An Algorithmic Study of Deduction in Simple Conceptual Graphs with Classical Negation

Michel Leclère and Marie-Laure Mugnier

LIRMM, CNRS - Université Montpellier 2,
161, rue Ada, F-34392 Montpellier cedex, France,
{leclere,mugnier}@lirmm.fr

Abstract. Polarized conceptual graphs (PGs) are simple conceptual graphs added with a restricted form of negation, namely negation on relations. Classical deduction with PGs (in short PG-Deduction) is highly intractable; it is indeed Π_P^2 complete. In [LM06] a brute-force algorithm for solving PG-Deduction was outlined. In the present paper, we extend previous work with two kinds of results. First, we exhibit particular cases of PGs for which the complexity of PG-Deduction decreases and becomes not more difficult than in simple conceptual graphs. Secondly, we improve the brute-force algorithm with several kinds of techniques based on properties concerning the graph structure and the labels.

1 Introduction

Simple conceptual graphs (SGs) [CM92] constitute the kernel of conceptual graphs (CGs) [Sow84]. They can be used as such, to represent facts or queries. They are also basic bricks for more complex constructs, corresponding to more expressive conceptual graphs, for instance rules or constraints [BM02]. Full conceptual graphs are obtained when negation is added to SGs without restriction. Several works inspired from Peirce's existential graphs, a diagrammatical system for logics, have studied full conceptual graphs, in particular [Sow84,WL94,Dau03]. Full conceptual graphs have the expressive power of FOL. We think that they are too complicated at the end-user level, for building knowledge-based systems and understanding how they work. They are also too complex from a computational viewpoint since deduction becomes non decidable. We thus prefer to add a limited form of negation to SGs, namely atomic negation. Atomic negation allows us to express knowledge of form "this kind of relation does not hold between these entities", "this entity does not have that property" or "this entity is not of that type".

Polarized Graphs. SGs plus atomic negation yield polarized graphs (PGs), which are equivalent to the FOL fragment of existentially closed conjunctions of positive and negative literals. Several works have pointed out difficulties introduced by atomic negation (and, similarly, inequality) in classical FOL [Mug00] [Ker01] [Kli05]. In [LM06] [ML07], we discussed several semantics of negation in relation with deduction checking but also with querying PGs: negation with closed-world assumption, negation in classical FOL, and negation in intuitionistic logic. In the first and in the third case, negation can be processed without complexity overhead. In the classical case, deduction checking in PGs (PG-Deduction) becomes highly intractable: indeed, it can be shown that it is Π_P^2-complete (Π_p^2 is co-NP^{NP}), whereas deduction in SGs is NP-complete (see f.i. [Mug07] for a proof of Π_P^2-completeness).

Contribution. In [LM06] [ML07], we proposed a brute-force algorithm for (classical) PG-Deduction. In the present paper, we extend this previous work with two kinds of results. First, we exhibit particular cases of PGs for which the complexity of PG-Deduction decreases and becomes not more difficult than in SGs. These particular cases rely on the notion of pair of *exchangeable* relation nodes (that appear in the graph to be deduced). Secondly, we improve the brute-force algorithm with several kinds of techniques based on properties concerning the graph structure and the labels. Finally, let us mention that this paper extends another work of ours on the containment problem of conjunctive queries with negation, in the context of databases [LM07]. Indeed, this problem can be seen as a particular case of PG-Deduction, where relation types are not partially ordered. Furthermore, the notion of exchangeable relations defined here generalizes that of opposite literals in [LM07].

The sequel of this paper is organized as follows. Section 2 is devoted to preliminary definitions and results. Exchangeable pairs of relations and related special cases are studied in section 3 and algorithmic improvments in section 4. These improvements are based first on a limitation of the "completion vocabulary", and secondly on a specific exploration of the search space.

2 Polarized Graphs

In this section, we define notations and recall some definitions and results of [LM06] about polarized conceptual graphs. We assume that the reader is familiar with the basics of conceptual graphs (cf. for instance [ML07] for definitions consistent with the present paper).

Basic notations and results. A conceptual graph *vocabulary* contains at least a poset (partially ordered set) of concept types, a poset of relation types and a set of individual markers. We denote by $\mathcal{V}$ a vocabulary, and by T_R its set of relation types. Φ is the classical translation from conceptual graphs (and vocabularies) to first-order logic (FOL). $\Phi(G)$ denotes the logical formula assigned by Φ to a graph G, and $\Phi(\mathcal{V})$ denotes the set of formulas translating the concept and relation type posets. We use the symbol $\vDash$ to denote both the logical entailment and the deduction (as both notions are equivalent in FOL). *Projection* is the fundamental mechanism to reason with simple conceptual graphs (SGs). Since it is indeed a graph homomorphism, and the term "projection" can be misleading because of the operation with the same name in databases, we prefer to call it *SG* *homomorphism*, or simply *homomorphism* if there is no ambiguity. If there is a homomorphism from G to H, we say that G can be *mapped* to H. A SG is *normal* if it does not possess two concept nodes with the same individual marker. Under natural assumptions every SG has a unique normal form. SG homomorphism is *sound* and *complete* with respect to Φ, i.e.: for all SGs G and H on a vocabulary $\mathcal{V}$, if there is a homomorphism from G to H then $\Phi(\mathcal{V}), \Phi(H) \models \Phi(G)$ (soundness, [Sow84]) and if $\Phi(\mathcal{V}), \Phi(H) \models \Phi(G)$ then there is a homomorphism from G to the normal form of H (completeness, [CM92]).

