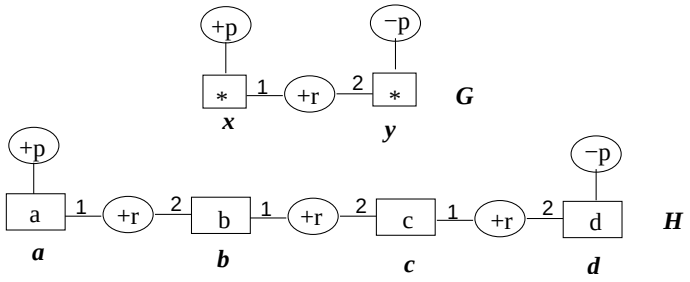


Figure 5: Illustration of Algorithm 1

Algorithm 1: ENTAILMENT$_1$

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 any literal in G such that relation name p belongs to the completion vocabulary w.r.t. (G, H).

Result: true if G is entailed by 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 ENTAILMENT_1(G, H + {¬∼p(π(u))}, ∼p(u))
end
```

$\{\overline{\sim p}(\pi(u))\}$) 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[9]), 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 ENTAILMENT$_1(G, H, \sim$

[9]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.

$p(u)$) returns true if G is entailed by 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 entailed by H) and $\text{ENTAILMENT}_1(G, H, \sim p(u))$ 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 entailed by H) and $\text{ENTAILMENT}_1(G, H, \sim p(u))$ 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, $\text{ENTAILMENT}_1(G, H, \sim p(u))$ returns true iff $\text{ENTAILMENT}_1(G, H + \{\overline{\sim p}(\pi(u))\}, \sim p(u))$ returns true, i.e., by induction hypothesis, iff G is entailed by $H + \{\overline{\sim p}(\pi(u))\}$. It remains to show that G is entailed by H iff G is entailed by $H + \{\overline{\sim p}(\pi(u))\}$. If G is entailed by H then G is entailed by $H + \{\overline{\sim p}(\pi(u))\}$ since every completion of $H + \{\overline{\sim p}(\pi(u))\}$ is a completion of H. Conversely, we suppose that G is entailed by $H + \{\overline{\sim p}(\pi(u))\}$. 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 entailed by H. □

The following proposition immediately follows from Algorithm 1.

Proposition 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 entailed by H if and only if there is a sequence $(\pi_i)_{i \in 1,\ldots,m}$ such that:*

1. *π_1 is an extensible homomorphism from $G \setminus \{\sim p(u)\}$ to $H_1 = H$*
2. *$\forall i \in 2, \ldots, m-1$,*
   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 ENTAILMENT_1.

Theorem 4 *The problem ENTAILMENT_1 is NP-complete.*

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

Note that Algorithm 1 still holds if G has an unbounded number of exchangeable pairs but only one positive (resp. negative) literal. It follows that the entailment problem remains NP-complete in that case. In contrast, the technique used in this algorithm does not seem to be generalizable to $k \geq 2$. Take for instance the case where $k = 2$ and try to generalize Algorithm 1, replacing the literal $\sim p(u)$ by two literals $\sim p(u)$ and $\sim q(v)$. Then the recursive call with input $H + \{\overline{\sim p}(\pi(u))\}$ would be replaced by the conjunction of three recursive calls with inputs $H + \{\overline{\sim p}(\pi(u)), \sim q(\pi(v))\}$, $H + \{\sim p(\pi(u)), \overline{\sim q}(\pi(v))\}$ and $H + \{\overline{\sim p}(\pi(u)), \overline{\sim q}(\pi(v))\}$ respectively, each of these recursive calls potentially generating three new recursive calls etc, so that generalized Prop. 11 would contain an exponential number of PGs H_i and homomorphisms π_i.

4.3 ENTAILMENT$_k$

We now show that, for any value of parameter k, ENTAILMENT$_k$ falls into the class P^{NP}, and even $P^{NP}_{||}$, i.e., the class of decision problems solvable in polynomial time with one round of parallel queries to an NP oracle. Note that the condition on parallel queries can be relaxed by considering a constant number of rounds of parallel queries instead of a single round [BH91].

For that, we rely on Th. 2. We first deduce from this theorem a necessary and sufficient entailment condition (Prop. 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 is entailed (i.e., G can be mapped to $H + \{+p(b)\}$), and if $p(b)$ is known to be false then G is entailed 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 formula $p(b) \vee \neg p(b)$ being a tautology. We see $p(b) \vee \neg p(b)$ as a propositional formula on a propositional language containing the atom $p(b)$; if b was a variable node associated with variable z, the propositional language would contain the atom $p(z)$ and the propositional tautology would be $p(z) \vee \neg p(z)$. Similarly, for the graphs in Figure 5, there are three extensible homomorphisms π_1, π_2 and π_3 from G_s to H, which map G_s to $+r(a, b)$, $+r(b, c)$ and $+r(c, d)$ respectively, and 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 completion subgraph G' of G contained in G_s to H a propositional formula that is a tautology if and only if G is entailed by H.

We define for each completion subgraph G' of G and each extensible homo-

morphism π from G' to H the set $L(\pi)$ of literals that are "missing" in H for π to be extendable to a homomorphism from G to H. Therefore, the literals from $L(\pi)$ have to be in any completion H^c of H such that π can be extended to a homomorphism from G to H^c. From $L(\pi)$, we define propositional formulas $C(\pi)$ and $D_{G'}(G, H)$ on a propositional language denoted P_H.

Notations 1 *Let G and H be two PGs, with H being consistent, and let G' be a completion subgraph of G.*
P_H denotes the set of atoms occurring in $\Phi(H^c \setminus H)$, where H^c is an arbitrary completion of H.
