RAPPORT DE RECHERCHE

On Querying Simple Conceptual Graphs With Negation

Marie-Laure Mugnier and Michel Leclère

13/07/2005 R.R.LIRMM 05-051

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

On Querying Simple Conceptual Graphs with Negation

Marie-Laure Mugnier Michel Leclère

LIRMM, University of Montpellier,
161, rue Ada, F-34392 Montpellier cedex - France
{mugnier, leclere}@lirmm.fr

Abstract

We consider basic conceptual graphs, namely *simple conceptual graphs (SGs)*, which are equivalent to the existential conjunctive positive fragment of first-order logic. The fundamental problem, *deduction*, is performed by a graph homomorphism, called *projection*. The existence of a projection from a SG Q to a SG G means that the knowledge represented by Q is deducible from the knowledge represented by G. In this framework, a knowledge base is composed of SGs representing facts and a query is itself a SG. We focus on the issue of querying SGs, which leads to consider another problem as being fundamental, namely *query answering*. Each projection from a query to a fact defines an *answer* to the query, an answer being itself a SG. The query answering problem asks for all answers to a query.

This paper introduces *atomic negation* into this framework. Several understandings of negation are explored, which are all of interest in real world applications. In particular we focus on situations where, in the context of incomplete knowledge, classical negation is not satisfactory because deduction can be proved but there is no answer to the query. We show that intuitionistic deduction captures the notion of an answer and can be solved by projection checking. Algorithms are provided for all studied problems. They are all based on projection. As a consequence, they can be combined to deal with several kinds of negation simultaneously. Relationships with problems on conjunctive queries in databases are recalled and extended. Finally, we point out that this discussion can be put in the context of the semantic web databases.

1 Introduction

Conceptual graphs (CGs) are a knowledge representation formalism mathematically founded both on logics and graph theory [Sow84]. In this paper we focus on basic conceptual graphs, called *simple conceptual graphs* (SGs). The essential characteristics of SGs are the following:

- Knowledge is represented by *labeled graphs*, which describe entities and relationships between these entities.
- Reasoning is performed by graph operations, mainly a labeled graph homomorphism called *projection*. Intuitively the existence of a projection from a SG Q to a SG G means that the knowledge represented by Q is contained in the knowledge represented by G.
- SGs are logically founded, projection being sound and complete w.r.t. *deduction* in first-order logic (FOL).

Although of limited expressiveness simple conceptual graphs can be seen as a keystone of CGs for several kinds of reasons. With knowledge-based systems in mind, an essential point is that a SG can be interpreted by an end-user (at least if its drawing is not too big and complex). And a projection is understandable too, for two reasons: it operates directly on the pieces of knowledge defined by the user and it can be visualized in a natural way. Thus objects *and* reasonings are at the user's level. On the other hand, CGs do not suffer from the lack of formal semantics which was the main criticism made to first semantic networks [Woo75] [Bra77]. Besides these modeling qualities, SGs also have computational qualities. Indeed labeled graph homomorphism (or, equivalently, relational structure homomorphism) is a fundamental notion in computer science. Using this notion, it can be shown that projection checking is essentially the same problem as the constraint satisfaction problem (CSP) or the conjunctive query containment (CQC) problem in databases (see [KV00] for the equivalence between CSP and CQC, [Mug00] for a synthesis on problems equivalent to projection checking and section 3.4 of this paper for details about relationships with problems on queries in databases). This allows to benefit from efficient algorithms developed in these domains, e.g. filtering techniques [Bag03].

The fundamental problem on SGs, called *deduction*, takes as input two SGs and asks whether there is a projection from the first one to the second one. In this paper we focus on querying SGs, which leads to consider another problem as being fundamental, namely *query answering*. The query answering problem takes as input a knowledge base (KB) composed of SGs representing facts and a SG Q,

which represents a query, and asks for all answers to Q in the KB. Each projection from Q to a fact leads to an *answer*. If we consider the query answering problem in its decision form ("is there an answer to Q in the KB?") we obtain the deduction problem ("is Q deducible from the KB?").

Querying SGs with negation. SGs express conjunction of positive knowledge. They are equivalent to the positive conjunctive existential fragment of FOL [BM02]. The question tackled in this paper is the introduction of a restricted form of negation into this framework, namely atomic negation (in logical terms, negation of form $\neg p$, where p is an atom). Atomic negation allows to express knowledge like "this kind of relation does not hold between these entities", "this entity does not have this property" or "this entity is not of this type". This question is studied from a semantic viewpoint (what does negation mean?) and from a computational viewpoint (what is the complexity of the main problems?). An important guideline is also to keep this extension as much as possible at the end-user level.

Let us point out that general conceptual graphs include negation, but in a way that does not suit our needs. General CGs are obtained from SGs by adding boxes representing negation, lines representing equality, and diagrammatic derivation rules, following Peirce's existential graphs (see [Sow84] for the original ideas, [WL94] for a logically sound and complete formalization, [Dau02] for another formalization called "concept graphs"). They are equivalent to FOL thus deduction becomes undecidable. Furthermore, notwithstanding the qualities general CGs might have for diagrammatic logical reasoning, they are not at the end-user level (see f.i. [BBV97]). As a matter of fact most applications are based on SGs and extensions keeping their intuitive appeal such as nested graphs, rules and constraints (f.i. the $\mathcal{SG}$-family in [BM02]). Notice that these latter extensions do not provide negation.

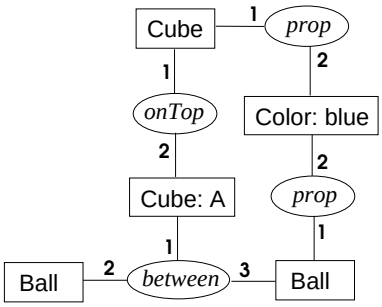
Results. Several understandings of negation are explored in this paper, which are all of interest in real world applications. Briefly, when a query asks "find the x and y such that *not* $r(x, y)$", "not" can be understood in several ways. It might mean "the knowledge $r(x, y)$ cannot be proved" or "the knowledge *not* $r(x, y)$ can be proved". The first view is consistent with the closed-world assumption, the second one with the open-world assumption. In turn, the notion of proof can have several meanings. We point out that, as soon as negation is introduced, the classical FOL deduction problem is no more equivalent with the query answering problem in its decision form. Indeed, there are cases where classical deduction can be proved but no answer can be exhibited. These situations exactly correspond to cases where the law of excluded middle ("either A is true or *not* A is true") is used in the proof.

This observation leads to consider intuitionistic logic, in which the law of excluded middle does not hold. It is shown that intuitionistic deduction exactly captures the notion of an answer. Let us add that, independently of the notion of answer, there are many cases in real world applications where the law of excluded middle is not desirable, because neither (A) nor $(not\ A)$ might hold. It is the case for instance for a property like "being a smoker": some people smoke occasionally, and cannot be considered as smokers neither as non smokers. This point is not developed in the present paper but we shall come back to it in the perspectives.

Basic algorithms are provided for deduction and query answering and for each studied kind of negation. All algorithms use projection as the basic notion. In particular, it is proved that projection is sound and complete with respect to intuitionistic deduction in the logical fragment corresponding to SGs with atomic negation. It follows that atomic negation can be introduced in SGs with no overhead cost for the query answering problem. These algorithms can be combined to deal with several kinds of negation simultaneously.

The tight relationships with query problems in databases are recalled and extended to SGs with negation. Finally, let us point out that our questions and results are potentially transferable to semantic web databases. Indeed, SGs can express RDFS in natural ways as shown by several authors (see the last section of this paper) and the essential task in the RDFS context is to query knowledge. The issue of extending RDFS with some kinds of negation, which seems unavoidable since all existing knowledge-based and database systems do provide a form of negation, is thus directly related to the topic of this paper.

Paper organization. The paper is organized as follows. Section 2 synthesizes some basic definitions and results about SGs, and presents the query framework. In section 3 SGs are extended with atomic negation, leading to *polarized* SGs. Several understandings of negation are discussed and related with the projection notion. Basic algorithms for solving all the query problems are provided. The equivalences with problems on conjunctive queries in databases are pointed out. This section ends with complexity issues. In section 4 perspectives are outlined in relation with connected works. Proofs related to intuitionistic logic are given in Appendix.



G

Figure 1: Simple conceptual graph. The generic marker * is omitted in drawings. A concept label (t, m) is written t:m.

2 Basic Notions: Simple Conceptual Graphs

In this section we synthesize basic definitions and results, mainly published in [CM92] and [BM02]. SGs are the basic constructs. Standing alone they represent queries and facts, but are building blocks for more complex kinds of knowledge, as constraints and rules [BM02].

2.1 Syntax

All kinds of knowledge are defined with respect to a vocabulary called a *support*.

Definition 1 (Support) *A* support *is a 4-tuple* $\mathcal{S} = (T_C, T_R, \mathcal{I}, \tau)$. T_C *and* T_R *are two partially ordered finite sets, respectively of* concept types *and* relation types. T_C *possesses a greatest element, called the universal type, and denoted by* $\top$. *Relation types may be of any arity greater or equal to 1. Only relation types with same arity are comparable.* $\mathcal{I}$ *is the set of* individual markers. T_C, T_R *and* $\mathcal{I}$ *are pairwise disjoint.* τ *is a mapping from* $\mathcal{I}$ *to* T_C. *We denote by* $*$ *the* generic marker, *where* $* \notin \mathcal{I}$. *The set* $\mathcal{I} \cup \{*\}$ *is partially ordered in the following way:* $*$ *is the greatest element and elements of* $\mathcal{I}$ *are pairwise non comparable.*

The partial orders on types are interpreted as specialization relations ($t \leq t'$ is read as t *is a specialization of* t'). In the sequel we will sometimes consider supports with trivial partial orders, i.e. where distinct types are not comparable. We call them *flat* supports.

A *simple conceptual graph (SG)* has two kinds of nodes: *concept nodes* represent entities of the application domain and *relation nodes* represent relationships between these entities.

Definition 2 (Simple Graph) *A* simple graph *(SG) G, defined over a support $\mathcal{S}$, is a 4-tuple $(C_G,\ R_G,\ E_G,\ l_G)$ such that:*

- *$(C_G \cup R_G, E_G)$ is a bipartite multigraph (there may be several edges between two nodes). C_G and R_G are the (disjoint) node sets, respectively of* concept nodes *and of* relation nodes. *E_G is the multiset of* edges.
- *l_G is a labeling function of the nodes and edges defined as follows. A concept node c is labeled by a couple* (type(c), marker(c)), *where* type(c) *is an element of T_C, called its* type, *and* marker(c) *is an element of $\mathcal{I} \cup \{*\}$, called its* marker. *If marker(c) is an individual marker m then $type(c) = \tau(m)$. A relation node r is labeled by* type(r), *an element of T_R, called its* type, *and the degree of r (i.e. the number of edges incident to r) must be equal to the arity of type(r). Edges incident to a relation node r are totally ordered; there are labeled from 1 to the degree of r.*

Notations. An edge labeled by i connecting a relation node r and a concept node c is denoted by (r, i, c). $r = t_r(c_1,\ ...,\ c_k)$ is a short notation for a relation node r with type t_r and incident edges $(r, 1, c_1)$ ... (r, k, c_k). Notice that $c_1,\ ...,\ c_k$ are not necessarily distinct nodes.

A concept node with an individual marker represents a specific individual, it is called an *individual node* (e.g. nodes with labels $(Cube, A)$ and $(Color, blue)$ in figure 1 respectively refer to *the* cube A and *the* color blue). Otherwise it represents any entity of a given type and is called a *generic node* (e.g. the node with label $(Cube, *)$ in figure 1 represents *a* cube). In figure 1 the SG G can be intuitively interpreted as expressing that "there is a cube on top of the cube A, which is between balls; the first cube and one ball both have color blue".