Polarized conceptual Graphs (PGs). PGs are built from SGs by "polarizing" their relation nodes. Beside positive relation nodes, there are now negative relation nodes.

Definition 1 (Polarized Graph (PG)). *A* polarized graph *(PG) is defined similarly to a SG except that relation nodes are labeled not only by a type but also by a polarity (denoted + or −). A* positive *(resp.* negative*) relation node is labeled by $(+r)$ (resp. $(-r)$), where r is a relation type. $(+r)$ can also be noted (r).*

A negative relation node with label $(-r)$ and arguments $(c_1 ... c_k)$ expresses that "there is no relation of type r between $c_1 ... c_k$" (or if $k = 1$, "c_1 does not possess the property r"); it is logically translated by Φ into the literal $\neg r(e_1 ... e_k)$, where e_i is the term assigned to c_i. PGs are equivalent to the FOL fragment composed of existentially closed conjunctions of (positive and negative) literals (without functions). In the following, we note $+r(c_1 ... c_k)$ (resp. $-r(c_1 ... c_k)$), a relation node with label $+r$ (resp. $-r$) and neighbor list $c_1 ... c_k$, where the $c_1 ... c_k$ are not necessarily distinct nodes.

Definition 2 (inconsistent PG). *A PG is said to be* inconsistent *if its normal form contains two relation nodes $+r(c_1 ... c_k)$ and $-s(c_1 ... c_k)$ with contradictory labels, i.e. with $r \leq s$. Otherwise it is said to be* consistent.

The following property is immediate:

Property 1. For any PG G on a vocabulary $\mathcal{V}$, G is inconsistent iff $\Phi(\mathcal{V}) \cup \{\Phi(G)\}$ is inconsistent.

The order on relation labels is extended as follows: we set $-r_1 \leq -r_2$ if $r_2 \leq r_1$.

Definition 3 (Extended order on relation labels). *Given two relation labels l_1 and l_2, $l_1 \leq l_2$ if, either l_1 and l_2 are both positive labels, say $l_1 = (r_1)$ and $l_2 = (r_2)$, and $r_1 \leq r_2$, or l_1 and l_2 are both negative labels, say $l_1 = (-r_1)$ and $l_2 = (-r_2)$, and $r_2 \leq r_1$.*

Given this extended order on relation labels, homomorphism can be used without changing its definition. Recall that homomorphism is sound and complete w.r.t. logical deduction for SGs. For PGs, one part of the property still holds:

Property 2. Given two PGs G and H on a vocabulary $\mathcal{V}$, if there is a homomorphism from G to H then $\Phi(\mathcal{V}), \Phi(H) \models \Phi(G)$.

Thus, homomorphism remains sound. But it is no longer complete. Indeed, we might have $\Phi(\mathcal{V}), \Phi(H) \models \Phi(G)$ and no homomorphism from G to H, as illustrated by Figure 1. The formulas assigned to G and H by Φ (here we ignore the atoms associated with concept nodes) are respectively $\Phi(G) = \exists x \exists y p(x) \wedge \neg p(y) \wedge r(x,\ y)$ and $\Phi(H) = p(a) \wedge r(a,b) \wedge r(b,c) \wedge \neg p(c)$. $\Phi(G)$ can be deduced from $\Phi(H)$ using the tautology $p(b) \vee \neg p(b)$ (indeed, every model of $\Phi(H)$ satisfies either $p(b)$ or $\neg p(b)$; if it satisfies $p(b)$, then x and y are interpreted as b and c; in the opposite case, x and y are interpreted as a and b; thus every model of $\Phi(H)$ is a model of $\Phi(G)$).