For any extensible homomorphism π from G' to H, $L(\pi)$ 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 $C(\pi)$ denotes the conjunction of the literals in $L(\pi)$ which is a proposition on P_H.
$D_{G'}(G, H)$ denotes the disjunction of the propositions $C(\pi)$ 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 $G' = G_s$: let π_1, π_2 and π_3 be the extensible homomorphisms from G_s to H; $L(\pi_1) = \{-p(b)\}$, $L(\pi_2) = \{+p(b), -p(c)\}$, $L(\pi_3) = \{+p(c)\}$, $C(\pi_1) = \neg p(b)$, $C(\pi_2) = p(b) \wedge \neg p(c)$ and $C(\pi_3) = p(c)$; finally, $D(G, H) = \neg p(b) \vee (p(b) \wedge \neg p(c)) \vee p(c)$.

Next Lemma 1 follows immediately from the definition of $L(\pi)$.

Lemma 1 *Let G and H be two PGs, let H^c be a completion of H, let G' be a completion subgraph of G, and let π be an extensible homomorphism from G' to H. Then π can be extended to a homomorphism from G to H^c if and only if $L(\pi)$ 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 completion subgraph G' of G and any extensible homomorphism π from G' to H, $L(\pi)$ is a set of literals in H^c if and only if $f(H^c)$ satisfies $C(\pi)$.*

Proof: Let f be the mapping from the set of completions of H to the set of truth assignments on P_H defined as follows: 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. □

Proposition 12 *Let G and H be two PGs, with H being consistent, and let G' be any completion subgraph of G contained in G_s. Then G is entailed by H if and only if $D_{G'}(G, H)$ is a tautology.*

Proof: By Th. 2 (since G' is contained in G_s) and Prop. 5 (since G' is a completion subgraph of G), G is entailed by 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. Let us show that the latter proposition holds iff $D_{G'}(G, H)$ is a tautology, using the bijection f of Lemma 2. $\Rightarrow$: We suppose that 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. Let us show that $D_{G'}(G, H)$ is a tautology. Let v be a truth assignment on P_H, let us show that v satisfies $D_{G'}(G, H)$. Let $H^c = f^{-1}(v)$, and let π be an extensible homomorphism from G' to H that can be extended to a homomorphism from G to H^c. By Lemma 1, $L(\pi)$ is a set of literals in H^c, so by Lemma 2, v satisfies $C(\pi)$, and therefore $D_{G'}(G, H)$. $\Leftarrow$: We suppose that $D_{G'}(G, H)$ is a tautology. Let H^c be a completion of H, let us show that there is an extensible homomorphism from G' to H that can be extended to a homomorphism from G to H^c. Let $v = f(H^c)$. As $D_{G'}(G, H)$ is a tautology, there is an extensible homomorphism π from G' to H such that v satisfies $C(\pi)$. By Lemmas 1 and 2, π can be extended to a homomorphism from G to H^c. □

In order to prove that ENTAILMENT$_k$ 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 to an extensible homomorphism from G_s to H. Note that 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 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. According to Prop. 12, Algorithm 2 computes $D(G, H)$ to determine whether G is entailed by H.

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 ENTAILMENT$_k$ fall into P^{NP}.

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

Data: G and H two PGs, such that H is consistent
Result: true if G is entailed by 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

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

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 fixed number of rounds of parallel calls to an NP oracle. This is indeed the case with three rounds of parallel calls since:
- to compute $\mathcal{E}$, it is sufficient to determine for each pair of opposite literals in G (whose number is polynomial) if it is exchangeable, which is in NP (first round),
- $|\mathcal{T}_\mathcal{E}| \leq 2kw$, where w 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 $(n_H)^{2kw}$, 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, since an extension provides a polynomial certificate (second round),
- determining if a proposition is not a tautology is in NP (third round). □

It follows from Algorithm 2 that in any case where it can be decided in polynomial time whether the formula Φ computed by this algorithm is a tautology, the entailment problem is in NP. A polynomial certificate is given by a set of extensible homomorphisms π from G to H extending the mappings φ considered in this algorithm (one for each extendable mapping φ, so that the number of homomorphisms π is polynomial), since computing the disjunction of the formulas $C(\pi)$ and checking that it is a tautology can be done in polynomial time (we do not need to compute the set $\mathcal{E}$ of exchangeable literals nor G_s nor the mappings φ themselves). In particular, it can be decided in polynomial time whether a disjunction

of conjunctions of literals in which each conjunction contains at most one positive literal (or each conjunction contains at most one negative literal) is a tautology. It follows that ENTAILMENT$_1$ is in NP, which provides a new proof of Th. 4 and of the fact that the entailment problem remains NP-complete if G has an unbounded number of exchangeable pairs but only one positive (resp. negative) literal.

4.4 ENTAILMENT$_3$

We first prove that ENTAILMENT$_3$ is co-NP-hard with a reduction from "3-DNF Tautology". This reduction will be reused to prove the $P_{||}^{NP}$-hardness of ENTAILMENT$_3$.