2.2 Logical translation of SGs

SGs are provided with a semantics in FOL by mean of a mapping called Φ [Sow84]. Elements (T_C, T_R and $\mathcal{I}$) of a support $\mathcal{S}$ are translated into elements of a logical language $\mathcal{L}$ as follows: types are mapped to predicates with same arity and individual markers to constants. For simplicity, we consider that each constant or predicate has the same name as the associated element of the support.

A set of formulas $\Phi(\mathcal{S})$ is assigned to any support $\mathcal{S}$, translating the partial orders on types. More specifically, for all types t_1 and t_2 such that $t_1 \geq t_2$, one has the formula $\forall x_1 ... x_k (t_2(x_1,\ ..., x_k) \rightarrow t_1(x_1,\ ..., x_k))$, where $k = 1$ for concept types, and k is otherwise the arity of the relation types.

Given any SG G, a formula $\Phi(G)$ is built as follows. A term is assigned to each concept node: a distinct variable for each generic node, and the constant

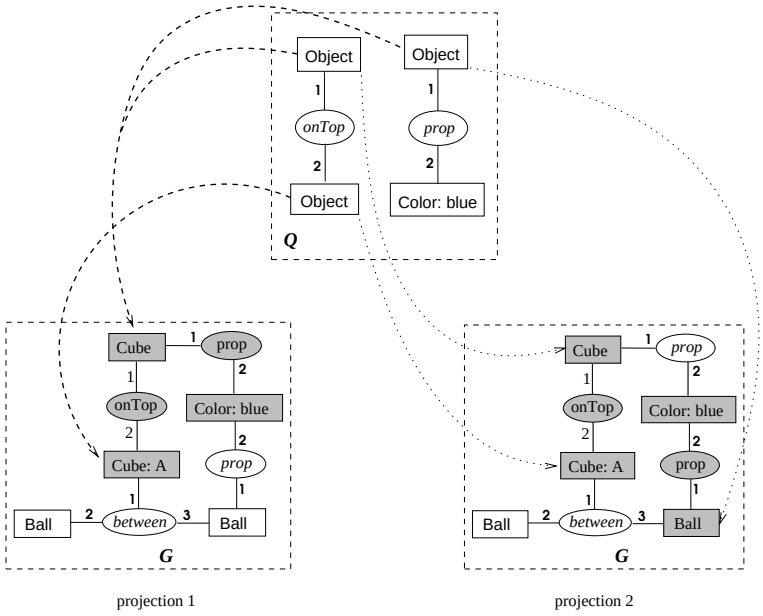


Figure 2: The projections from Q to G

corresponding to its marker otherwise. Then an atomic formula $t(e)$ is associated to each concept node, where t is the type of the node, and e is the term assigned to this node. Similarly, an atomic formula $t(e_1, \ldots, e_k)$ is assigned to each relation node $r = t(c_1, \ldots, c_k)$, e_i denoting the term assigned to c_i. Let $\alpha(G)$ be the conjunction of these atomic formulas. $\Phi(G)$ is the existential closure of $\alpha(G)$. E.g. the formula assigned to G in figure 1 could be $\exists x \exists y \exists z\ (Cube(x) \wedge Ball(y) \wedge Ball(z) \wedge Cube(A) \wedge Color(blue) \wedge onTop(x, A) \wedge prop(x, blue) \wedge prop(y, blue) \wedge between(A, z, y))$.

2.3 Deduction

The basic notion for doing reasoning is a graph homomorphism called *projection*. Let us recall that a homomorphism h from a graph G_1 to a graph G_2 is a mapping from the nodes of G_1 to the nodes of G_2 which preserves edges, i.e. if xy is an edge of G_1 then $h(x)h(y)$ is an edge of G_2. Intuitively, the existence of a projection from a SG Q to a SG G means that the knowledge represented by Q is contained in the knowledge represented by G.

Definition 3 (Projection) *Let $Q = (C_Q, R_Q, E_Q, l_Q)$ and $G = (C_G, R_G, E_G, l_G)$ be two SGs defined on a support $\mathcal{S}$, a* projection *from Q to G is a mapping π from*

C_Q to C_G and from R_Q to R_G that preserves edges (it is a graph homomorphism) and may specialize the labels of concept and relation nodes:

1. $\forall (r, i, c) \in Q, (\pi(r), i, \pi(c)) \in G$*;*
2. $\forall x \in C_Q \cup R_Q$, $l_G(\pi(x)) \leq l_Q(x)$ *(if x is a concept node, $\leq$ is the product of the orders on T_C and $\mathcal{I} \cup \{*\}$, i.e. $type(\pi(x)) \leq type(x)$ and $marker(\pi(x)) \leq marker(x)$; if x is a relation node, $\leq$ is simply the order on T_R).*

Q is called the *source* SG while G is the *target* SG. We note $Q \succeq G$ (Q subsumes G, Q is more general than G) or $G \preceq Q$ if there is a projection from Q to G. In figure 2, Q represents the knowledge "there is an object on top of an object and a blue object". Assuming that $Cube, Ball \leq Object$, there are two projections from Q to G. Typically, Q could represent a query (for instance "find all objects x, y and z, such that x is on top of y and z is a blue object"), G a fact, and projections from Q to G define answers to Q. We shall detail this aspect later. The deduction problem on SGs is defined as follows:

Definition 4 (SG Deduction Problem) *The SG deduction problem takes as input a SG G and a SG Q (defined on the same support) and asks whether $Q \succeq G$.*

This problem is NP-complete [CM92]. SG deduction is sound and complete w.r.t. deduction in FOL (denoted by $\models$), up to a normality condition for completeness: a SG is in normal form if each individual marker appears at most once in it[1]. The *normal form* of a SG G is the SG $nf(G)$ obtained by merging concept nodes having the same individual marker. This SG always exists, is unique and is logically equivalent to the original SG.

Theorem 1 ([CM92][GW95]) *Let Q and G be two SGs defined on a support $\mathcal{S}$. Then $Q \succeq nf(G)$ if and only if $\Phi(\mathcal{S}), \Phi(G) \models \Phi(Q)$.*

In what follows, we assume that the target graph G is in normal form. However, when relationships are established with logic, we specify $nf(G)$ if G has to be in normal form.

[1]Indeed, concept nodes with the same individual marker are mapped by Φ to the same constant. Thus, a SG in normal form is logically equivalent to the SG obtained from it by splitting one of its individual nodes into several nodes with the same individual marker, whereas there is no projection from the former SG to the latter one.

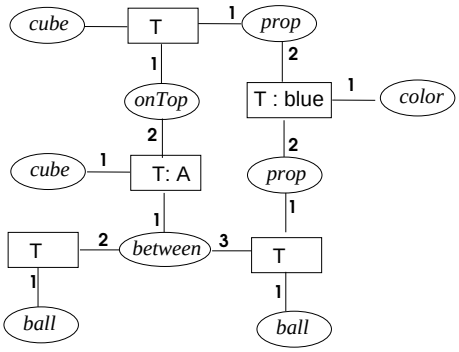


Figure 3: The f2g translation. $f = \exists x \exists y \exists z\ (Cube(x) \wedge Ball(y) \wedge Ball(z) \wedge Cube(A) \wedge Color(blue) \wedge onTop(x, A) \wedge prop(x, blue) \wedge prop(y, blue) \wedge between(A, z, y))$

Note about alternatives. Let us mention a variant of projection, which can be used to achieve completeness without normalization (see the *coref-projection* in [CM04]). The $\preceq$ relation can also be defined by a set of elementary graph operations ([Sow84, CM92] for the first definitions), in such as way that a sequence of elementary operations corresponds to a projection or a coref-projection (depending on the exact composition of the operation set). We prefer to base reasoning on a global mapping (projection/coref-projection) for modeling reasons as well as computational reasons, as explained in the introduction.

Equivalence with the existential positive fragment of FOL SGs are equivalent to the fragment of FOL composed of existentially closed conjunctive positive formulas (with constants but without functions). Let us denote by FOL$\{\exists, \wedge\}$ this fragment. The translation from FOL$\{\exists, \wedge\}$ to SGs is immediate. It maps a logical language (composed of a finite set of predicates and sets of variables and constants) into a flat support. All predicates are translated into relation types and a unique concept type $\top$ is introduced. Formulas are mapped to normal SGs. See figure 3 for an example (where f can be seen as $\Phi(G)$ for G in figure 1) and [BM02] for further developments.

Property 1 *[BM02] There is a bijection, say $f2g$, from the set of FOL$\{\exists, \wedge\}$ formulas over a language $\mathcal{L}$ to the set of normal SGs on the flat support $\mathcal{S}$ naturally associated to $\mathcal{L}$.*

For the translation in the other direction, the apparent problem is that formulas assigned to the support by Φ are universally quantified. However, we can do with-

out them. Indeed, SGs defined on a support can be polynomially transformed into SGs on a flat support while preserving the $\succeq$ relation among them [BM02].

2.4 S-substitution

We now introduce notions that will be used throughout the paper's proofs (and in proofs only): *S-substitution* from a formula to a formula, *acceptable* model and a *good assignment* from a formula to a model.

Notations. A logical language (denoted by $\mathcal{L}$) provided with a partial order on predicates (denoted by $\leq_{\mathcal{L}}$) is said to be *ordered*. Comparable predicates must have the same arity. When predicates are pairwise not comparable the language is said to be *flat*. Given an ordered language $\mathcal{L}$ we note $\mathcal{O}(\mathcal{L})$ the set of formulas associated with the partial order. $\mathcal{O}(\mathcal{L})$ is equal to $\Phi(\mathcal{S})$ where $\mathcal{S}$ is the support naturally associated with $\mathcal{L}$. A (positive) literal is denoted by $p(\vec{e})$ where p is a predicate and $(\vec{e})$ a tuple of terms.

The *S-substitution* can be seen as the logical translation of projection. When the language is flat it corresponds to the classical logical substitution. Constants are mapped for convenience.

Definition 5 (S-substitution) *Let f and g be two formulas of FOL$\{\exists, \wedge\}$ on an ordered language $\mathcal{L}$. A* S-substitution *S from f to g is a mapping from terms (variables and constants) of f to terms of g such that:*

- *for all constant a in f, $S(a) = a$;*
- *for all literal $p(\vec{e})$ in f, there is a literal $q(S(\vec{e}))$ in g such that $q \leq_{\mathcal{L}} p$.*

Property 2 (Equivalence S-substitution projection) *Let G and H be two SGs. There is a projection from G to $nf(H)$ iff there is a S-substitution from $\Phi(G)$ to $\Phi(H)$. Let f and g be two FOL$\{\exists, \wedge\}$ formulas on an ordered language. There is a S-substitution from f to g iff there is a projection from $f2g(f)$ to $f2g(g)$ (where $f2g$ is the mapping defined in section 2.3).*

More precisely, each projection π defines a S-substitution. We only need to consider the restriction of π to concept nodes. Reciprocally, to each S-substitution corresponds a set of projections which are identical on the concept nodes but can vary on relations nodes.

Let us denote by $M = (D, \delta)$ a classical FOL model of a language $\mathcal{L}$ where D is the non-empty domain and δ maps each constant of $\mathcal{L}$ to an element of D and each k-ary predicate to a subset of D^k.