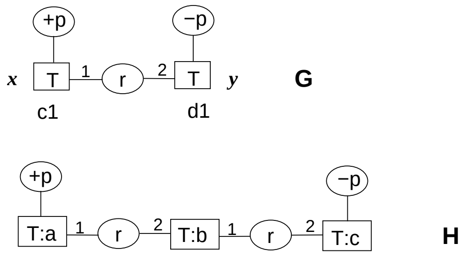


Fig. 1. Atomic negation and homomorphism

More generally, negation introduces disguised disjunctive information that cannot be taken into account by homomorphism. This disjunctive information is related to the law of the excluded-middle which holds in classical logic: given a proposition P, either P is true, or $\neg P$ is true. This leads to reasoning by cases: if a relation is not asserted in a fact, either it is true or its negation is true. We thus have to consider all ways of *completing* the knowledge asserted by a PG. Let us look again at the example in Fig. 1. H does not say whether the unary relation 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 Fig. 2). There is a homomorphism from G to H_1 and there is a homomorphism from G to H_2. From the soundness of homomorphism, we conclude that G can be logically deduced from H.

The next definition specifies the notion of completion of a PG relative to a vocabulary $\mathcal{V}$.

Definition 4 (Complete PG). *A* complete PG *on a vocabulary $\mathcal{V}$ with relation type set T_R is a consistent (normal) PG satisfying the following completion condition: for each relation type r of arity k in T_R, for each k-tuple of concept nodes $(c_1, \ldots, c_k)$, where $c_1, \ldots, c_k$ are not necessarily distinct nodes, there is a relation $+s(c_1, \ldots, c_k)$ with $s \leq r$ or (exclusive) there is a relation $-s(c_1, \ldots, c_k)$*

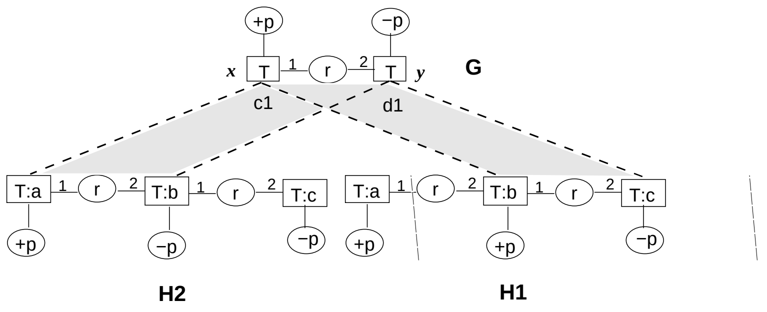


Fig. 2. When the law of the excluded-middle intervenes

with $r \leq s$. A PG is complete w.r.t. a subset of relation types $T \subseteq T_R$ *if the completion condition considers only elements of T. If a PG G^c that is complete w.r.t. T is obtained by adding relations to a graph G, it is called a T-*completion *of G (or simply a* completion *of G if T is implicit).*

Property 3. If a relation node is added to a complete PG, either this relation node is redundant (there is already a relation node with the same argument list and a label less or equal to it) or it makes the PG inconsistent.

A complete PG is obtained from a consistent PG G by repeatedly adding positive and negative relations as long as adding a relation brings new information and does not yield an inconsistency. Since a PG is a finite graph defined over a finite vocabulary, the number of different complete PGs that can be obtained from it is finite.

We can now define deduction on PGs.

Definition 5 (PG-Deduction problem). *The PG-Deduction problem takes two PGs G and H as input, with H being consistent, and asks whether G can be PG-deduced from H, i.e. whether for each complete PG H^c obtained from H, there is a homomorphism from G to H^c.*

Theorem 1. *Let G and H be two PGs defined on a vocabulary $\mathcal{V}$. H is a consistent PG. Then G can be PG-deduced from normal(H) if and only if $\Phi(\mathcal{V}), \Phi(H) \models \Phi(G)$.*

From now on, we will confuse PG-deduction and logical deduction on the associated formulas, and simply say "deduction". Algorithm 1 is a brute-force algorithmic schema for checking deduction (algorithmic improvements are studied in Section 4). An immediate observation for generating the completions H^c is that we do not need to consider all relation types but only those appearing in G. The algorithm generates all complete PGs relative to this set of types and for each of them checks whether G can be mapped to it. A complete graph to which G cannot be mapped can be seen as a counter-example to the assertion that G is deducible from H.

Algorithm 1: PG-eduction

```
Data: PGs $G$ and $H$, s.t. $H$ is consistent
Result: true if $G$ can be deduced from $H$, false otherwise
begin
    Compute $\mathcal{H}$ the set of complete PG obtained from $H$ w.r.t. relation types in $G$;
    forall $H^c \in \mathcal{H}$ do
        if there is no homomorphism from $G$ to $H^c$ then
            return false ;    // $H^c$ is a counter-example
    return true;
end
```

3 Special Cases with Lower Complexity for PG-Deduction

The existence of a PG homomorphism from G to H is a sufficient condition for G to be deducible from H. However it is not a necessary condition, as we have seen before. In this section we study the question "when is a homomorphism from G to H a necessary condition for G to be deducible from H?". Answers to this question yield particular cases where the theoretical complexity of PG-Deduction decreases. We shall also identify special subgraphs of G for which there must be a homomorphism to H when G is deducible from H. These subgraphs can be used as filters or guides during the completion algorithm.