Theorem 6 *The problem* ENTAILMENT$_3$ *is co-NP-hard.*

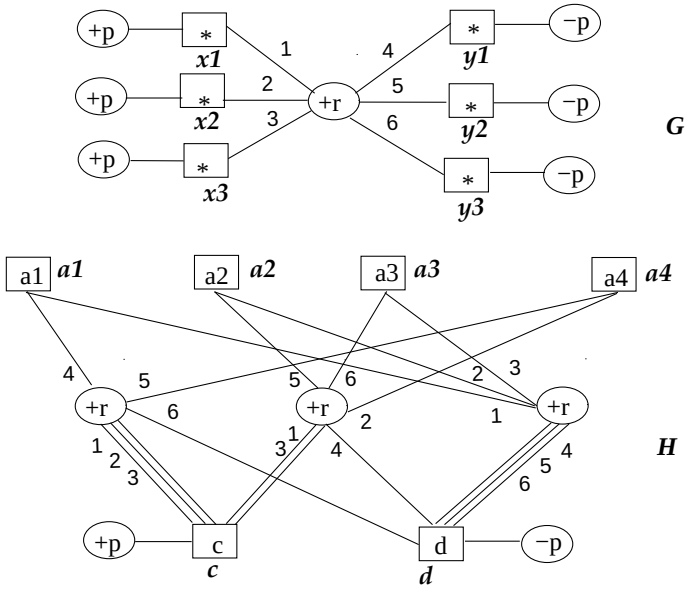
Proof: To prove that ENTAILMENT$_3$ is co-NP-hard, we define a reduction from the co-NP-complete problem 3-DNF Tautology to ENTAILMENT$_3$.
3-DNF Tautology
Input: a 3-DNF propositional formula Φ, i.e., a proposition Φ in disjunctive normal form (disjunction of conjunctions of literals) such that each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

The reduction uses Prop. 12. Let Φ be a 3-DNF proposition. By Prop. 12, it is sufficient to build two PGs G and H in polynomial time, with H being consistent and containing at most 3 exchangeable pairs, such that for some completion 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 *exchange-reducing* mapping w.r.t. Φ (standing for "mapping allowing to reduce the number of exchangeable pairs in the graph G built by the reduction"). We will build a graph G with 3 positive literals and 3 negative literals with relation name p. Using an exchange-reducing mapping in the construction of graph H will make each positive literal $+p(u)$ in G be potentially exchangeable with only one negative literal, which reduces the number of potential exchangeable pairs from 9 to 3. This will be explained in the last paragraph of this proof.
Let P be the set of atoms occurring in Φ. A mapping α from P to $\{1, 2, 3\}$ is said to be *exchange-reducing* (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 exchange-reducing. Note that there may be no

exchange-reducing 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 an exchange-reducing 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 an exchange-reducing 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 an exchange-reducing mapping w.r.t. Φ, such that for some completion 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 all $p_1, \ldots, p_h$. It follows that Φ' is a tautology iff it is satisfied by each truth assignment assigning the same truth value to $p_1, \ldots, p_h$ for each atom p in P_H. Thus Φ' is a tautology iff Φ is.
An exchange-reducing 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 exchange-reducing w.r.t. Φ'.

2. Construction of G and H
Let Φ be a 3-DNF formula and α be an exchange-reducing 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 $1, \ldots, 3$, $+p(x_i)$ 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



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

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

$a_1, \ldots, a_h$, c and d, and it has $q+2$ literals: $+p(c)$, $-p(d)$ and, for all i in $1, \ldots, q$, $+r(u_i)$, 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 $1, \ldots, q$ and all j in $1, \ldots, 3$:
- 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 exchange-reducing) 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 exchange-reducing) 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 completion subgraph of G contained in G_s. Let us show that $D_{G'}(G, H)$ is a tautology iff Φ is. There are exactly q extensible homomorphisms $\pi_1, \ldots, \pi_q$

from G' to H such that for all i in $1, \ldots, q$, π_i maps G' to the literal $+r(u_i)$ and $C(\pi_i)$ is the formula obtained from C_i by replacing each atom p_j by $p(a_j)$. It follows that $D_{G'}(G, H)$ is obtained from Φ by replacing each atom p_j by $p(a_j)$. For instance, in the example of Figure 6, there are 3 extensible homomorphisms from G' to H, and $D_{G'}(G, H) = (\neg p(a_1) \wedge \neg p(a_4)) \vee (p(a_4) \wedge \neg p(a_2) \wedge \neg p(a_3)) \vee (p(a_1) \wedge p(a_2) \wedge p(a_3))$. Hence $D_{G'}(G, H)$ 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 $1, \ldots, 3$. However, if x_i and y_j are mapped to the same node z in H by two homomorphisms from G to completions of H, then there is an integer k in $1, \ldots, h$ such that z is labeled a_k, with $i = j = \alpha(p_k)$. Thus, each exchangeable pair must be of the form $\{+p(x_i), -p(y_i)\}$, with i in $1, \ldots, 3$. As announced at the beginning of this proof, using an exchange-reducing mapping w.r.t. Φ to define H allows to bound the number of exchangeable pairs to 3 instead of 9. □

Theorem 7 *The problem* ENTAILMENT$_3$ *is* $P_{||}^{NP}$*-hard.*

To prove this claim, we will rely on the following lemmas.