Definition 6 (Acceptable model) *A classical model $M = (D, \delta)$ is* acceptable *w.r.t. an ordered language $\mathcal{L}$ if it satisfies: for all predicate p, if $(\vec{d}) \in \delta(p)$ then for all $q \geq_{\mathcal{L}} p$, $(\vec{d}) \in \delta(q)$.*

Property 3 *M is acceptable for $\mathcal{L}$ iff $M \models \mathcal{O}(\mathcal{L})$ (i.e. each formula of $\mathcal{O}(\mathcal{L})$ is true in M).*

Without loss of generality we consider that formulas in FOL$\{\exists, \wedge\}$ are in prenex form (all quantifiers precede the conjunction of literals). For this fragment, a formula f is true for a model M if there exists a special mapping from f to M, called a *good assignment*.

Definition 7 (Good assignment) *Given a FOL$\{\exists, \wedge\}$ formula f in prenex form and a model $M = (D, \delta)$, a good* assignment *of f to M is a mapping α from the terms of f to D such that:*

- *for all constant c of f, $\alpha(c) = \delta(c)$;*
- *for all literal $p(\vec{e})$, $(\alpha(\vec{e})) \in \delta(p)$.*

A proof of soundness and completeness of the S-substitution can be based on previous definitions and properties. Notice it yields a new proof of theorem 1 (soundness and completeness of projection).

Property 4 (S-substitution soundness) *Let f and g be two formulas of FOL$\{\exists, \wedge\}$ on an ordered language $\mathcal{L}$. If there is a S-substitution from f to g then $\mathcal{O}(\mathcal{L}), g \models f$.*

Proof. Let S be a S-substitution from f to g. Let $M = (D, \delta)$ be any model of $\mathcal{O}(\mathcal{L})$ and g. Let α be a good assignment of g to M. M has to be acceptable thus $\alpha \circ S$ defines a good assignment from f to M. M is thus a model of f. □

Property 5 (S-substitution completeness) *Let f and g be two formulas of FOL$\{\exists, \wedge\}$ on an ordered language $\mathcal{L}$. If $\mathcal{O}(\mathcal{L}), g \models f$ then there is a S-substitution from f to g.*

Proof. Since every model satisfying g and $\mathcal{O}(\mathcal{L})$ also satisfies f, we choose one from which we can build the desired S-substitution. Let $M = (D, \delta)$ be an acceptable model, such that D is the set of terms of g, δ is the identity over constants and for each k-ary predicate q of $\mathcal{L}$, $\delta(q)$ is given by the literals of g with predicate

$p \leq_{\mathcal{L}} q$ (i.e. $(\vec{e}) \in \delta(q)$ if there is $p(\vec{e})$ in g with $p \leq_{\mathcal{L}} q$). Let α be a good assignment of f to M. By construction of M, α is a mapping from terms of f to terms of g. Let $q(\vec{e})$ be a literal of f, one has $(\alpha(\vec{e})) \in \delta(q)$. There is thus a literal $p(\alpha(\vec{e}))$ in g with $p \leq_{\mathcal{L}} q$. It follows that α is a S-substitution from f to g. □

2.5 Queries and answers

Classical applications built upon SGs consider a base composed of SGs expressing facts and search this base by queries which are themselves SGs. The notion of *answer* can be defined in different ways depending on the application context.

In an information retrieval context, the base can be seen as a set of couples of form (G, ref) where G is a SG annotating a document and ref is a reference to this document (for instance [CG05]). In this framework, a query Q represents a piece of information searched by the user and the set of answers is the subset of couples (G, ref) in the base such that the knowledge asserted by G entails the knowledge asked by Q. A couple (G, ref) is thus an answer if there is a projection from Q to G, which corresponds to solve the SG deduction problem[2] Each SG can be searched independently.

In a knowledge-based system context, the knowledge base (KB) is simply composed of SGs expressing facts. As SGs need not to be connected graphs, these SGs can be seen as composing a single SG. In the sequel, the graph G represents the entire knowledge base. A query, hence the notion of answer, can be interpreted in different ways.

- A query Q can be seen as a representation of "yes/no question": *"Is the knowledge represented by Q asserted by the KB?"* The answer is thus a boolean, which is true if and only if there is a projection from Q to G. The problem to solve is thus the SG deduction problem on the entire base.

- A query Q can be seen as a "pattern" allowing to extract knowledge from the KB. Generic nodes in the query represent variables to instantiate with individual or generic nodes in the base. With this interpretation, each projection from Q to G defines an answer to Q. An answer can be seen as the projection itself, which to each node of the query associates a node of the base. Or it can be seen as the subgraph of G induced by this projection. We call it the *image graph* of Q by π.

[2]Notice we restrict the framework to the so-called *exact* answers, whereas information retrieval systems often extend the query mechanism to retrieve *approximate* answers.

Definition 8 (Image graph) *Let π a projection from Q to G. The image graph of Q by π, denoted by $Image(Q, \pi)$, is the subgraph of G induced by the images of the nodes in Q by π.*

The gray subgraphs of G in figure 2 are the two image graphs of Q induced by the projections indicated by dashed arrows. Note that distinct projections from Q to G may produce the same image graph, thus defining answers as image graphs instead of projections induces a potential loss of information. One advantage however of this definition of an answer is that the set of answers can be seen as a SG. We thus have the property that the results returned by a query are in the same form as the original data. This property is mandatory to process complex queries, i.e. queries composed of simpler queries. In complex query processing, the answers to a query are considered as a KB to process another query (e.g. [BK03]). This semantics of query/answering is for instance the one retained in the semantic web context in [GHM04], where the so-called *pre-answer* corresponds to our set of answers, and two ways of passing from a pre-answer to a single graph are analyzed.

The definition of the query answering problem that will be used in next sections is the following:

Definition 9 (SG query answering problem) *Let Q be a query and G be a KB. The query answering problem asks for the set of image graphs of Q by all projections to G.*

3 Different Kinds of Atomic Negation

SGs describe the world in terms of conjunctions of positive assertions. With the idea of keeping decidability of reasoning and graph operations, we are interested in a restricted form of negation, *atomic negation*. In this paper, we define negation on relations only, but as explained below the results can easily be translated to concept types.

3.1 Polarized SGs

Beside positive relation nodes, we now have negative relation nodes. A positive node is labeled by (r) or $(+r)$, and a negative one by $(-r)$, where r is a relation type. We call *polarized* SGs (**PG**s) such SGs (as in [Ker01]).

A negative relation node with label $(-r)$ and neighbors $(c_1 \ldots c_k)$ expresses that "there is not the relation r between $c_1 \ldots c_k$" (or if $k = 1$, "c_1 does not possess the property r"); it is logically translated by Φ into the literal $\neg r(e_1 \ldots e_k)$, where e_i is the term assigned to c_i.

Projection on PGs is similar to projection on SGs with a simple extension of the order on relation node labels:

Definition 10 (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 $r_1 \geq r_2$. In other words, the opposite type order is considered for negative nodes.*

Negation on concept types. Negation in concept labels can defined in a similar way. A concept node labeled by $-t$ (and a marker) is interpreted as *"there exists an entity which is not of type t"*, and not as *"there does not exist an entity of type t"*, that is we keep an existential interpretation. Since the universal concept type is supposed to represent all entities, it cannot be negated. Let us point out that, if negation on concept types is interesting from a modeling viewpoint, it does not add expressiveness. Indeed concept types can be processed as unary relation types. More precisely, consider SGs on a support $\mathcal{S}$. Let $\mathcal{S}'$ be the support built by translating all concept types, except the universal type $\top$, into unary relation types (keeping the same partial order). The concept type set of $\mathcal{S}'$ is composed of the single type $\top$. Then, SGs on $\mathcal{S}$ can be transformed into SGs on $\mathcal{S}'$, while preserving projections and logical deduction: each concept node with label $(\sim t, m)$, where $\sim t$ can be positive or negative and $t \neq top$, is translated into a concept node with label $(\top, m)$ and one neighboring relation node with label $(\sim t)$. For an example, see the transformation from the SG of figure 1 to the SG of figure 3. A simple and uniform way of processing negation on concepts and relations consists thus in applying the transformation suggested above, processing the obtained graphs with algorithms given in this paper and, if needed, applying the reverse transformation to present the results.

Relationships with FOL. Let us denote by FOL$\{\exists, \wedge, \neg_a\}$ the extension of FOL$\{\exists, \wedge\}$ to negative literals. The transformations between FOL $\{\exists, \wedge\}$ and SGs presented in section 2.3 are naturally extended to transformations between PGs and FOL$\{\exists, \wedge, \neg_a\}$. PGs are thus equivalent to the FOL$\{\exists, \wedge, \neg_a\}$ fragment. Similarly, the notions of S-substitution and good assignment defined for SGs in section 2.4 (and used in proofs) can be extended to PGs in a straightforward way:

Definition 11 (Extension of S-substitution) *Let f and g be two formulas of FOL$\{\exists, \wedge, \neg_a\}$ on an ordered language $\mathcal{L}$. A S-substitution S from f to g is a mapping from terms (variables and constants) of f to terms of g such that:*

- *for all constant a in f, $S(a) = a$;*
- *for all positive literal $p(\overrightarrow{e})$ in f, there is a positive literal $q(S(\overrightarrow{e}))$ in g such that $q \leq_{\mathcal{L}} p$.*
- *for all negative literal $\neg\, p(\overrightarrow{e})$ in f, there is a negative literal $\neg\, q(S(\overrightarrow{e}))$ in g such that $q \geq_{\mathcal{L}} p$.*

Definition 12 (Good assignment) *Given a model $M = (D, \delta)$, a good assignment α of a formula f in FOL$\{\exists, \wedge, \neg_a\}$ to M is a mapping α from the terms of f to D, which maps each constant c to $\delta(c)$, and such that, for all positive literal $p(\overrightarrow{e})$, $\alpha(\overrightarrow{e}) \in \delta(p)$ and for all negative literal $\neg p(\overrightarrow{e})$, $\alpha(\overrightarrow{e}) \notin \delta(p)$.*

It is immediately checked that the equivalence between S-substitution and projection is preserved (extension of property 2).

Property 6 (Equivalence S-substitution projection on PGs) *Let G and H be two PGs. There is a projection from G to $nf(H)$ iff there is a S-substitution from $\Phi(G)$ to $\Phi(H)$. Let f and g be two FOL$\{\exists, \wedge\neg_a\}$ formulas on an ordered language. There is a S-substitution from f to g iff there is a projection from $f2g(f)$ to $f2g(g)$ (where $f2g$ becomes the natural extension of mapping defined in section 2.3).*

Since negation is introduced, a PG can be inconsistent.

Definition 13 (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 $r \leq s$. Otherwise it is said to be* consistent.

Property 7 *For any PG G on a support $\mathcal{S}$, G is inconsistent iff $\Phi(\mathcal{S}) \cup \{\Phi(G)\}$ is (logically) inconsistent.*

Proof. ($\rightarrow$): trivial. ($\leftarrow$): see that $\Phi(\mathcal{S})$ cannot be inconsistent since it contains positive information only. Now consider the clausal form of $\Phi(\mathcal{S}), \Phi(G)$: the only way to deduce the empty clause from it is to have two clauses of form $r(\overrightarrow{e})$ and $\neg s(\overrightarrow{e})$ with same argument list $\overrightarrow{e}$ and $r \leq s$ (which allows to obtain $\neg(r(\overrightarrow{e}))$ from $\neg s(\overrightarrow{e})$ and $\Phi(\mathcal{S})$). □

This extension is straightforward but its precise semantics needs discussion. Let us consider the very simple example of figure 4. G describes a situation where there is a pile of three cubes A, B and C; A is blue and C is not blue. Whether B is blue or not is not specified. There are two classical ways of considering missing information, known as closed-world assumption (CWA) or open-world assumption

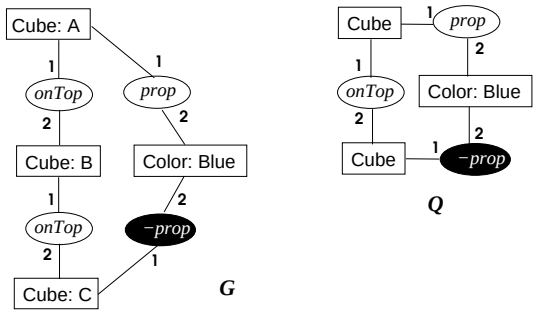


Figure 4: Atomic negation

(OWA). In the first case, knowledge about the world is supposed to be complete. Only positive information needs to be encoded in the base, negative information being obtained by difference with the content of the base. A missing information is thus considered as being false. In the second case a missing information is simply not known.