Let us first identify relation nodes in G which might play a role in the problem complexity, in the sense that they may lead to use the law of the excluded-middle.

Definition 6 (Opposite relation labels and nodes). *Two relation labels are said to be* opposite *if they have opposite polarities and if the type r of the positive label is more general than the type s of the negative label (i.e. $r \geq s$). By extension, two relations nodes are said to be opposite if they have opposite labels $(+r)$ and $(-s)$.*

Let us say that two opposite relation nodes of G are "exchangeable" if they can have the same list of images for their arguments by homomorphisms to (necessarily distinct) completions of H.

Definition 7 (Exchangeable relations). *Two relation nodes $+r(c_1...c_k)$ and $-s(d_1...d_k)$ in G are* exchangeable *with respect to H if (1) they are opposite, (2) there are two completions of H, say H_1 and H_2, and two homomorphisms h_1 and h_2, respectively from G to H_1 and from G to H_2, such that for all $i : 1...k$, $h_1(c_i) = h_2(d_i)$.*

See, for instance, the PG G in Fig. 1. Let us consider the opposite relation nodes $r_1 = p(c_1)$ and $r_2 = -p(d_1)$. These nodes are exchangeable, as can be seen in Fig. 2: there is a homomorphism h_1 from G to a completion H_1 of H and there is a homomorphism h_2 from G to another completion H_2 of H, such that $h_1(c_1) = h_2(d_1)$ (and is the concept node in H with marker b).

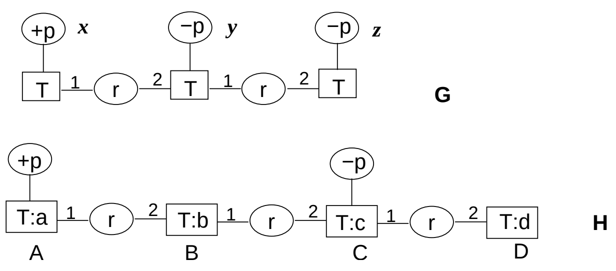


Fig. 3. Exchangeable and opposite relation nodes

The definition of exchangeable relations is strictly more restrictive than the definition of opposite relations. See Figure 3 for instance (from [Tho07], where x and y are opposite nodes, as well as x and z. x and y are exchangeable, which can be seen with the following two completions of H: in one completion, say H_1 , $-p(B)$ is added (and a homomorphism from G to H_1 maps the neighbor of y to B); in another completion, say H_2, $p(B)$ and $-p(D)$ are added (and a homomorphism from G to H_2 maps the neighbor of x to B). It can be checked that x and z are not exchangeable: there are no two completions such that their argument can be mapped to the same node.

Property 4. Let G and H be two PGs, with G having no pair of exchangeable relations w.r.t. H. If G is deducible from H, then there is a homomorphism from G to H.

Proof. Let H^{c+} be a completion of H with solely positive relations, and furthermore with all possible positive relations: for all n-ary type p and for all n-tuple u of concept nodes in H, if there no relation $-q(u)$ with $q \geq p$, then $+p(u)$ is added if it is not already present in H. We call it the *maximal positive completion of* H. If there is no homomorphism from G to H but G is deducible from H, then for each homomorphism from G to H^{c+}, there is at least one added relation in H^{c+}, say $p(u)$, such that a relation $p'(v)$ ($p' \geq p$) in G is mapped to $p(u)$. Let us replace *all* such $p(u)$ by $-p(u)$. Note that it remains a completion and it does not lead to an inconsistency: indeed, for all $+p'(u)$ with $p' \leq p$, $p'(u)$ is also inversed. Let $H^{c'}$ be this completion. Let h be a homomorphism from G to $H^{c'}$ (there is such a homomorphism since G is deducible from H). h maps a relation $-p''(w)$ in G to a relation $-p(u)$ ($p'' \leq p$), otherwise there would be a homomorphism from G to H. By construction, there is a relation $p'(v)$ mapped to $p(u)$ by a homomorphism from G to H^{c+}, thus $p'(v)$ and $-p''(w)$ (one has $p' \geq p''$) are exchangeable relations, which contradicts the hypothesis on G. □

We thus obtain a case for which PG-Deduction has the same complexity as homomorphism checking (and is thus NP-complete):

Property 5. Let G and H be two PGs, with G having no pair of exchangeable relations w.r.t. H. G is deducible from H if and only if there is a homomorphism from G to H.

Note also that G is deducible from H if and only if each connected component of G is deducible from H. Thus in previous property, the condition on G can be replaced by "each connected component of G has no pair of exchangeable relations".

If G is acyclic (and more generally has bounded treewidth, or bounded hypertreewidth when seen as a hypergraph) then homomorphism checking is polynomial ([MC92] for acyclicity, and f.i. [GLS01] for more general notions), hence PG-Deduction.