Lemma 3 *For any problem A in NP, there is a translation f mapping every instance I of A to an instance $f(I) = (f_G(I), f_H(I))$ of* ENTAILMENT *such that:*

- *$f_G(I)$ is entailed by $f_H(I)$ if and only if I is a positive instance of A,*
- *$f_G(I)$ and $f_H(I)$ do not contain any negative literal,*
- *$f_G(I)$ and $f_H(I)$ do not contain any constant node.*

Proof: As ENTAILMENT on PGs that do not contain any negative literal is NP-complete, there is a translation f satisfying the two first conditions on $f(I)$. In order to satisfy the third condition, we modify $G = f_G(I)$ and $H = f_H(I)$ as follows: for each constant a appearing in G or in H, replace the constant node labeled a in G (respectively H) by a variable node x and add the literal $+p_a(x)$ to G (respectively H), where p_a is a new unary relation name, i.e., that does not occur in G nor in H. □

Lemma 4 *There is a PG G and a set Q of 3 pairs of opposite literals in G such that for any problem B in co-NP, there is a translation g mapping every instance J of B to an instance $g(J) = (g_G(J), g_H(J))$ of* ENTAILMENT$_3$ *such that:*

- *$g_G(J)$ is entailed by $g_H(J)$ if and only if J is a positive instance of B,*

- *$g_G(J) = G$, each exchangeable pair w.r.t. $(G, g_H(J))$ is in Q, and the set of relation node labels in $g_H(J)$ is the same as in G,*
- *G and $g_H(J)$ do not contain any constant node.*

Proof: As 3-DNF Tautology is co-NP-complete, it is sufficient to prove the existence of the translation g in the case where B is 3-DNF Tautology. In that case it is sufficient to define the translation g as in the proof of Th. 6, except that the term nodes of H are defined as variable nodes instead of constant nodes, with $Q = \{(+p(x_i), -p(y_i)),\ 1 \leq i \leq 3\}$, the set of relation node labels being equal to $\{+p, -p, +r\}$ in $g_H(J)$ and in G. □

Proof: [of Theorem 7] We build a reduction from the following problem, known to be $P_{||}^{NP}$-complete [SV00]:
Min-card-vertex cover compare
Input: two undirected graphs $F_1 = (V_1, E_1)$ and $F_2 = (V_2, E_2)$.
Question: Does $min\text{-}vc(F_1) \leq min\text{-}vc(F_2)$, where $min\text{-}vc(F_i)$ denotes the minimum cardinality of a vertex cover[10] of F_i?

Let (F_1, F_2) be an instance of Min-card-vertex cover compare. We have to build two PGs G' and H' such that (1) G' has at most 3 exchangeable pairs w.r.t. (G', H') and (2) $min\text{-}vc(F_1) \leq min\text{-}vc(F_2)$ if and only if G' is entailed by H'.
Let i be an integer. Since deciding whether the minimum size of a vertex cover of F_1 is less than i is in NP, from Lemma 3, there is an instance of ENTAILMENT $f(F_1, i) = (f_G(F_1, i), f_H(F_1, i))$ such that $min\text{-}vc(F_1) \leq i$ iff $f_G(F_1, i)$ is entailed by $f_H(F_1, i)$, and $f_G(F_1, i)$ and $f_H(F_1, i)$ do not contain any negative literal or constant node. Similarly, since deciding whether the minimum size of a vertex cover of F_2 is more than i is in co-NP, from Lemma 4, there is a PG G, a set Q of 3 pairs of opposite literals in G and an instance of ENTAILMENT$_3$ $g(F_2, i) = (g_G(F_2, i), g_H(F_2, i))$ such that $i \leq min\text{-}vc(F_2)$ iff $g_G(F_2, i)$ is entailed by $g_H(F_2, i)$, $g_G(F_2, i) = G$, each exchangeable pair w.r.t. $(G, g_H(F_2, i))$ is in Q, the set of relation node labels in $g_H(F_2, i)$ is the same as in G, and G and $g_H(F_2, i)$ do not contain any constant node, with G and Q being independent from i. Let $G_i = f_G(F_1, i)$, $H_i = f_H(F_1, i)$ and $H'_i = g_H(F_2, i)$. Comparing the sizes of the minimum vertex covers for F_1 and F_2 can be done by asking $q + 1$ questions, where $q = |V_2|$: is there some i, $0 \leq i \leq q$, such that $min\text{-}vc(F_1) \leq i$ and $i \leq min\text{-}vc(F_2)$, i.e., such that G_i is entailed by H_i and G is entailed by H'_i? Thus we have to build G' and H' from the PGs G_i, H_i, G and H'_i such that (1) G' has at most 3 exchangeable pairs w.r.t. (G', H') and (2) G' is entailed by H' if and

[10] A vertex cover of F is a set S of vertices such that each edge is adjacent to at least a vertex of S.

G_0

G_q

G_1

G_j

G_i

.

.

.

.

G_0

G_q

G_1

G_j

H_i

.

.

.

.

.

.

H'_i

H'^{c2}, π_1 maps G to $H_i'^{c1}$ and π_2 maps G to $H_i'^{c2}$. Hence $\{+p(u), -p(v)\}$ is an exchangeable pair of G' w.r.t. (G, H_i'), and therefore is in Q by hypothesis on (G, H_i'), which completes the proof that G' has at most 3 exchangeable pairs w.r.t. (G', H').

It remains to prove that G' is entailed by H' if and only if there is some i, $0 \leq i \leq q$, such that G_i is entailed by H_i and G is entailed by H_i'.

$\Rightarrow$: By contradiction: we assume that G' is entailed by H' but that there is no i such that G_i is entailed by H_i and G is entailed by H_i'. Then for each i in $[0, q]$ there is a completion H_i^c of H_i such that G_i cannot be mapped to H_i^c or there is a completion $H_i'^c$ of H_i' such that G cannot be mapped to $H_i'^c$. Let H'^c be a completion of H' such that for each i in $[0, q]$, if H_i^c exists then H_i^c is a subgraph of the part H_i of H'^c, otherwise $H_i'^c$ is a subgraph of the part H_i' of H'^c. As G' is entailed by H', there is a homomorphism π from G' to H'^c. By Lemma 5, there are an integer i in $[0, q]$, a completion H_i^d of H_i and a completion $H_i'^d$ of H_i' such that H_i^d is a subgraph of the part H_i of H'^c, $H_i'^d$ is a subgraph of the part H_i' of H'^c and π maps G_i to H_i^d and G to $H_i'^d$. If H_i^c exists then H_i^c and H_i^d are completions of H_i that are both subsets of the part H_i of H'^c, hence ${H_i}^c = {H_i}^d$ and π maps G_i to H_i^c, a contradiction. Otherwise $H_i'^c$ exists, similarly $H_i'^c = H_i'^d$ and π maps G to $H_i'^c$, a contradiction.