In the two following subsections we will study three ways of understanding negation in relation with the notions of *query* and *answer*. The last subsection is devoted to computational complexity considerations, after a detour to databases.

3.2 Closed-world assumption

A first way of understanding "not A" is "A is not present in the knowledge base" (and more generally A cannot be obtained from the knowledge base by the inference mechanisms). Such a view is consistent with the "closed-world assumption" generally made in databases, and the "negation by failure" in logic programming. Although only positive information needs to be represented in the base, we will not forbid a PG representing facts to contain negative relations (figure 5: G and G' encode the same knowledge).

A *completed* PG is obtained from a PG by expliciting in a negative way all missing information about relations. Then a query is not mapped to the original base but to its completed version.

Definition 14 (completed PG) *The* completed PG *of a PG G, denoted by $completed(G)$, defined over a support $\mathcal{S}$, is the unique PG obtained from the normal form of G by adding all possible negative relations: for all relation type r of arity k in $\mathcal{S}$, for all concept nodes $c_1 ... c_k$, if there is no relation $r'(c_1 ... c_k)$ in $nf(G)$ with $r' \leq r$, add the relation $-r(c_1 ... c_k)$.*

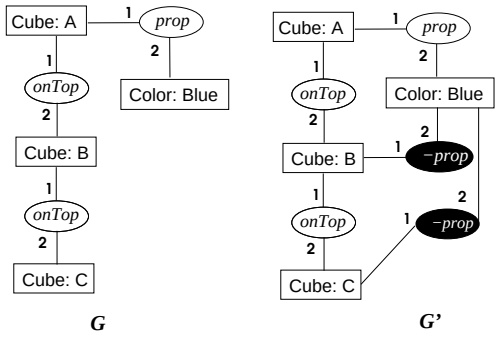


Figure 5: Closed-world assumption

Definition 15 (CWA-PG deduction problem) *The PG deduction problem with closed-world assumption semantics takes as input two PGs Q and G and asks whether $Q \succeq completed(G)$.*

The mapping to classical logical deduction is obtained via the completed base:

Property 8 *Let Q and G be two polarized SGs defined on a support $\mathcal{S}$, G being consistent. $Q \succeq completed(G)$ if and only if $\Phi(\mathcal{S}), \Phi(completed(G)) \models \Phi(Q)$.*

Definition 16 (CWA-PG query answering problem) *Let Q be a (polarized) query and G be a (polarized) KB. The query answering problem asks for the image graphs of Q by all projections to $completed(G)$.*

Obviously the completed PG (or the part of it concerning the negated relations of the query) does not have to be computed in practice. Indeed let Q^+ be the subgraph obtained from Q by considering concept nodes and solely positive relations. One only has to compute the projections from Q^+ to G and select those that do not lead to "map" a negative relation in Q to a contradictory positive relation in G.

Definition 17 *A negative relation $-r(c_1 \ldots c_k)$ from a PG Q is* satisfied *by a projection π from Q^+ to a PG G if G does not contain a positive node $+s(\Pi(c_1) \ldots \Pi(c_k))$ with $s \leq r$.*

The following property is immediately checked:

Property 9 *Let Q and G be two PGs defined over a support $\mathcal{S}$ and let us assume that G is consistent. There is a bijection from the set of projections from Q^+ to G such that each negative relation of Q is satisfied, to the set of projections from Q to $completed(G)$.*

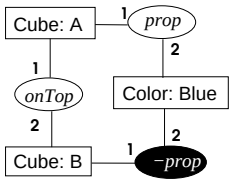


Figure 6: Single result obtained by applying CWA-PG query answering to figure 4

Algorithm 1 for deduction and algorithm 2 for query answering take advantage from this property. In the second algorithm, $Ans(Q, \pi)$ is the PG obtained from the image graph of Q^+ by π by adding negative relations corresponding to negative relations in Q (i.e. for each $-r(c_1 \ldots c_k)$ in Q, one adds $-r(\pi(c_1) \ldots \pi(c_k))$ to $\pi(Q^+)$). In other words, $Ans(Q, \pi)$ is the image of Q by a projection (extending π) to $completed(G)$ and not to G. Indeed, the closed-world assumption cannot be made on answer graphs: the absence of a relation in an answer graph would not imply its absence in the KB, it could come from its absence in the query. For instance, consider the figure 6, which shows the unique answer obtained by processing the query Q to the KB G in figure 4. The relation of label $(-prop)$ is added whereas it does not appear in G.

Algorithm 1: CWADeduction

Data: PGs Q and G
Result: true if Q can be deduced from G with CWA, false otherwise
begin
  Compute P the set of projections from Q^+ to G;
  forall $\pi \in P$ **do**
    Good $\leftarrow$ true;
    forall *negative relation* $-r(c_1 \ldots c_k)$ *in* Q **do**
      if *there is* $s(\pi(c_1) \ldots \pi(c_k))$ *in* G *with* $r \leq s$ **then**
        `// π is not good`
        Good $\leftarrow$ false;
        exit *this for loop* ;
    if *Good* **then return** *true*;
  return *false*;
end

Algorithm 2: CWAQueryAnswering

Data: PGs Q and G
Result: the set of answers to Q in G with closed-world assumption
begin
  Compute P the set of projections from Q^+ to G;
  $Answers \leftarrow \emptyset$;
  forall $\pi \in P$ **do**
    Good $\leftarrow$ true;
    forall *negative relation* $-r(c_1 \ldots c_k)$ *in* Q **do**
      if *there is* $s(\pi(c_1) \ldots \pi(c_k))$ *in* G *with* $r \leq s$ **then**
        `// π is not good`
        Good $\leftarrow$ false;
        exit *this for loop* ;
    if *Good* **then** $Answers \leftarrow Answers \cup \{Ans(Q, \pi)\}$;
  return *Answers*;
end

3.3 Open-world assumption

Let us now interpret the example of figure 4 with open-world assumption: nothing is known about the color of the cube B. Seen as a yes/no question, Q asks *whether* there is a blue cube on top of a non blue cube. Seen as a query, Q asks for *exhibiting* objects having these properties. In both cases, what should be answered to Q?

Let us first point out that spontaneously a non-logician (an end-user for instance) would say that the answer to the yes/no question is *no*. This intuition corresponds to the observation that there is no answer to the query. However in classical FOL the answer to the yes/no question is *yes*. Indeed the logical formulas assigned to Q and G by Φ are respectively of form $\Phi(Q) = \exists x \exists y\ (p(x, Blue) \wedge \neg p(y, Blue) \wedge r(x, y))$ and $\Phi(G) = p(A, Blue) \wedge r(A, B) \wedge r(B, C) \wedge \neg p(C, Blue)$ (where $p = prop$, $r = onTop$ and atoms assigned to concept nodes are ignored). $\Phi(Q)$ can be deduced from $\Phi(G)$ using the valid formula $p(B, Blue) \vee \neg p(B, Blue)$ (every model of $\Phi(G)$ satisfies either $p(B, blue)$ or $\neg p(B, blue)$; $\Phi(Q)$ is obtained by interpreting x and y as B and C if $p(B, blue)$ holds, and as A and B in the opposite case). Classical deduction thus ensures there exists a "solution" to Q but it is not able to *construct* it. Hence there is no answer to Q as a query.

This example leads to the following observations:

- The assertions "Q is (classically) deducible from G" and "the set of answers

to Q in G is not empty" might disagree. In other words, deduction and the decision problem associated with query answering are different problems (which was not the case in SGs).

- The difference between the notions of deduction and the existence of an answer is due to the use of the law of excluded middle, which states here that "either B is blue or it is not blue".

Trying to formalize preceding observations has led us to distinguish two semantics for negative relations, with respect to two logics: *intuitionistic* logic and classical logic. In intuitionistic logic, the law of excluded middle does not hold. In fact this logic appears to capture exactly the notion of answer, as detailed in next section: Q is intuitionistically deducible from G if and only if the set of answers to G is not empty. Notice we do not claim that intuitionistic logic is the only logic suitable to our framework. Beside the fact that its deduction translates exactly the notion of answer, an interest of this logic is that its formulas are the same as in classical logic.

3.3.1 Intuitionistic negation

Intuitionistic logic is a well-established logic belonging to constructive mathematics [Fit69]. It is built upon the notion of *constructive proof*, which rejects the *reductio-ad-absurdum* reasoning. For instance, a proof of $(A \vee B)$ is given by a proof of A or a proof of B ; a proof that the falsity of $(A \vee B)$ leads to a contradiction does not yield a proof of $(A \vee B)$ since it does not determine which of A or B is true. The intuitionistic (natural) deduction rules are those of classical logic except that the absurdity rule (from $\Gamma, \neg A \vDash \bot$ deduce $\Gamma \vDash A$) does not hold. Clearly each theorem of intuitionistic logic is a theorem of classical logic but not conversely. Some characteristic examples of classical logic theorems not provable in intuitionistic logic are $(A \vee \neg A)$, $(\neg\neg A \rightarrow A)$ and $((A \rightarrow B) \rightarrow (\neg A \vee B))$.

The relationship between classical and intuitionistic logic can be expressed as follows in our framework:

Property 10 *Let us call law of excluded middle formula associated with a predicate r with arity k, the formula $\mathcal{E}(r) = \forall x_1 \ldots x_k \, (r(x_1, \ldots, x_k) \vee \neg r(x_1, \ldots, x_k))$. Given a support $\mathcal{S}$, let $\mathcal{E}_\mathcal{S}$ be the set of formulas $\mathcal{E}(r)$ for all predicates r corresponding to relation types in $\mathcal{S}$. Then: $\Phi(Q)$ is intuitionistically deducible from $\Phi(\mathcal{S})$, $\mathcal{E}_\mathcal{S}$, $\Phi(G)$ if and only if $\Phi(Q)$ is classically deducible from $\Phi(\mathcal{S})$, $\Phi(G)$.*

Let us come back to the example of figure 4. According to intuitionistic logic the formula $p(B, Blue) \vee \neg p(B, Blue)$ can be considered as true only if it can be

shown that $p(B, Blue)$ is true, or that $\neg p(B, Blue)$ is true. Since none of these two statements can be proved, Q cannot be deduced; hence the answer to Q as a yes/no question is *no*, which corresponds to the fact that there is no answer to Q as a query. Such an interpretation of a yes/no question can be seen as the *query answering problem in its decision form* which asks for the existence of an answer. In the following, we call this problem *PG intuitionistic deduction*.

Definition 18 (PG intuitionistic deduction problem) *The PG intuitionistic deduction problem (with open-world assumption) takes as input two PGs Q and G and asks whether $Q \succeq G$.*

Independently of the notion of answer, there are a lot of real world applications in which the law of excluded middle is not desirable, or not desirable for all properties or relations. It might be the case that properties or relations are neither true nor false, because they cannot be determined with certainty (for instance it might not be true that a cube is either blue or not blue, because its color might be not "really" blue ...) or they intrinsically admit "truth value gaps" (recall the example of the property "being a smoker" given in the introduction and see [Wag03] for a discussion about the need of taking into account this kind of semantics). In these cases intuitionistic deduction has to be chosen rather than classical deduction.