A desirable property is that recognizing exchangeable relations is not difficult compared to checking PG-deduction. It is indeed the case: checking whether G has exchangeable relations, or checking whether a pair of relations in G is exchangeable, is NP-complete [Tho07]. More precisely, it has the same complexity as homomorphism checking (from G to H), and is polynomial when G is acyclic.

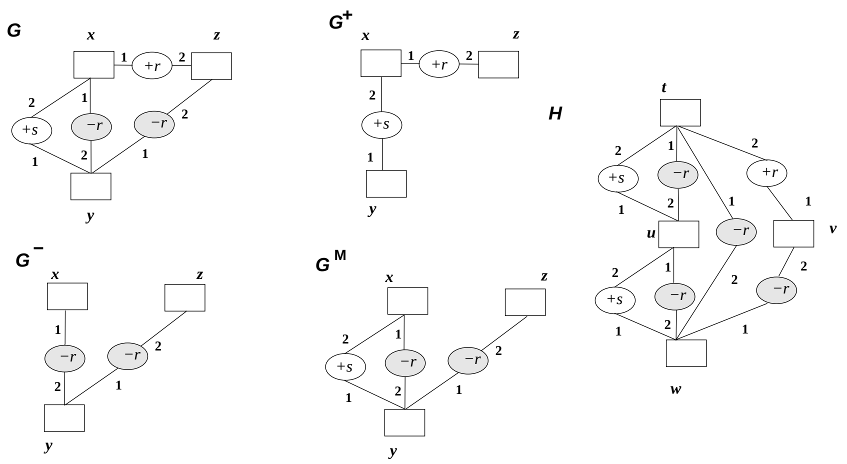
The following property will be used to prove other properties:

Property 6. Let G and H be two PGs, where H is consistent and G is deducible from H. Let G' be a subgraph of G without a pair of exchangeable relation nodes. Then there is a completion H^c of H and a homomorphism from G to H^c that maps G' entirely to H.

Proof. Consider any H^c maximal completion of H (for each n-ary type p and each n-tuple u of concept nodes in H, either one has $+p(u)$ or one has $-p(u)$). Assume that there is no homomorphism from G to H^c that maps G' to H. For each homomorphism from G to H^c, there is at least one added relation $\sim p(u)$ in H^c which is image of a relation in G' and such that H does not contain a node $\sim q(u)$ with $\sim p \geq \sim q$. Let R be the set of *all* such relation nodes in $H^c \setminus H$ for all homomorphisms from G to H^c. Let us inverse the polarity of the nodes in R. The graph obtained cannot be inconsistent[1] and it is of max size: thus it is again a maximal completion of H. Let $H^{c'}$ be this maximal completion. As G' does not possess exchangeable relations, there is no homomorphism from G to $H^{c'}$ that maps a relation node in G' to a node in R. If there is no homomorphism from G to $H^{c'}$ that maps G' entirely to H, let R' be the set of all relation nodes $\sim p(u)$ in $H^{c'} \setminus H$ which are images of a node in G' and such that H does not contain a node $\sim q(u)$ with $\sim p \geq \sim q$. As previously, reverse the polarity of all nodes in R', which yields the graph $H^{c''}$. Add the nodes of R' to R. We thus build a sequence of maximal completions of H and a set R of relation nodes of these completions not belonging to H (nor redundant with nodes of H). As R grows strictly from one completion to another, this sequence is finite. The last graph of this sequence is a completion satisfying the property. □

[1] Indeed, assume we obtain two contradictory relation nodes $-q(u)$ and $p(u)$, with $q \geq p$. One of these nodes does not belong to R, otherwise G would have exchangeable nodes. Let x be this node and y be the node that belongs to R. The label of x in H^c is necessarily more general than the label of y in H^c (note that both nodes were comparable and had the same polarity in H^c). Thus, by inverting the label of y, it is impossible to obtain an inconsistency.

If G is deducible from H, for each subgraph of G without exchangeable relations, there must be a homomorphism from this subgraph to H. Moreover, there must be such a homomorphism that is potentially extensible to a homomorphism from the entire G to a completion of H. We call it a *compatible* homomorphism. See Figure 4: all concept nodes are assumed to have the same label $(\top, *)$ and relation types are incomparable. There are three homomorphisms from G^- to H: $h_1 = \{x \to t, y \to u, z \to w\}$, $h_2 = \{x \to t, y \to w, z \to v\}$, $h_3 = \{x \to u, y \to w, z \to v\}$. To check the compatibility, we have to consider $s(y, x)$ and $r(x, z)$. h_1 is not compatible because it cannot be extended to $r(x, z)$ due to the presence of $-r(t, w)$ in H.