$\Leftarrow$: We assume that there is some i such that G_i is entailed by H_i and G is entailed by H_i'. Let us show that G' is entailed by H'. Let H'^c be a completion of H'. Let us show that G' can be mapped to H'^c. Let H_i^c be the completion of H_i such that H_i^c is a subgraph of the part H_i of H'^c, and let π_1 be a homomorphism from G_i to H_i^c. Let $H_i'^c$ be the completion of H_i' such that $H_i'^c$ is a subgraph of the part H_i' of H'^c, and let π_2 be a homomorphism from G to $H_i'^c$. Then there is a homomorphism π from G' to H' extending π_1 and π_2: π maps each v_j to v_j^i and each G_j, with $j \neq i$, to the part G_j inside A_i of H'^c.

We have thus built a polynomial reduction from Min-card-vertex cover compare to ENTAILMENT$_3$, which proves the theorem. □

From Th. 5 and Th. 7, we conclude that ENTAILMENT$_3$ is $P_{||}^{NP}$-complete.

4.5 When Homomorphism Checking is Polynomial

Checking the existence of a homomorphism becomes polynomial when G has a tree-like structure. More precisely, if G is seen as a graph, it is said to have a tree-like structure if it a 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, it has a tree-like structure if it has 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 another query or a knowledge base composed of a set of facts. Indeed, a query generally has a simple structure.

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

Theorem 8 *When G has bounded treewidth or hypertreewidth, the following complexity results hold:*

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

Proof: ENTAILMENT 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-ENTAILMENT (the size of H^c is polynomial in the size of H and checking that there is no homomorphism from G to H^c can be done in polynomial time since G has bounded treewidth or hypertreewidth). Its completeness for this complexity class follows from the proof of Th. 6, which shows that ENTAILMENT$_3$ remains co-NP-hard when G has bounded treewidth (in the reduction, the graph G built is a tree). Hence, ENTAILMENT$_k$ is also co-NP-complete for any $k \geq 3$. That ENTAILMENT$_0$ is in P follows immediately from Prop. 8. To show that ENTAILMENT$_1$ is also in P, let us consider Algorithm 1. Checking if there is an extensible homomorphism from $G \setminus \{\sim p(u)\}$ to H can be done in polynomial time as follows.

We recall that there is an extensible homomorphism from $G \setminus \{\sim p(u)\}$ to H if and only if there is a homomorphism π from $G \setminus \{\sim p(u)\}$ to H such that $\overline{\sim p}(\pi(u))$ is not in H. Let s be the arity of p, let r be a new relation name, i.e., that does not occur in G and H, with arity s, let $G' = (G \setminus \{\sim p(u)\}) + \{+r(u)\}$, and let H' be the PG obtained from H by adding the literal $+r(v)$ for each tuple v of s term nodes of H such that $\overline{\sim p}(v)$ is not in H (these tuples are in polynomial number since the arity of relation names is bounded by a constant). There is a homomorphism π from $G \setminus \{\sim p(u)\}$ to H such that $\overline{\sim p}(\pi(u))$ is not in H if and only if there is a homomorphism from G' to H'. As G' is obtained from G by replacing relation name p by r, G' has also bounded treewidth or hypertreewidth, hence the existence of a homomorphism from G' to H' can be checked in polynomial time. It follows that ENTAILMENT$_1$ is in P. □

The previous theorem can be generalized to all cases where the existence of a homomorphism from G to H can be checked in polynomial time, provided, in the

case of ENTAILMENT$_1$, that this property is preserved on the PGs G' and H' built from G and H in the previous proof.

4.6 Pieces

We will now take advantage of some simple graph properties to extend the previous results. First note that G is entailed by H if and only if each connected component of G is entailed by H. Second, by splitting 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 of a homomorphism from G to any normal graph. 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.

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 two relation nodes r and s 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 $x_0(= r) \ldots x_n(= s)$ such that, for $0 < i < n$, 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 forms its own piece. See Figure 9, 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.

Proposition 13 *Let G and H be two PGs, with H being consistent. G is entailed by H if and only if each piece of G is entailed by H.*

Thus, 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.

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 can thus be broken, which may produce a graph decomposable into a tree (cf. Section 4.5).

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

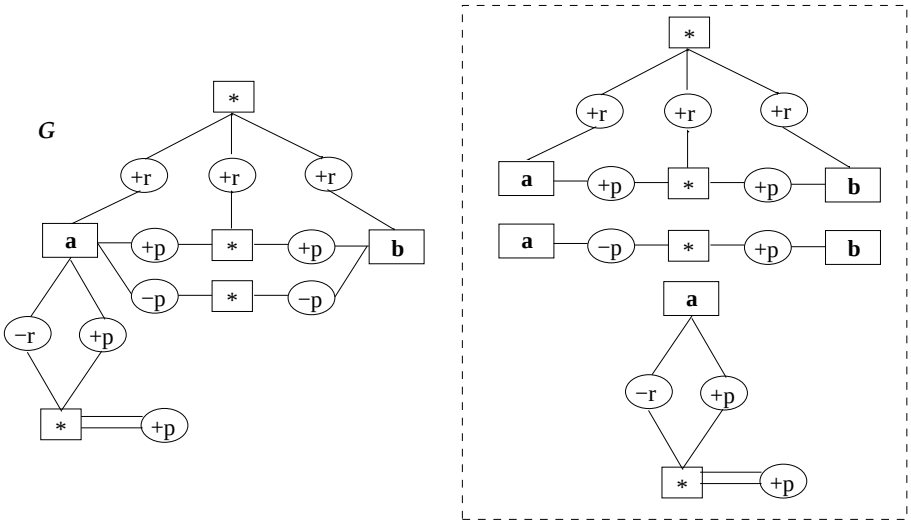


Figure 9: Pieces

5 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 entailed by H if and only if G can be mapped to each completion of H. By Th. 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.