Definition 19 (OWA-PG query answering problem) *Let Q be a (polarized) query and G be a (polarized) KB. The query answering problem asks for image graphs from Q by all projections to G.*

The following properties and theorem establish the soundness and completeness of PG intuitionistic deduction w.r.t. deduction in intuitionistic logic, noted $\Vdash$. Since the proofs require notions of intuitionistic logic (as its formal semantics based on Kripke's models) we have put them in appendix.

Property 11 (S-Substitution soundness) *Let f and g be two formulas of FOL$\{\exists, \wedge, \neg_a\}$ on an ordered language $\mathcal{L}$. If there is a S-substitution from f to g then $\mathcal{O}(\mathcal{L}), g \Vdash f$.*

Property 12 (S-substitution completeness) *Let f and g be two formulas of FOL$\{\exists, \wedge, \neg_a\}$ on an ordered language $\mathcal{L}$. If $f(\mathcal{L}) \cup \{g\}$ is consistent and $\mathcal{O}(\mathcal{L}), g \Vdash f$ then there is a S-substitution from f to g.*

Property 13 *A polarized SG G defined on a support $\mathcal{S}$ is inconsistent iff $\Phi(\mathcal{S}) \cup \{\Phi(G)\}$ is intuitionistically inconsistent.*

Theorem 2 *Let Q and G be two polarized SGs defined on a support $\mathcal{S}$, with G is consistent. $Q \succeq nf(G)$ if and only if $\Phi(\mathcal{S}), \Phi(G) \Vdash \Phi(Q)$.*

This theorem yields the following property, which shows that intuitionistic negation captures exactly the notion of answer to a query.

Property 14 *Given two PGs Q and G, when Q is deducible from G with classical negation but not with intuitionistic negation, there is no answer to Q in G.*

3.3.2 Classical negation

The classical semantics of negation leads to a case-based reasoning: 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. Next definition specify the notion of the completion of a PG relative to a support $\mathcal{S}$.

Definition 20 (Complete PG) *A* complete PG *on a support $\mathcal{S}$ is a PG satisfying the following condition: for each relation type r of arity k in $\mathcal{S}$, for each k-tuple of concept nodes $(c_1...c_k)$, where $c_1...c_k$ are not necessarily distinct nodes, there is a relation $+s(c_1...c_k)$ with $s \leq r$ or (exclusive) there is a relation $-s(c_1...c_k)$ with $s \geq r$. A PG is complete w.r.t. a subset of relation types $T \subseteq T_R$ if the completion considers only elements of T.*

Property 15 *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 neighbor list and a label less or equal to it) or it makes the PG inconsistent.*

A complete PG is obtained from G by repeatedly adding positive and negative relations as long as adding a relation brings new information and does not yield an inconsistency. The so-called completed PG defined for closed-world assumption (cf. section 3.2) is a particular case of a complete PG obtained from G by adding negative relations only. Since a PG G is a finite graph defined over a finite support, the number of different complete PGs that can be obtained from G is finite. We can now define deduction on PGs.

Definition 21 (OWA-PG (classical) deduction problem) *The PG (classical) deduction problem with open-world assumption semantics takes as input two PGs Q and G and asks whether each complete PG G^c obtained from G is such that $Q \succeq G^c$.*

The following property expresses that PG deduction is sound and complete with respect to classical deduction in FOL$\{\exists, \wedge, \neg_a\}$.

Theorem 3 *Let Q and G be two PGs defined on a support $\mathcal{S}$. G is a consistent PG. Then Q can be (classically) deduced from $nf(G)$ if and only if $\Phi(\mathcal{S}), \Phi(G) \models \Phi(Q)$.*

Proof. $\Rightarrow$ Assume Q can be deduced from $nf(G)$ (actually the *nf* is not needed in this direction). Let M be a model of $\Phi(\mathcal{S}), \Phi(G)$. By definition of complete graphs, there is a complete graph, say G', obtained from $nf(G)$, such that M is a model of $\Phi(G')$. By hypothesis there is a projection from Q to G', thus a S-substitution from $\Phi(Q)$ to $\Phi(G')$. The composition of the assignment from $\Phi(G')$ to M and this S-substitution defines a good assignment from $\Phi(Q)$ to M. M is thus a model of $\Phi(Q)$.

$\Leftarrow$ Let G' be a complete SG obtained from $nf(G)$. Consider M the canonical model of $\Phi(G')$ (the domain is made by the terms of the formula as in the model in property 5's proof) and make it acceptable. M is a model of $\Phi(G)$ and $\Phi(\mathcal{S})$. Thus it is a model of $\Phi(Q)$ by hypothesis. There is thus a good assignment from $\Phi(Q)$ to M, which is here a mapping from $\Phi(Q)$ to $\Phi(G')$. This mapping is a S-substitution from $\Phi(Q)$ to $\Phi(G')$. There is thus a projection from Q to $nf(G') = G'$ (G' being built from $nf(G)$ it is in normal form). □

Algorithms 3 presents a naïve algorithm for OWA deduction. Algorithms for the other problems in the OWA case are recalled to ease comparison (see algorithms 4 5). An immediate observation for generating complete PGs containing G is that we do not need to consider all relations types but only those appearing in Q. The algorithm generates all complete PGs relative to this set of types and for each of them checks whether Q can be projected to it. A complete graph to which Q cannot be projected can be seen as a counter-example to the assertion that Q is deducible from G. Since fine-grained algorithms lie out of the scope of this paper we do not present a more efficient algorithm. Note however that this brute-force algorithm can be improved by benefiting from the algorithm of [WL03], which solves an equivalent problem on database queries, as pointed out in next section.

Algorithm 3: OWAClassicalDeduction

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

Algorithm 4: OWAIntuitionisticDeduction

Data: PGs Q and G
Result: true if Q is intuitionistically deducible from G, false otherwise
begin
  if *there is a projection from* Q *to* G **then return** *true*;
  else return *false*;
end

Algorithm 5: OWAQueryAnswering

Data: PGs Q and G
Result: the set of answers to Q in G with OWA
begin
  $Answers \leftarrow \emptyset$;
  forall *projection* π *from* Q *to* G **do**
    // Let $Image(Q, \pi)$ be the image of Q by π
    $Answers \leftarrow Answers \cup \{Image(Q, \pi)\}$;
  return *Answers*;
end

3.4 Equivalences with conjunctive query problems in databases

The similarity between query problems in SGs and in databases has been often pointed out. In [CMS98] the strong equivalence between SG deduction and conjunctive query containment is shown. In this section we synthesize and make precise the relationships between SG deduction and two problems in databases: query evaluation and query containment. We then extend them to PGs and conjunctive queries with negation.

Basic notions about databases Let us first recall some database definitions and results, according to a logic-based perspective (this presentation is based on [AHV95]). A *database schema* $S = (R, dom)$ includes a finite set of relation names R and a set of constants dom. A (positive) *conjunctive query* can be seen as a rule of the following form: $q = ans(u) \leftarrow r_1(u_1),\ ...\ r_n(u_n),\ n \geq 1$, where r_1 ... r_n are relation names, ans is a special relation name not belonging to R, u and u_1 ... u_n are tuples of terms (variables or constants of dom), and each variable of u occurs at least once in u_1 ... u_n. Without loss of generality we assume that a same atom does not appear twice in the right part of the rule. A *database instance* D over S maps each *k*-ary relation r_i of R to a finite subset of dom^k (denoted $D(r_i)$). Given

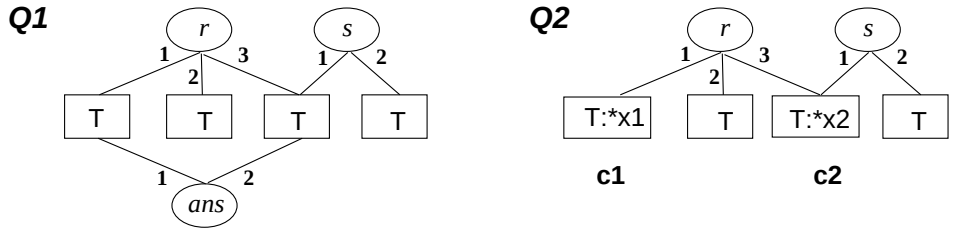
a query $q = ans(u) \leftarrow r_1(u_1),\ ...\ r_n(u_n)$ and an instance D on S, $q(D)$ denotes the set of answers to q in D; $q(D)$ is the set of tuples $\mu(u)$ where μ is a substitution of the variables in q by constants in dom such that for any j in $\{1, ..., n\}$, $\mu(u_j) \in D(r_j)$. When the arity of ans is 0, $q(D)$ is the set $\{()\}$ if there is such a substitution μ, otherwise it is $\emptyset$.

The *conjunctive query evaluation problem* is the following: given a database instance D and a conjunctive query q, does D contain an answer to q ? In practice one is interested in computing $D(q)$, the set of answers to q in D.

A query q is said to contain a query q' ($q' \subseteq q$) if, for any instance D on S, $q'(D) \subseteq q(D)$. The *conjunctive query containment problem* is the associated problem: given two queries q and q', does q contain q'? This problem can be reformulated as a query homomorphism problem, where a homomorphism is defined as follows. A *homomorphism* from $q = ans(u) \leftarrow r_1(u_1),\ ...\ r_n(u_n)$ to $q' = ans'(u') \leftarrow r'_1(u'_1),\ ...\ r'_{n'}(u'_{n'})$ is a substitution θ of the variables of q by terms of q' (variables or constants of dom) such that $\theta(u) = u'$ (thus u and u' have the same size) and for any j in $\{1, ..., n\}$, there is i in $\{1, ..., n'\}$ such that $\theta(r_j(u_j)) = r'_i(u'_i)$. The homomorphism theorem proves that, given two queries q and q', q contains q' iff there is a homomorphism from q to q' .

Equivalence with SG deduction The query evaluation problem and the query containment problem can both be polynomially transformed into the deduction problem over SGs, by way of the following mapping $d2g$ from database notions to SG notions. Notice that $d2g$ is essentially the same as the transformation $f2g$ from FOL$\{\exists, \wedge\}$ to SGs (section 2.3). To a database schema is naturally assigned a flat support: R is mapped to the set T_R of relations types and dom is mapped to the set of individual markers I. There is only one concept type $\top$. A database instance D can then naturally be seen as a totally instantiated SG $d2g(D)$ on this support (i.e. there are no generic nodes) and the right part q_r of a query q can be naturally translated into a SG $d2g(q_r)$. Each projection from $d2g(q_r)$ to $d2g(D)$ defines an answer from which the part corresponding to ans can be selected.

The *ans* part can be represented in the SG framework by means of the *lambda-SG* notion. A *lambda-SG* $L1 = (c_1\ ...\ c_k)Q$ is a SG Q with a tuple of distinguished concept nodes $(c_1\ ...\ ck)$ [Sow84]. In drawings the distinguished generic concept nodes are usually marked by naming their generic marker (see Q_2 in figure 7-right). A projection from a lambda-SG $L1 = (c_1\ ...\ c_k)Q$ to a SG G is a projection from Q to G. A projection from a lambda-SG $L1 = (c_1\ ...\ c_k)Q_1$ to a lambda-SG $L2 = (d_1\ ...\ d_k)Q_2$ is a projection from Q_1 to Q_2 which maps each c_i to d_i. A conjunctive query q can be translated into a lambda-SG $d2g(q) = (c_1\ ...\ c_k)d2g(q_r)$, where the distinguished nodes $c_1\ ...\ c_k$ correspond to the arguments of *ans*. One obtains the



$q = ans(x,\ z) \leftarrow r(x,\ y,\ z),\ s(z,\ v)$

q is transformed into a SG $Q1$ or a lambda-SG Q_2

Figure 7: Transformations from a database query to a SG

following property:

Property 16 *The conjunctive query evaluation problem can be polynomially reduced to the SG deduction problem (or equivalently to the SG query answering problem in its decision form) by the mapping $d2g$, in such a way that the set $D(q)$ is in bijection with the set of tuples $(\pi(c_1)\ ...\ \pi(c_k))$, where π is a projection from $d2g(q)$ to $d2g(D)$.*

To transform the database query containment problem into the SG deduction problem, there are two ways of mapping queries to SGs (figure 7): either ans is mapped to a special relation type and the query is mapped to a SG (as in [CMS98]), or the query is mapped to a lambda-SG as previously. It is immediately checked that in both cases one obtains the following property: given two queries, q and q' over a database schema S, every homomorphism from q to q' yields a projection from $d2g(q)$ to $d2g(q')$ over the flat support $d2g(S)$, and reciprocally. Moreover this correspondence between homomorphisms and projections is a one-to-one mapping.