$G = s(y, x) \land r(x, z) \land \neg r(x, y) \land \neg r(y, z)$
$H = s(u, t) \land r(t, v) \land s(w, u) \land \neg r(u, w) \land \neg r(t, u) \land \neg r(w, v) \land \neg r(t, w)$

Fig. 4. Special subgraphs of G (G is deducible from H)

Definition 8 (compatible homomorphism). *Given two PGs G and H, and G' any subgraph of G, a homomorphism h from G' to H is said to be* compatible *(w.r.t. G) if for each relation node x of G that does not belong to G' but has all its neighbors in G', say $c_1 ... c_k$, there is no relation y on $h(c_1)...h(c_k)$ in H with a label contradictory to that of x (i.e. with label $-r$ if the label of x is $+s$, or with label $+s$ if the label of x is $-r$, s.t. $s \leq r$).*

Property 7. If G is deducible from H, then there is a compatible homomorphism from every subgraph of G without exchangeable relations to H.

Proof. Let G' be any subgraph of G without exchangeable relations. From property 6, there is a homomorphism from G to a completion of H which maps G' entirely to H. By restricting the domain of this homomorphism to G', we have a homomorphism from G' to H which is compatible w.r.t. G. □

One can remark some easily identifiable subgraphs without exchangeable relations: the *positive subgraph* of G, denoted G^+, is the subgraph obtained from G by selecting all concept nodes and only the positive relation nodes. The *negative subgraph* G^- of G is the dual notion, i.e. the subgraph obtained from G by selecting all concept nodes in G and only the negative relation nodes. Negative and positive subgraphs are particular cases of subgraphs without opposite relation labels. A subgraph of G without opposite relations and maximal for the inclusion is easily built by selecting for each relation type appearing in G, either all its positive occurrences or all its negative occurrences, while satisfying the following constraint: if one selects the positive (resp. negative) occurrences of a relation type r, then the same choice must be done for all subtypes (resp. supertypes) of r.

Example. In Fig. 4, several subgraphs without opposite relations of G are pictured. G^+ and G^- are respectively the positive and negative subgraphs of G. G has two subgraphs without opposite nodes maximal for inclusion: G^+ and G^M.

4 Algorithmic Improvements

Let us say that a concept or relation label l_x occurring in G has a support in H if there is a label l_y in H with $l_y \leq l_x$ (and l_y is said to support l_x). A first observation is that if a node in G has no support in H then G is not deducible from H. This is trivial for concept nodes. For relation nodes, if this node is negative (resp. positive), consider the positive (resp. negative) completion of H. There is no homomorphism from G to this completion.

4.1 Limitation of the Completion Vocabulary

Let us call "completion vocabulary" the set of relation types used to build completions of G. The size of the completion vocabulary determines the number of completions of G. The number of completions of G is itself a key element in the complexity of deduction checking. It is thus essential to decrease as much as possible the number of relation types involved in completion. One can observe that the completion vocabulary can be restricted to the relation types occurring in G, and furthermore to the relation types occurring in opposite relations:

Property 8. G is deducible from H iff G can be mapped to each completion of H w.r.t. relation types occurring in opposite relations of G (i.e. r and s such that there are nodes in G with labels $+r$ and $-s$ and $s \leq r$).

Proof. Let T_R be the set of relation types in the vocabulary, and let T be the set of relation types occurring in opposite relations in G. ($\Leftarrow$) We prove that if G can be mapped to each T-completion of H then it can be mapped to each T_R-completion of H. Indeed, let H^c be any T_R-completion of H. Let $H^{c'}$ be the graph obtained from H^c by replacing all relations with types outside T with a set of relations built as follows: let r be a node labeled by $(+t)$ (resp. $(-t)$)

such that $t \notin T$. Let $\{t_1 ... t_n\}$ be the types in T greater than t (resp. less than t). If r is positive, consider the minimal elements of this set, otherwise consider the maximal elements of this set. Let S be the obtained set. Replace r with $|S|$ relation nodes, each labeled by a type in S, with the same polarity and the same arguments as r. $H^{c'}$ is a T-completion of H. By construction, there is a homomorphism, say h_1, from $H^{c'}$ to H^c (which is the identity on concept nodes). By hypothesis, there is a homomorphism, say h, from G to $H^{c'}$. The composition of these homomorphisms $h_1 \circ h$ is a homomorphism from G to H^c.

($\Rightarrow$) Let G be deducible from H and assume that H^c is a T-completion of H such that there is no homomorphism from G to H^c. We show that this assumption leads to a contradiction. From H^c, we build the following T_R-completion of H, say $H^{c'}$. For all type t in $T_R \setminus T$, let us add only $(+t)$ nodes if $(+t)$ does not support any node in G; otherwise, add only $(-t)$ nodes if $(-t)$ does not support any node in G (if neither $(+t)$ nor $(-t)$ support nodes in G, relation nodes typed t can be added with any polarity); if both $(+t)$ and $(-t)$ support nodes in G, there are opposite nodes in G with label $(+r)$ and $(-s)$ and $r \geq t \geq s$, thus r and s belong to T, and nodes labeled T would be redundant in $H^{c'}$ thus are not needed to obtain a completion. In all cases, nodes are added only if they do not lead to an inconsistency. Since G is deducible from H, there is a homomorphism from G to $H^{c'}$. By construction, no node in G can be mapped to an added node. Thus this homomorphism is a homomorphism from G to H^c, which contradicts the hypothesis on H^c. □

We can even restrict the vocabulary completion to the relation types of *exchangeable* relations in G.