5.1 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. We will give a simple process leading to an inclusion-smaller completion vocabulary (and therefore an inclusion-smaller set of exchangeable pairs). 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 obtained by this process is denoted by $\mathcal{R}(G, H)$. We give a declarative definition of $\mathcal{R}(G, H)$ and prove that it can be computed by the process explained above, which is formalized in Algorithm 3.

Definition 10 (Refined completion vocabulary $\mathcal{R}(G, H)$) *Let G and H be two PGs, with H being consistent, and let $\mathcal{R}_0$ be the completion vocabulary w.r.t. (G, H). The* refined completion vocabulary w.r.t. (G, H)*, denoted by $\mathcal{R}(G, H)$, is the inclusion-maximum subset $\mathcal{R}$ of $\mathcal{R}_0$ such that each relation name in $\mathcal{R}$ appears in some exchangeable literal w.r.t. $(G, H, \mathcal{R})$.*

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

Proposition 14 *Algorithm 3 is correct.*

Proof: Let $\mathcal{R}^*$ be the set computed by Algorithm 3. Let us show that $\mathcal{R}^* = \mathcal{R}(G, H)$, i.e.,

a) each relation name in $\mathcal{R}^*$ appears in some exchangeable literal w.r.t. $(G, H, \mathcal{R}^*)$, b) each subset $\mathcal{R}'$ of $\mathcal{R}_0$ such that each relation name in $\mathcal{R}'$ appears in some exchangeable literal w.r.t. $(G, H, \mathcal{R}')$ is a subset of $\mathcal{R}^*$.
Item a) follows from the exit condition of the repeat loop. Let us prove item b). Let $\mathcal{R}'$ be a subset of $\mathcal{R}_0$ such that each relation name in $\mathcal{R}'$ appears in some exchangeable literal w.r.t. $(G, H, \mathcal{R}')$. Let us show that $\mathcal{R}' \subseteq \mathcal{R}^*$. It is sufficient to show that the inclusion $\mathcal{R}' \subseteq \mathcal{R}$ is an invariant of the repeat loop. It holds at the initialization step since the initial value of $\mathcal{R}$ is $\mathcal{R}_0$ and $\mathcal{R}'$ is a subset of $\mathcal{R}_0$.

We suppose that $\mathcal{R}' \subseteq \mathcal{R}_i$. Let us show that $\mathcal{R}' \subseteq \mathcal{R}_{i+1}$, where $\mathcal{R}_{i+1}$ is the set obtained from $\mathcal{R}_i$ after one iteration of the repeat loop. Let $r \in \mathcal{R}'$. Let us show that $r \in \mathcal{R}_{i+1}$. By hypothesis on $\mathcal{R}'$, as $r \in \mathcal{R}'$, r appears in some exchangeable literal w.r.t. $(G, H, \mathcal{R}')$, which is also an exchangeable literal w.r.t. $(G, H, \mathcal{R}_i)$ since $\mathcal{R}' \subseteq \mathcal{R}_i$, and therefore $r \in \mathcal{R}_{i+1}$. □

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 5.2). 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 for the last one where $\mathcal{R}$ is unchanged.

Note that the number of iterations of the repeat loop is unbounded. Indeed, given any positive integer n, we can build two PGs G' and H' such that the execution of Algorithm 3 on G' and H' needs $n + 2$ iterations. We define G' and H' from the PGs G and H shown in Figure 4 as follows. For each i in $1, \ldots, 2n - 1$, let G_i be the PG obtained from G by adding the literals $+r_i(x, y)$ and $-r_{i+1}(y, z)$.

For instance, if $n = 2$, we need 4 relation names r_1, r_2, r_3 and r_4, and G_1 (resp. G_2, G_3) is obtained from G by adding literals $+r_1(x, y)$ and $-r_2(y, z)$ (resp. $+r_2(x, y)$ and $-r_3(y, z)$, $+r_3(x, y)$ and $-r_4(y, z)$). Note that none of these PGs G_i contains a relation node labeled with $-r_1$ or with $+r_{2n}$. Let G' be the PG obtained from the disjoint union of copies of the PGs G_i for all i in $1, \ldots, 2n - 1$ by adding the literals $-r_1(e, e)$ and $+r_{2n}(e, e)$. Let H_* be the PG obtained from $G \setminus \{+p(x), -p(y), -p(z)\}$ by adding the literals $+r_i(x, y)$ and $-r_i(y, z)$ for all i in $1, \ldots, 2n$. For instance, if $n = 4$, the literals of H_* are $+r(x, y)$, $+r(y, z)$, $+r_1(x, y)$, $+r_2(x, y)$, $+r_3(x, y)$, $+r_4(x, y)$, $-r_1(y, z)$, $-r_2(y, z)$, $-r_3(y, z)$ and $-r_4(y, z)$. Let H' be the PG obtained from the disjoint union of H and H_* by adding the literals $-r_1(e, e)$ and $+r_{2n}(e, e)$. The set $\mathcal{R}$ is initialized with $\{p, r_1, \ldots, r_{2n}\}$. Relation names r_1 and r_{2n} are eliminated from $\mathcal{R}$ at the first iteration of the repeat loop. As r_1 and r_{2n} are no longer in $\mathcal{R}$, r_2 and r_{2n-1} are eliminated from $\mathcal{R}$ at the second iteration, and so on. The set $\mathcal{R}$ is reduced to $\{p\}$ after iteration n, and becomes empty at iteration $n + 1$. As there is

no homomorphism from G' to H', we conclude that G' is not entailed by H'.