Property 17 *(basically [CMS98]) The database query containment problem can be polynomially reduced to the SG deduction problem by the mapping $d2g$, in such a way that the set of homomorphisms from q to q' is in bijection with the set of projections from $d2g(q)$ to $d2g(q')$.*

Finally, let us point out that, since SGs on a support can be (polynomially) transformed into SGs on a flat support while preserving projections, SG deduction can in turn be (polynomially) transformed into conjunctive query evaluation and conjunctive query containment.

Decision problem	*CWA*	*OWA*
Query answering	NP-Complete	NP-complete
Intuitionistic Deduction	NP-Complete	NP-complete
Classical Deduction	NP-Complete	Π_P^2-complete

Table 1: Time complexity

Extension to conjunctive queries with negation and PGs. Preceding results can be directly extended to (safe) queries with atomic negation, i.e. queries with the following form:

$$q = ans(u) \leftarrow r_1(u_1),\ ...\ r_n(u_n), \neg s_1(y_1),\ ...\ \neg s_m(y_m) \quad n \geq 1,\ m \geq 0$$

The query evaluation problem makes the closed-world assumption and is thus equivalent to PG-CWA deduction. The query containment problem is equivalent to PG-OWA classical deduction. [Ull97] presents the first algorithm for solving query containment. In [WL03] a more efficient algorithm is presented. This algorithm is based on the *containment mapping* notion, which corresponds to the query homomorphism for positive queries and can be translated into the PG projection.

3.5 Complexity

Let us end this section with computational complexity issues. Given the tight relationships with databases problems, a "query" can be seen as a polarized graph or a conjunctive query with negation, and a "base" as a polarized graph (or a set of polarized graphs) or a database.

Table 1 summarizes the complexity for the three following problems, with closed-world and open-world assumptions:

- Query answering in its decisional form: Given a query Q and a base G, is the set of answers to Q in G non empty?
- Intuitionistic deduction: Given a query Q and a a base G, is Q intuitionistically deducible from G?
- Classical deduction: Given a query Q and a a base G, is Q classically deducible from G?

All NP-complete results follow from the NP-completeness of projection checking. We consider here the usual complexity definition, which takes into account the

size of the input, that is both the query and the base. In databases, for query answering problems, there is a distinction between data complexity, query complexity and combined complexity. Data complexity considers that the size of the query is insignificant with respect to the size of the database; the query is thus fixed and not part of the input of the problem. In the contrary, for query complexity, the database is fixed. Combined complexity is the usual complexity.

A naïve algorithm for checking the existence of a projection from Q to G consists in generating and testing all mappings from Q to G. The number of mappings is bounded by $size(G)^{size(Q)}$ and testing is obviously polynomial in the size of Q and G. Thus the problem is polynomial for data complexity.

If we consider combined complexity, specific polynomial cases are known for projection checking. Roughly said, the problem becomes polynomial when Q has a tree-like structure (see [GLS01] for a general synthesis and [Ker01] in the framework of SGs). This particular case is interesting in a query answering context where the source SG (a query here, but in more expressive fragments, the hypothesis of a rule, the trigger of a constraint, ...) is generally small and structurally simple, whereas the target SG (the base) can be complex. Notice that the problem remains NP-complete in its query complexity version when Q is a tree (one obtains the H-coloring problem [HNZ96]).

In comparison [Ull97] argues that classical deduction checking for PGs is Π_P^2-complete (Π_P^2 is $co\text{-}NP^{NP}$). As far as we know no efficient particular cases have been exhibited. In the worst case, known algorithms for projection checking will generate all complete graphs w.r.t. the relation types appearing in Q. This number is exponential in the size of the concept node set of the fact graph, thus the problem is not polynomial for data complexity. Consider for instance that Q contains relations of a single type t of arity k. The number of complete graphs generated is bounded by $2^{|C_G|^k}$ (since for each tuple of concept nodes in G, say $c_1,\ ...,\ c_k$, we have $t(c_1,\ ...,\ c_k)$ or $\neg t(c_1,\ ...,\ c_k)$). If T relation types are considered, then $(2^{|C_G|^k})^T$ graphs can be generated. The choice between classical and intuitionistic deduction has thus important consequences on the practical solving of deduction. The combinatorial exploding introduced by the case-based reasoning of classical deduction can be a crucial problem if the base is big and quick answers are required. A way of reducing the complexity overhead can be to apply the law of excluded middle on certain relations only, depending on their semantics, as discussed in section 4.2.

4 Related Works and Perspectives

In section 3.4 relationships with problems on queries in databases have been closely examined. This final section is devoted to other connected works and perspectives in relation with these works.

4.1 Conceptual graphs

As mentioned in the introduction, conceptual graphs are built in two stages: SGs equivalent to existential conjunctive FOL and general conceptual graphs equivalent to FOL obtained by nesting SGs into boxes representing negation. There is thus a big gap of expressiveness between the two stages. Extensions of SGs have been developed, keeping positive knowledge, as positive rules of form "if *condition* then *conclusion*" (where *condition* and *conclusion* are lambda-SGs sharing the distinguished nodes; see [BM02] for a study of their expressiveness and complexity).

Few works have considered atomic negation, that is polarized SGs. Examples showing that projection is not complete for polarized graphs have been exhibited and an algorithm based on an adaptation of the resolution method has been proposed (G. Simonet, unpublished note, 1998). In [Ker01] simpler examples are exhibited and it is shown that projection is complete on polarized SGs on a very particular case (briefly when positive and negative relations are separated into distinct connected components). As far as we know the problem of atomic negation in relationship with query problems had never been explored. The next step consists in extending this work to integrate particular cases of rules for which problems remain decidable.

4.2 Combining the three kinds of negation

In practice it may be useful to combine different kinds of negation. An interesting approach in this perspective is that of G. Wagner [Wag91] [Wag03] who, after an analysis of different kinds of negation that can be found in existing systems and languages, as well as in natural language, proposes to distinguish between several kinds of predicates. Predicates are separated into *total* predicates and *partial* predicates that may have "truth value gaps" (that is it may be the case that neither P nor $\neg P$ is true). The law of excluded middle applies to the first ones but not the second ones. Total predicates can be *open* or *closed*, according to the underlying completeness assumption, namely OWA or CWA. To each kind of predicate corresponds a kind of negation. The proposed logic for distinguishing between these three kinds of predicates is a *partial logic* with three truth values (true, false and undefined). Let us point out that computational aspects (complexity of deduction

and algorithms) are not tackled.

Although we do not consider the same logical framework, the above three kinds of predicates correspond to the three cases analyzed in the present paper. Similarly to Wagner's proposal, we could combine the three ways of processing negation. If information about a relation type is assumed to be complete, closed-world negation is used. If it is not, the question is whether the law of excluded middle applies or not. If the answer is yes, the negation for this relation type is the classical negation, otherwise it is the intuitionistic negation. Since all mechanisms defined in this paper are based on projection, combining them is not difficult.

4.3 Extended queries

Another issue, which might involve the use of several kinds of negation, is that of processing extended queries. Let us consider the simple extension where a query is not simply a SG but a SG with distinguished nodes defining the part to be considered to build an answer (similarly to the lambda-SGs used in section 3.4 to translate conjunctive queries, except that there is no ordering on distinguished concept nodes). Classically in SGs, question marks on nodes are used to define this part (see Q_2 and Q_3 in figure 8). Now an answer to a query is not the image graph itself, but the image graph restricted to the images of the nodes with question marks. This extension has no incidence for the CWA case. Nor in the OWA case if intuitionistic deduction is considered for the deduction problem. But has important consequences when deduction is based on the classical negation. Let us recall that each time classical deduction and intuitionistic deduction disagree, no answer can be exhibited. Now if an answer is built w.r.t. a subgraph of the query, it is possible to "mix" intuitionistic deduction and classical deduction.

As an illustration, figure 8 considers again the example of the cubes in figure 4, adding the information that these cubes belong to a certain scene. Three queries are considered:

- Q_1 is a query without question mark. It becomes a "yes/no" query in this new framework. Q_1 asks whether this SG can be deduced from G. Since we consider classical deduction, the answer is "yes" (cf. section 3.3);

- If question marks are put on all generic concept nodes of the query, yielding Q_2, the query asks for "all *scenes* and pairs of *cubes* such that the cubes (one blue and the other not blue) belong to the scene and the blue one is on top of the other". There is no answer to Q_2;

- In Q_3 the sole question mark is on the node representing the scene; the query asks for "all *scenes* containing a blue cube on top of a non blue cube". In this

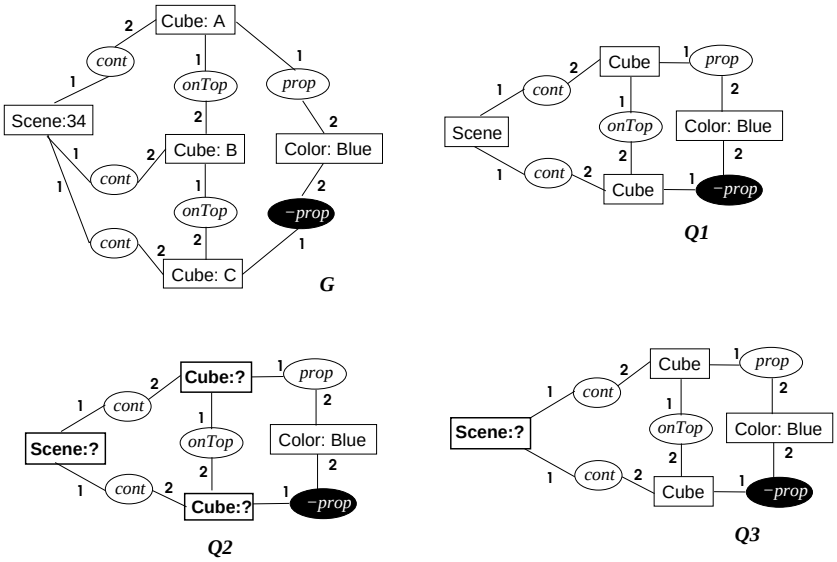


Figure 8: Queries with question marks

case, classical deduction yields an answer, which is the node [Scene:34]. Indeed, for all complete graphs G^c obtainable from G there is a projection from Q_3 to G^c mapping the node [Scene:?] to the node [Scene:34].

A natural extension to previous framework is the following:

Definition 22 (Stable projection) *Given a query Q, let us note $Q?$ the subgraph of Q composed of nodes with question marks. A PG G contains an answer to Q if there is a projection from $Q?$ to G and, for each complete SG G^c that can be obtained from G, this projection can be extended to a projection from Q to G^c . Such a projection is called a* stable *projection. It defines an image graph for $Q?$, which is an answer to Q.*

Notice that when $Q?$ is empty, G contains an answer to Q if and only if it can be classically deduced from G (since there is a projection from the empty graph to any graph).