Theorem 2. *G is deducible from H iff G can be mapped to each completion of H w.r.t. relation types occurring in exchangeable relations of G w.r.t. H (i.e. relation types r such that there is a pair of exchangeable relations in G with one of the two labeled $+r$ or $-r$).*

Proof. Let $exchangeable(G)$ denote the types occurring in exchangeable relations in G. ($\Leftarrow$) Same as in the proof of Property 8, where T is replaced by $exchangeable(G)$.

($\Rightarrow$) Let H^c be a completion of H w.r.t. exchangeable(G) such that there is no homomorphism from G to it. As in the proof of Property 8, we build a completion of H, say $H^{c'}$, as follows: for any type t occurring in G but not in exchangeable(G), let us add only $(-t)$ nodes if t supports only positive nodes, $(+t)$ nodes if t supports only negative nodes. Let us add it positively if it supports both forms. A homomorphism from G to $H^{c'}$ is a homomorphism to H^c plus the nodes added positively for types supporting both forms. Let us now inverse the polarity of these latter nodes if they are images of nodes in G. No node in G can be mapped to these nodes, otherwise their type would be in exchangeable(G). Thus, we have a homomorphism from G to H^c, which contradicts the hypothesis on H^c. □

4.2 Space Algorithm

Consider the space of graphs leading from H to its completions. All graphs in this space have the same set of concept nodes. The space is ordered as follows: given two graphs H_1 and H_2 in this space, $H_2 \leq H_1$ if for each relation x in H_1, there is a relation with the same list of neighbors in H_2 and a label less or equal to the label of x. The question "is there a homomorphism from G to each completion H^c" can be reformulated as follows "is there a *covering set* of completions, that is a subset of incomparable graphs of this space $\{H_1, ..., H_k\}$ such that (1) there is a homomorphism from G to each H_i ; (2) for each H^c there is a H_i with $H^c \leq H_i$".

The brute-force algorithm (Algorithm 1) takes the set of all completions of H as covering set.

The next algorithm (Algorithm 2 and recursive search Algorithm 3) searches the space in a top-down way starting from H and tries to build a covering set with partial completions of H. Reasoning by cases is applied at each step: for a given relation type r with arity k and a tuple $(t_1...t_k)$ of concept nodes such that neither $+r$ nor $-r$ is supported by a relation node on $(t_1...t_k)$ in the current partial completion, two graphs are generated according to each case. Note that if r or $-r$ is supported by a $\sim s$ in the current completion, then adding $+r(t_1...t_k)$ or $-r(t_1...t_k)$ to it would lead to a redundancy or inconsistency.

The algorithm is justified by the following property:

Theorem 3. *G is deducible from H if and only if:*
1. There is a homomorphism h from G to H **or**
2. G is deducible from H' and H'' where H' (resp. H'') is obtained from H by adding the positive relation $r(t_1...t_k)$ (resp. the negative relation $-r(t_1...t_k)$) where r is a relation type of arity k occurring in G (and more specifically r belongs to the completion vocabulary) and $t_1...t_k$ are concept nodes of H such that neither $+r$ nor $-r$ is supported by a relation node on $(t_1...t_k)$ in H.

Proof. (sketch) ($\Rightarrow$) Any completion of H' or H'' is a completion of H. ($\Leftarrow$) Condition 1 corresponds to property 2. For condition 2, check that $\{H', H''\}$ is a covering set (of completions of H). $\square$

Subalgorithm 3 is supposed to have direct access to data available in the main algorithm 2. The choice of r and $t_1...t_k$, in Algorithm 3, can be guided by a compatible homomorphism from a special subgraph of G.

The Filtering subalgorithm performs "simple" tests corresponding to necessary or sufficient conditions of deduction that would allow us to conclude without entering the completion steps:

1. If a concept or relation node of G has no support in H, then return false.
2. If there is a homomorphism from G to H, then return true.
3. Compute some subgraphs of G without exchangeable relations (for instance a subgraph without opposite relations maximal for the inclusion). If one of these subgraphs does not map to H by a compatible homomorphism then return false.