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 Th. 2 and Th. 3 and Prop. 12 still hold. For this, it is sufficient to show that completions w.r.t. $\mathcal{R}(G, H)$ satisfy the Completion Property.

Definition 11 (Completion Property for $\mathcal{R}$) *Let G and H be two PGs, with H being consistent, and let $\mathcal{R}$ be a set of relation names.* $\mathcal{R}$ satisfies the Completion Property w.r.t. (G, H) *if the following equivalence holds: G is entailed by H if and only if G can be mapped to each completion of H w.r.t. $\mathcal{R}$.*

Proposition 15 *Let G and H be two PGs, with H being consistent. $\mathcal{R}(G, H)$ satisfies the Completion Property w.r.t. (G, H).*

Proof: Let $P(\mathcal{R})$ be the property defined for any set $\mathcal{R}$ of relation names by:
$P(\mathcal{R})$: $\mathcal{R}$ satisfies the Completion Property w.r.t. (G, H).
Let us show that $P(\mathcal{R})$ is an invariant of the repeat loop in Algorithm 3. By Property 3, $P(\mathcal{R})$ 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 and any completion of H w.r.t. $\mathcal{R}'$ is a subgraph of a completion of H w.r.t. $\mathcal{R}$, it is sufficient to show that if G can be mapped to each completion of H w.r.t. $\mathcal{R}$ then it can be mapped to each completion of H w.r.t. $\mathcal{R}'$. We suppose that G can be mapped to each completion of H w.r.t. $\mathcal{R}$, and let H^c be a completion of H w.r.t. $\mathcal{R}'$. Let us show that G can be mapped to H^c. Let H' be a completion of H w.r.t. $\mathcal{R}$ containing H^c. As $P(\mathcal{R})$ holds, Th. 2 (with completions and G_s being defined w.r.t. $\mathcal{R}$) holds too. Let π be a homomorphism from G to H' mapping G_s to H. Each literal of G that is not mapped to a literal in H is exchangeable w.r.t. $(G, H, \mathcal{R})$, and therefore is mapped to a literal in H^c (since its relation name is in $\mathcal{R}'$). Hence π maps G to H^c. □

It follows that all results of this paper still hold with $\mathcal{R}(G, H)$ as completion vocabulary.

Note that any superset of $\mathcal{R}(G, H)$ also satisfies the Completion Property. 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 of 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.

5.2 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 12 (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 in* H *such that neither* $+p(w)$ *nor* $-p(w)$ *is a literal in* H.

Definition 13 (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 14 (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* $\{+p(u), -p(v)\}$ *such that for some* w, $\{+p(u), -p(v), w\}$ *is an exchangeable triple w.r.t.* $(G, H, \mathcal{T})$.

The set $\mathcal{T}(G, H)$ is defined similarly to $\mathcal{R}(G, H)$ and computed by Algorithm 4.

Definition 15 ($\mathcal{T}(G, H)$) *Let* G *and* H *be two PGs, with* H *being consistent, and let* $\mathcal{T}_0$ *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.* $(G, H, \mathcal{R}(G, H))$. $\mathcal{T}(G, H)$ *is the inclusion-maximum subset* $\mathcal{T}$ *of* $\mathcal{T}_0$ *such that each triple in* $\mathcal{T}$ *is an exchangeable triple w.r.t.* $(G, H, \mathcal{T})$.

Proposition 16 *Algorithm 4 is correct.*

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

Proof: It is similar to that of Prop. 14. □

Let us illustrate Algorithm 4 on the PGs G and H pictured in Figure 4. $\mathcal{T}$ is initialized with $\{\{+p(x), -p(y), b\}, \{+p(x), -p(y), d\}\}$. It becomes $\{\{+p(x), -p(y), b\}\}$ after the first iteration of the repeat loop, which reduces the set of completions of H w.r.t. $\mathcal{T}$ to $\{H + \{+p(b)\}, H + \{-p(b)\}\}$. It becomes empty after the second iteration, since $+p(x)$ can no longer be mapped to $+p(b)$ by a homomorphism from G to a completion of H w.r.t. $\mathcal{T}$, as 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 entailed by H (provided that Prop. 8 still holds, which is checked below).

We prove that all results of this paper still hold, similarly to the proofs for $\mathcal{R}(G, H)$ by replacing $\mathcal{R}(G, H)$ with $\mathcal{T}(G, H)$.

Definition 16 (Completion Property for $\mathcal{T}$) *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).* $\mathcal{T}$ satisfies the Completion Property w.r.t. (G, H) *if the following equivalence holds: G is entailed by H if and only if G can be mapped to each completion of H w.r.t. $\mathcal{T}$.*

Proposition 17 *Let G and H be two PGs, with H being consistent. $\mathcal{T}(G, H)$ satisfies the Completion Property w.r.t. (G, H).*

Proof: Let $P(\mathcal{T})$ be the property defined for any set $\mathcal{T}$ of triples w.r.t. (G, H) by:
$P(\mathcal{T})$: $\mathcal{T}$ satisfies the Completion Property w.r.t. (G, H).