Definition 23 (Query answering with classical deduction) *Let Q be a (polarized) query and G be a (polarized) KB. The query answering problem asks for all subgraphs G' of G such that there is a stable projection, say π, from $Q?$ to G, and $\pi(Q?) = G'$.*

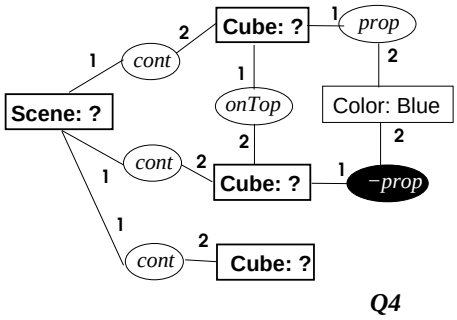


Figure 9: Is there an answer to Q_4 in the PG G of figure 8 ?

A question directly related to these definitions is whether the notion of a stable projection could be replaced by the notion of a "stable image graph". Indeed, an answer being an image graph, the existence of an answer could be based on the existence of the same image graph for a set of projections from $Q?$ to G (such that for all G^c one of these projections can be extended to a projection from $Q?$ to G^c). As an example, consider the query Q_4 in figure 9 which asks for "all *scenes* and sets of three *cubes*[3] belonging to the scene such that one of these cubes is blue, another is not blue, and the former is on top of the latter". With the stable image graph notion, the graph `[Scene:34] [Cube:A] [Cube:B] [Cube:C]` would be considered as an answer to Q_4.

The logical semantics of this extension is still to study. An important distinction is to be made with Wagner's proposal presented above. In his proposal, the semantics of negation is attached to the kind of relation. Now, it is attached to the status of the node in the query. How could both approaches be combined?

4.4 Application to the semantic web

The semantic web enhances the web by adding semantic metadata to resources for better enabling computers and people to work in cooperation [BLHL01]. The basic metadata language of the semantic web is the Resource Description Format (RDF) language and its extensions (RDFS, OWL...) [GK04].

Several works have shown that RDFS can be translated into the SG framework, RDFS-entailment being processed by way of projection. In [Bag04] the relationships between the two formalisms are formally studied. A translation from RDFS

[3]The three nodes representing the cubes are supposed to be pairwise connected by *difference* links which indicate that they represent distinct entities; as the notions of equality and inequality are not in the scope of this paper we do not represent these links.

to SGs preserving the model-theoretic semantics of RDFS [Hay04] is built and it is proved that projection is sound and complete w.r.t. RDFS-entailment. In [CDH00] another translation is proposed, from which the RDFS semantic search engine, Corese [CDKFZ04], has been developed. Corese allows to query RDFS metadata by a query language (in which basic queries correspond to the query graphs proposed in section 4.3) compatible with the SPARQL proposal of the RDF-query language [PS05]. Two essential differences between the semantic web databases and traditional databases are on one hand the presence of blank nodes (which correspond to the generic nodes of conceptual graphs and play the role of variables) and the open-world assumption concerning the representation of knowledge. The query answering problem in the SG framework becomes very close to the same problem in semantic web databases, as studied in [GHM04]. The issue discussed here is thus directly related to the issue of introducing negation into RDFS (as proposed in [AADW04] for instance).

A Appendix: Projection and intuitionistic logic

This section proves that projection on polarized SGs is sound and complete with respect to deduction in intuitionistic FOL (theorem 2).

A.1 Semantics of intuitionistic logic

The semantics of intuitionistic logic can be defined as a possible world semantics, based on Kripke models. A model is given by a set of worlds and a partial order on this set. Intuitively, each world represents a state of knowledge about a domain and knowledge increases monotonically as we go from a world to its successors. A world can be seen as a classical FOL model and the relation between worlds as the representation of the growth of knowledge. Please notice that following definitions consider logical languages without functions.

Definition 24 (Kripke model) *Let $\mathcal{L} = (S_P, S_C, S_V)$ be a FOL language composed of three pairwise disjoint sets (resp. predicate, constant, and variable symbols). A Kripke model for $\mathcal{L}$ is a 5-tuple $M = (W, \leq, D, C, F)$, where:*

- *W is a non-empty set (each element of W is called a point or a world of M);*
- *$\leq$ is a partial order on W;*
- *D is a mapping which assigns to each $m \in W$ a non-empty set $D(m)$, called the domain of m, such that: for all $m_1,\ m_2 \in S$, if $m_1 \leq m_2$ then $D(m_1) \subseteq D(m_2)$;*

- *C is a mapping which assigns to each (m, c) of $W \times S_C$ an element of $D(m)$ such that: for all $m_1,\ m_2 \in W$, if $m_1 \leq m_2$ then for each $c \in S_C$, $C(m_1, c) = C(m_2, c)$;*

- *F is a mapping which assigns to each element $m \in W$ a set of atomic formulas of form $p(d_1, ...d_n)$ with $p \in S_P$ and $d_1, ...\ d_n \in D(m)$ such that: for all $m_1,\ m_2 \in W$, if $m_1 \leq m_2$ then $F(m_1) \subseteq F(m_2)$.*

The forcing relation is defined by extending F to all formulas. A ground atomic formula is forced in m if the formula obtained from it by replacing constants with the associated elements of $D(m)$ belongs to $F(m)$. The forcing relation is inductively defined below. To simplify notations we extend formulas to formulas where some terms (variables or constants) have been replaced by elements of a domain.

Definition 25 (Forcing relation) *Let $\mathcal{L}$ be a first order language without functional symbols and let M be a Kripke model for $\mathcal{L}$. The forcing relation, denoted by $\Vdash$, between M and $\mathcal{F}(\mathcal{L}, \mathcal{M})$ the set of well-formed formulas on $\mathcal{L}$ extended to elements of domains occurring in M, is defined as follows (where m denotes a point of M, f_a, f_1 and f_2 denote wffs of $\mathcal{F}(\mathcal{L}, \mathcal{M})$, and f_a is a ground atomic formula):*

- *$m \Vdash f_a$ iff $f_a[c_1/C(m, c_1), ..., c_q/C(m, c_q)] \in F(m)$, where c_1 ... c_q are the constants appearing in f_a;*

- *$m \Vdash f_1 \wedge f_2$ iff $m \Vdash f_1$ and $m \Vdash f_2$;*

- *$m \Vdash f_1 \vee f_2$ iff $m \Vdash f_1$ or $m \Vdash f_2$;*

- *$m \Vdash \exists x f_1$ iff there exists $t \in D(m)$ such that $m \Vdash f_1[x/t]$;*

- *$m \Vdash f_1 \rightarrow f_2$ iff for every $m' \in W$ such that $m \leq m'$, if $m' \Vdash f_1$ then $m' \Vdash f_2$;*

- *$m \Vdash \neg f_1$ iff for every $m' \in W$ such that $m \leq m'$, $m' \nVdash f_1$;*

- *$m \Vdash \forall x f_1$ iff for every $m' \in W$ such that $m \leq m'$ and for every $t \in D(m')$, $m' \Vdash f_1[x/t]$.*

Definition 26 (Semantics of intuitionistic logic) *Let $\mathcal{F}$ and f denote respectively a set of formulas and a formula on a language $\mathcal{L}$. Let M denote a Kripke model for $\mathcal{L}$ and m denote a point of M.* m forces $\mathcal{F}$ *(*denoted by $m \Vdash \mathcal{F}$*) iff for all $f \in \mathcal{F}$, $m \Vdash f$.* M forces $\mathcal{F}$ *(*denoted by $M \Vdash \mathcal{F}$*) iff for every $m \in M$, $m \Vdash \mathcal{F}$.* f *is* valid *(*denoted by $\Vdash f$*) iff for all M, $M \Vdash f$.* $\mathcal{F}$ *is* inconsistent *iff there is no M with $M \Vdash F$.*

Example: to prove that $f = (p(a) \vee \neg p(a))$ is not valid we build a model which does not force f. Take for instance a model with two worlds, m_0 and m_1, $m_0 < m_1$ such that $C(m_0, a) = C(m_1, a) = e$, $p(e) \in F(m_1)$ but $p(e) \notin F(m_0)$. We have $m_0 \not\Vdash p(a)$ and, since $m_1 \Vdash p(a)$, $m_0 \not\Vdash \neg p(a)$. Thus $m_0 \not\Vdash p(a) \vee \neg p(a)$.

Definition 27 (Intuitionistic deduction) *Let $\mathcal{F}$ and f be respectively a set of formulas and a formula on a language $\mathcal{L}$. f is* deduced *from $\mathcal{F}$ (denoted by $\mathcal{F} \Vdash f$) iff for every Kripke model M for $\mathcal{L}$, if $M \Vdash \mathcal{F}$ then $M \Vdash f$.*

A.2 Soundness and completeness of projection w.r.t. intuitionistic deduction

Notations. A positive (resp. negative) literal is denoted by $p(v)$ (resp. $\neg p(v)$), where p is a predicate and v is a sequence of terms. When the considered literal can be either positive or negative it is denoted by $\sim p(v)$. For a literal l, $induced(l)$ denotes the set of literals induced by l according to the partial order on predicates i.e. $induced(p(v)) = \{q(v) | q \geq p\}$ and $induced(\neg p(v)) = \{\neg q(v) | q \leq p\}$. Given a formula f of FOL$\{\exists, \wedge, \neg_a\}$, $induced(f)$ is the union of sets of literals induced by each literal of f.

In order to focus on the essential points we first show the soundness and completeness of projection in the flat case. Thus, the set $\Phi(\mathcal{S})$ is empty and a projection maps nodes to nodes with the same type.

A.2.1 Flat case

Preliminary observations. W.l.o.g. we consider that formulas of FOL$\{\exists, \wedge, \neg_a\}$ are in prenex form. The notion of a *good assignment* defined previously for classical models is translated into Kripke models. For this fragment a point m of a Kripke model forces a formula f of a language $\mathcal{L}$ iff there is a mapping α from the terms of f to $D(m)$ which maps each constant c to $C(m, c)$, such that m forces $\alpha(f)$ the conjunction of literals obtained from f by replacing each term t by $\alpha(t)$. m forces $\alpha(f)$ iff it forces each of its (positive or negative) literals. α is called a *good assignment* of f to m.

Property 18 (S-substitution soundness in the flat case) *Let f and g be two formulas of FOL$\{\exists, \wedge, \neg_a\}$. If there is a S-substitution S from f to g then $g \Vdash f$.*

Proof. Let M be any Kripke model that forces g and let m be any point of M. Since m forces g, there exists a good assignment α from g to m. By definition of α, m forces each -positive or negative- literal of $\alpha(g)$. Now take the mapping $\alpha \circ S$

which maps the terms of f into $D(m)$. We show it is a good assignment of f into m. Let $\sim p(v)$ be any literal of f. Then $\sim p(S(v))$ is a literal of g by definition of S. Thus by definition of α, m forces $\sim p(\alpha(S(v)))$. $\alpha \circ S$ being a good assignment of f into m, m forces f. Thus $M \Vdash f$. □

Property 19 (S-substitution completeness in the flat case) *Let f and g be two formulas of FOL$\{\exists, \wedge, \neg_a\}$. If $\{g\}$ is not inconsistent and $g \Vdash f$ then there is a S-substitution from f to g.*

Proof. Since every model which forces g also forces f, we choose one from which we can build the desired S-substitution. We call it the *canonical* model of g. This model, say M, is built as follows. All points m have the same domain $D(m)$, which is the set of terms of g. M has a tree structure, with a root m_0. m_0 has two successors, $m+$ and $m-$. $F(m_0)$ contains all positive literals of g. $m-$ does not add anything to m_0, while m_+ adds all positive literals $p(v)$ such that neither $p(v)$ nor $\neg p(v)$ appears in g. M has the following property: let L_g be the set of literals that can be built using predicates and terms appearing in g; for all literal $\sim p(v)$ of L_g, M forces $\sim p(v)$ *iff* $\sim p(v)$ is a literal of g (notice this property could not be ensured with a classical model). Indeed, let $p(v)$ be a positive literal of g; $p(v)$ is forced by m_0 thus by M. Let $\neg p(v)$ be a negative literal of g; no point forces $p(v)$, thus M forces $\neg p(v)$. Finally, let $\sim p(v)$ be a literal such that neither $p(v)$ nor $\neg p(v)$ appears in g: $p(v)$ is forced by $m+$ while $\neg p(v)$ is forced by $m-$, thus none of these two literals is forced by M. Notice that we could consider a model with two points instead of three, the root being just a convenient point representing exactly what is forced by *all* points, thus by M.