Algorithm 2: Check by space exploration

```
Data: Consistent graphs $H$ and $G$
Result: true if $G$ is deducible from $H$, false otherwise
begin
    Result ← Filtering();
    if (Result ≠ undetermined) then
        return Result
    Let $\mathcal{R}$ be the completion vocabulary;
    return RecCheck(H); // See Algorithm 3
end
```

Algorithm 3: RecCheck(H)

```
Data: Consistent graph $H$        Access: $G$, $\mathcal{R}$
Result: true if $G$ is deducible from $H$, false otherwise
begin
    if there is a homomorphism from $G$ to $H$ then
        return true
    if $H$ is complete w.r.t. $\mathcal{R}$ then
        return false
    $(r, t_1...t_k)$ ← ChooseRelationTypeToAdd;
    /* $r$ is a relation type of $\mathcal{R}$, $t_1...t_k$ are concept nodes in $H$ and
       neither $+r$ nor $-r$ is supported by a relation on $(t_1...t_k)$ in $H$
       */
    Let $H'$ be obtained from $H$ by adding the relation node $r(t_1...t_k)$;
    Let $H''$ be obtained from $H$ by adding the relation node $-r(t_1...t_k)$;
    return (RecCheck(H') AND RecCheck(H''))
end
```

The following property ensures that Algorithm 3 does not generate the same graph several times, which is a crucial point for complexity. Otherwise the algorithm could be worse than the brute-force algorithm in the worst-case.

Property 9. The subspace explored by Algorithm 3 is a (binary) tree.

Indeed, at each recursive call, $\{H',\ H''\}$ is a covering set inducing a bipartition of the covered space: each graph in this space is below exactly one of these two graphs.

Property 10. The time complexity of Algorithm 2 is in $\mathcal{O}(2^{(n_G)^k \times |\mathcal{R}|} \times hom(G, H^c))$, where n_G is the number of concept nodes in G, k is the maximum arity of a relation, $\mathcal{R}$ is the completion vocabulary and $hom(G, H^c)$ is the complexity of checking the existence of a homomorphism from G to H^c. Its space complexity is in $\mathcal{O}(max(size(G), size(H), (n_G)^k \times |\mathcal{R}|))$.

Proof. The size of a completion of H is bounded by $2^{(n_G)^k \times |\mathcal{R}|}$. Property 9 ensures that the number of graphs generated is at most twice the number of completions of H (in the worst case, all leaves of the generated tree of graphs correspond to complete graphs). If the relation types are not ordered, all completions have the same size, which is $\sum_{r\in\mathcal{R}}(n_G)^{arity(r)}$; checking whether a graph is complete can then be done in constant time if the number of relation nodes in the graph is incrementally maintained. When relation types are ordered, the size of completions vary according to the order in which relation types are considered. One solution is to count the addition of a relation node $\sim r(t_1...t_k)$ not for one, but for n, where n is the number of types s in $\mathcal{R}$, such that $\sim s$ is supported by the new node and was not before. Computing n at each node addition can be roughly bound by $|\mathcal{R}|^2$, which can be reasonably considered as less than $hom(G, H^c)$. For space complexity, see that the tree is explored in depth-first way. □

To summarize, the proposed algorithm is simple to describe and to implement. Its theoretical worst-case complexity is not better than that of the brute-force algorithm but, not surprisingly, first experiments show that its running time is much better. Further work will involve an experimental comparison of several heuristics. These heuristics concern in particular the choice of special subgraphs without exchangeable relation nodes in the filtering phase and the choice of the next relation to add in the completion phase (cf. the ChooseRelationTypeToAdd subalgorithm).

References

[BM02] J.-F. Baget and M.-L. Mugnier. The Complexity of Rules and Constraints. *JAIR*, 16:425–465, 2002.

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

[Dau03] F. Dau. *The Logic System of Concept Graphs with Negation And Its Relationship to Predicate Logic*, volume 2892 of *Lecture Notes in Computer Science*. Springer, 2003.

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

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

[Kli05] Julia Klinger. Local negation in concept graphs. In *ICCS*, pages 209–222, 2005.

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

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

[MC92] M.L. Mugnier and M. Chein. Polynomial algorithms for projection and matching. In H. D. Pfeiffer and T. E. Nagle, editors, *Proc. the 7th Workshop on Conceptual Graphs*, volume 754 of *LNAI*, pages 49–58, New Mexico State University, Las Cruces, New Mexico, 1992. Springer.

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

[Mug00] M.-L. Mugnier. Knowledge Representation and Reasoning based on Graph Homomorphism. In *Proc. ICCS'00*, volume 1867 of *LNAI*, pages 172–192. Springer, 2000.

[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.

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

[Tho07] M. Thomazo. Complexité et propriétés algorithmiques de la déduction dans les graphes polarisés. First year research report, ENS Cachan, 2007.

[WL94] M. Wermelinger and J.G. Lopes. Basic conceptual structures theory. In M.-L. Mugnier and M. Chein, editors, *Proc. ICCS'98*, volume 835 of *LNAI*, pages 144–159. Springer, 1994.