Let us show that $P(\mathcal{T})$ is an invariant of the repeat loop in Algorithm 4. $P(\mathcal{T})$ holds at the initialization of the loop since the completions of H w.r.t. $\mathcal{T}$ are the completions of H w.r.t. $\mathcal{R}(G, H)$. We suppose that $P(\mathcal{T})$ holds. Let $\mathcal{T}'$ be the set of exchangeable triples w.r.t. $(G, H, \mathcal{T})$. Let us show that $P(\mathcal{T}')$ holds. It is

sufficient to show that if G can be mapped to each completion of H w.r.t. $\mathcal{T}$ then it can be mapped to each completion of H w.r.t. $\mathcal{T}'$. We suppose that G can be mapped to each completion of H w.r.t. $\mathcal{T}$, and let H^c be a completion of H w.r.t. $\mathcal{T}'$. Let us show that G can be mapped to H^c. Let H' be a completion of H w.r.t. $\mathcal{T}$ containing H^c. It is no longer sufficient to apply Th. 2 on H', as we did for $\mathcal{R}(G, H)$, but we can use an argument similar to that used in the proof of Th. 2. Let R be the set of literals l in $H' \setminus H^c$ such that there is a homomorphism from G to H' mapping some literal of G to l. R is consistent since it is a set of literals in H'. Let H'' be the completion of H w.r.t. $\mathcal{T}$ obtained from H' by replacing every literal of R by its complementary literal, and let π be a homomorphism from G to H'' (such a homomorphism exists by hypothesis on $\mathcal{T}$). Let us show that π maps G to H^c. No literal of G can be mapped by π to the complementary literal of a literal l of R (otherwise this literal of G would be in an exchangeable triple w.r.t. $(G, H, \mathcal{T})$, so l would be a literal in H^c). Thus π is a homomorphism from G to H'. Therefore, by definition of R, every literal of G is mapped by π to either H^c or R. However, as π is a homomorphism from G to H'', which contains no literal of R, no literal of G can be mapped to R, thus π maps G to H^c. □

Note that any superset of $\mathcal{T}(G, H)$ also satisfies the Completion Property. In practice, we obtain a partially refined set of exchangeable triples by initializing $\mathcal{T}$ with the set of triples $\{+p(u), -p(v), w\}$ w.r.t. (G, H) such that p belongs to a partially refined completion vocabulary previously computed, and successively removing triples that can be recognized as non exchangeable. For instance, in the example of Figure 4 with (partially refined) completion vocabulary $\{p\}$, $\mathcal{T}$ initially contains the triples $\{+p(x), -p(y), b\}$, $\{+p(x), -p(y), d\}$, $\{+p(x), -p(z), b\}$ and $\{+p(x), -p(z), d\}$. The three last triples are clearly non exchangeable, and removing them makes $\{+p(x), -p(y), b\}$ clearly non exchangeable.

6 Related Work and Conclusion

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

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 ENTAILMENT, 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 is entailed by 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 ENTAILMENT$_0$. 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 of 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 a homomorphism check 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 ENTAILMENT$_0$. 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 ENTAILMENT$_0$.

Conclusion. In this paper, we have solved the main issues concerning the role of exchangeable literals in the complexity of FOL$(\exists, \wedge, \neg_a)$-ENTAILMENT. We have shown that, as soon as the number k of exchangeable pairs is bounded, the complexity falls into $P_{||}^{NP}$, and becomes even NP-complete if $k \leq 1$. We have also shown that the problem is $P_{||}^{NP}$-complete for any k greater of equal to 3. To

[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.

complete the picture, it would be interesting to determine its complexity for $k = 2$.

Let us mention that exchangeable literals can be exploited in algorithms solving ENTAILMENT 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 entailment 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, i.e., a subset of incomparable graphs of this space $\{H_1, \ldots, H_k\}$ such that (1) there is a homomorphism from G to each H_i ; (2) for each H^c there is a H_i with $H^c \leq H_i$". This algorithm is then refined and experimentally evaluated on random instances in [BLM10]. Some special subgraphs of G, that are necessarily mapped to H if G is entailed by 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 entailed by H) and to guide the space exploration. These subgraphs are without opposite literals. They can be replaced by subgraphs without exchangeable pairs (see Th. 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 Prop. 3): this set can be further restricted to relation names occurring in exchangeable literals of G (Prop. 15), and the notion of completion can be further refined, using exchangeable triples (Prop. 17).

This paper is devoted to theoretical issues. As for further work, it would be interesting to study experimentally the practical interest of the obtained results. An issue is to study to what extent they can be used to improve the above mentioned algorithm, either on difficult problem instances (as in [BLM10]) or on real data. On real conjunctive queries with negation namely, the number of exchangeable literal pairs is expected to be null in many cases. A question is whether this number is upper bounded by a fixed value in practical query sets.

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.

[BH91] S. R. Buss and L. Hay. On truth-table reducibility to sat. *Inf. Comput.*, 91(1):86–102, 1991.

[BLM10] K. Ben Mohamed, M. Leclère, and M.-L. Mugnier. Containment of conjunctive queries with negation: Algorithms and experiments. In *DEXA (2)*, volume 6262 of *Lecture Notes in Computer Science*, pages 330–345. Springer, 2010.

[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.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[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*, volume 4068 of *Lecture Notes in Artificial Intelligence*, pages 331–345. Springer, 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.

[SV00] H. Spakowski and J. Vogel. Theta$_2^{\mathrm{p}}$-completeness: A classical approach for new results. In Sanjiv Kapoor and Sanjiva Prasad, editors, *FSTTCS*, volume 1974 of *Lecture Notes in Computer Science*, pages 348–360. Springer, 2000.

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

[W3C04] W3C. Resource Description Framework (RDF). Technical report, www.w3.org, 2004.

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