Now, M forces g (with a good assignment equal to the identity) thus by hypothesis M forces f. Let α_f be a good assignment from f to any world m of M. By definition of $D(m)$, α_f is a mapping from the terms of f to the terms of g. Let $\sim p(v)$ be a literal of f. m forces $\sim p(\alpha_f(v))$, which, by construction of M, is in g. α_f is thus a S-substitution from f into g. □

Example. Let us consider the formula assigned to the SG G in figure 4 and its canonical model M. $F(m_0)$ and $F(m-)$ contain $on(A, B)$, $on(B, C)$, $cube(A)$, $cube(B)$, $cube(C)$ and $blue(A)$. $F(m+)$ contains $on(A, A)$, $on(A, C)$, $on(B, A)$, $on(B, B)$, $on(C, A)$, $on(C, B)$, $on(C, C)$, $blue(B)$. $\neg blue(C)$ is forced in M because no point forces it, while $\neg blue(B)$ is not forced because $m+$ forces $blue(B)$.

A.2.2 Ordered case

Given an ordered language $\mathcal{L}$, $\mathcal{O}(\mathcal{L})$ denotes the set of formulas associated with the partial order ($\mathcal{O}(\mathcal{L})$ is equal to $\Phi(\mathcal{S})$ where $\mathcal{S}$ is the support naturally associated with $\mathcal{L}$).

Definition 28 *A Kripke model M is acceptable w.r.t. an ordered language $\mathcal{L}$ iff the forcing relation is compatible with the partial order on predicates i.e. for all point m in M and all elements d_1 ... d_k of $D(m)$, if $p(d_1 \ldots d_k) \in F(m)$ then, for all $q \geq p$, $q(d_1 \ldots d_k) \in F(m)$.*

Notice that a Kripke model is acceptable w.r.t. $\mathcal{L}$ iff it forces $\mathcal{O}(\mathcal{L})$.

Property 11 (S-substitution soundness in the ordered case) *Let f and g be two formulas of FOL$\{\exists, \wedge, \neg_a\}$ on an ordered language $\mathcal{L}$. If there is a S-substitution from f to g then $\mathcal{O}(\mathcal{L}), g \Vdash f$.*

Proof. Let us take a model M that forces g and $\mathcal{O}(\mathcal{L})$. M is thus an acceptable model. The proof is similar to that of property 18 (flat case). Let m be any point of M and let α be a good assignment of g into m. We show that $\alpha \circ S$ which maps the terms of f to $D(m)$ is a good assignment. Let $p(v)$ be a positive literal of f. By definition of S there is a predicate $q \leq p$ such that $q(S(v))$ is in g. Thus m forces $q(\alpha(S(v)))$. Since M is acceptable, m forces $p(\alpha(S(v)))$. Now let $\neg p(v)$ be a negative literal of f. By definition of S there is a predicate $q \geq p$ such that $\neg q(S(v))$ is in g. Thus m forces $\neg q(\alpha(S(v)))$. Since M is acceptable, m forces the formula $\forall x_1 ... x_k \; p(x_1 ... x_k) \rightarrow q(x_1 ... x_k)$, therefore it forces the formula $\forall x_1 ... x_k \; \neg q(x_1 ... x_k) \rightarrow \neg p(x_1 ... x_k)$ (notice that the reciprocal deduction does not hold in intuitionistic logic). Thus m forces $\neg p(\alpha(S(v)))$. □

Property 12 (S-substitution completeness in the ordered case) *Let f and g be two formulas of FOL$\{\exists, \wedge, \neg_a\}$ on an ordered language $\mathcal{L}$. If $\mathcal{O}(\mathcal{L}) \cup \{g\}$ is not inconsistent and $\mathcal{O}(\mathcal{L}), g \Vdash f$ then there is a S-substitution from f to g.*

Proof. The proof is similar to that of property 19 (flat case). Instead of considering the literals of g we consider the literals induced by g, i.e. $induced(g)$. □

In order to translate these results into the SG framework, we first have to notice that the notion of SG inconsistency corresponds to intuitionistic logical inconsistency.

Property 13 *A polarized SG G defined on a support $\mathcal{S}$ is inconsistent iff $\Phi(\mathcal{S}) \cup \{\Phi(G)\}$ is intuitionistically inconsistent.*

Proof. Let $g = \Phi(G)$. ($\leftarrow$): if G is not inconsistent it is easy to build a classical FOL model which satisfies $\Phi(\mathcal{S}) \cup \{g\}$. It is also a Kripke model forcing $\Phi(\mathcal{S}) \cup \{g\}$. ($\rightarrow$): reciprocally, let m be any point of a Kripke model M forcing $\Phi(\mathcal{S}) \cup \{g\}$ and let α be a good assignment of g into m. If g contains two literals $p(v)$ and $\neg q(v)$ with $p \leq q$ then m forces $p(\alpha(v))$ and $\neg q(\alpha(v))$, and, since M is acceptable, m also forces $q(\alpha(v))$. Thus M is not a Kripke model. □

Theorem 2 *Let Q and G be two polarized SGs defined on a support $\mathcal{S}$, with G is not inconsistent. $Q \succeq nf(G)$ if and only if $\Phi(\mathcal{S}), \Phi(G) \Vdash \Phi(Q)$.*

Proof. If $Q \succeq nf(G)$ there is a S-substitution from $\Phi(Q)$ to $\Phi(G)$, these formulas being defined on the ordered language $\mathcal{L}$ such that $\mathcal{O}(\mathcal{L}) = \Phi(\mathcal{S})$ (property 6). By property 11, $\Phi(\mathcal{S}), \Phi(G) \Vdash \Phi(Q)$. Reciprocally, if G is not inconsistent and $\Phi(\mathcal{S}), \Phi(G) \Vdash \Phi(Q)$ there is a S-substitution from $\Phi(Q)$ to $\Phi(G)$ (properties 12 and 13) thus a projection from Q to $nf(G)$ (property 6).

□

References

[AADW04] A. Analyti, G. Antoniou, C.V. Damàsio, and G. Wagner. Negation and Negative Information in the W3C Resource Description Framework. *Annals of Mathematics, Computing and Teleinformatics*, 1(2):25–34, 2004.

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

[Bag03] J.-F. Baget. Simple Conceptual Graphs Revisited: Hypergraphs and Conjunctive Types for Efficient Projection Algorithms. In *Proc. of ICCS'03*, volume 2746 of *LNAI*. Springer, 2003.

[Bag04] J.F Baget. Homomorphismes d'hypergraphes pour la subsomption en RDF/RDFS. In *Conference LMO'04*, 2004.

[BBV97] C. Bos, B. Botella, and P. Vanheeghe. Modeling and Simulating Human Behaviors with Conceptual Graphs. In *Proc. of ICCS'97*, volume 1257 of *LNAI*, pages 275–289. Springer, 1997.

[BK03] J. Broekstra and A. Kampman. SeRQL: A Second Generation RDF Query Language. In *SWAD-Europe Workshop on Semantic Web Storage and Retrieval*, Amsterdam, 2003. Vrije Universiteit.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassile. The Semantic Web. *Scientific American*, 2001.

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

[Bra77] Ronald J. Brachman. What's in a Concept: Structural Foundations for Semantic Networks. *International Journal of Man-Machine Studies*, 9:127–152, 1977.

[CDH00] O. Corby, R. Dieng, and C. Herbert. A Conceptual Graph Model for W3C Resource Description Framework. In *springer verlag LNAI*, volume 1867, pages 468–482, 2000.

[CDKFZ04] O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker. Querying the Semantic Web with Corese Search Engine. In *Proceedings of* $3^r d$ *Prestigious Applications Intelligent Systems Conference (PAIS) Valence*, 2004.

[CG05] M. Chein and D. Genest. A Content-Search Information Retrieval based on Conceptual Graphs. *Knowledge and Information Systems (KAIS)*, 8(3), 2005.

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

[CM04] M. Chein and M.-L. Mugnier. Types and Coreference in Simple Conceptual Graphs. In *Proc. ICCS'04*, LNAI. Springer, 2004. to appear.

[CMS98] M. Chein, M.-L. Mugnier, and G. Simonet. Nested Graphs: A Graph-based Knowledge Representation Model with FOL Semantics. In *Proc. of KR'98*, pages 524–534. Morgan Kaufmann, 1998. Revised version available at http://www.lirmm.fr/~mugnier/.

[Dau02] F. Dau. *The Logic System of Concept Graphs with Negation And Its Relationship to Predicate Logic*. PhD thesis, Technische Universitat Darmstadt, 2002.

[Fit69] M. C. Fitting. *Intuitionistic Logic, Model Theory and Forcing*. North Holland, Amsterdam, 1969.

[GHM04] C. Gutierrez, C. Hurtado, and A.O. Mendelzon. Foundations of Semantic Web Databases. In *PODS*. ACM, 2004.

[GK04] J. Carroll G. Klyne. Resource Description Framework (RDF): Concepts and Abstract Syntax. Recommandation, W3C, 2004.

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

[GW95] B. C. Ghosh and V. Wuwongse. A Direct Proof Procedure for Definite Conceptual Graphs Programs. In *Proc. of ICCS'95*, volume 954 of *LNAI*, pages 158–172. Springer, 1995.

[Hay04] P. Hayes. RDF Semantics. Recommandation, W3C, 2004.

[HNZ96] P. Hell, J. Nešetřil, and X. Zhu. Duality and Polynomial Testing of Tree Homomorphisms. *Transactions of the American Mathematical Society*, 348(4):1281–1297, 1996.

[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.

[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.

[PS05] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. Technical report, W3C working draft, 2005.

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

[Ull97] J. D. Ullman. Information Integration using Logical Views. In *Proc. ICDT'97*, 1997.

[Wag91] G. Wagner. A Database Needs Two Kinds of Negation. In *Proc. 3rd symposium on Mathematical Fundamentals of Database and Knowledge Base Systems*, volume 495 of *LNCS*, pages 357–371. Springer, 1991.

[Wag03] G. Wagner. Web Rules Need Two Kinds of Negation. In *Proc. 1st International Workshop PPSW3*, volume 2901 of *LNCS*, pages –. Springer, 2003. Available at http://tmitwww.tm.tue.nl/staff/gwagner.

[WL94] M. Wermelinger and J.G. Lopes. Basic Conceptual Structures Theory. In *Proc. ICCS'98*, volume 835 of *LNAI*, pages 144–159. Springer, 1994.

[WL03] F. Wei and G. Lausen. Containment of Conjunctive Queries with Safe Negation. In *International Conference on Database Theory (ICDT)*, 2003.

[Woo75] W.A. Woods. What's in a Link: Foundations for Semantic Networks. In D.G. Bobrow and A. Collins, editors, *Representation and Understanding*, pages 35–82. Academic Press, 1975.