Simple Conceptual Graphs with Atomic Negation and Difference

Michel Leclère and Marie-Laure Mugnier

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

Abstract. This paper studies the introduction of atomic negation into simple conceptual graphs. Several semantics of negation are explored w.r.t. the deduction problem and the query answering problem. Sound and complete algorithm schemes based on projection (or coref-projection) are provided in all cases. The processing of equality/inequality is added to the framework.

1 Introduction

Simple conceptual graphs (SGs) form the keystone of conceptual graphs (CGs). They are equivalent to the positive conjunctive existential fragment of first-order-logic [BM02]. The issue tackled in this paper is their extension with a restricted form of negation, namely atomic negation (in logical terms, negation of form $\neg p$, where p is an atom). Atomic negation allows to express knowledge as "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 issue is studied both from semantic and computational viewpoints.

The framework. The reader is assumed to be familiar with basic notions about SGs. For further details about definitions and results used in this paper please see [CM04]. SGs are defined w.r.t. a vocabulary, called a *support* and denoted by $\mathcal{S}$. A support includes partially ordered sets of concept types and relations T_C and T_R. In the first sections we consider SGs without explicit coreference links ; coreference will be introduced as the same time as difference (section 4). Note however that a SG may include several concept nodes with the same individual marker, which is a case of implicit coreference. A SG is denoted by $G = (C,\ R,\ E,\ l)$, where C and R are respectively the concept and relation nodes, E is the family of edges and l is a mapping labeling nodes and edges (edges incident to a relation node are labeled from 1 to the arity of the relation). $r(c_1...c_k)$ is a short notation for a relation node with type r and argument list $(c_1...c_k)$, where $c_1...c_k$ are (not necessarily distinct) concept nodes. Φ is the classical translation from *SGs* (and the support) into FOL and $\vdash$ denotes classical FOL deduction. Given two SGs Q and G, it is known that when G is in normal form, there is a *projection* from Q to G if and only if $\Phi(\mathcal{S}), \Phi(G) \vdash \Phi(Q)$. With natural conditions on coreference, a SG always possesses a unique normal form (which we note $nf(G)$ for a SG G). However, in case the normal form does not exist or cannot be computed, *coref-projection* can be used instead of projection [CM04]. In the sequel we use projection as the basic notion (knowing that it can be replaced by coref-projection if necessary). We note $G \preceq Q$ (or $Q \succeq G$) if there is a projection from Q to G. The deduction problem is then defined as follows.

Definition 1 (SG Deduction Problem). *The SG deduction problem takes two SGs G and Q as input and asks whether $Q \succeq G$.*

Another important problem is *query answering*. This problem takes as input a knowledge base (KB) composed of SGs representing facts and a SG Q representing a query, and asks for all answers to Q in the KB. The query Q is 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, or it can be seen as the subgraph of G induced by this projection. We call it the *image graph* of Q by π.

Definition 2 (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.*

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 answer definition 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.

Definition 3 (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.*

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?").

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 proven" or "the knowledge *not* $r(x, y)$ can be proven". 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 deduction problem is no longer equivalent with the query answering problem in its decision form. Indeed, there are cases where classical deduction can be proven 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 shifts the attention to logics in which the law of excluded middle does not hold. We have chosen to consider one of these logics, intuitionistic logic [Fit69]. It is shown that intuitionistic deduction exactly captures the notion of an answer. Furthermore, we establish 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. We also give a projection-based algorithm scheme for deduction with classical interpretation of negation. Finally, the processing of inequality is added to this framework.

Related works. One may wonder why bother about atomic negation, as general CGs (obtained from SGs by adding boxes representing negation, lines representing equality, and diagrammatic derivation rules, following Peirce's existential graphs) include general negation [Sow84] [WL94] [Dau03]. The main point is that checking deduction becomes an undecidable problem. Another important point is that, notwithstanding the qualities general CGs might have for diagrammatic logical reasoning, they are not at the application end-user level (see f.i. [BBV97]). Indeed, most applications are based on SGs and extensions that keep their intuitive appeal such as nested graphs, rules and constraints (f.i. the $\mathcal{SG}$-family in [BM02]). Note that these latter extensions do not provide negation.

Few works have considered SGs with atomic negation. Simonet exhibited examples showing that projection is not complete anymore and proposed an algorithm based on an adaptation of the resolution method (unpublished note, 1998; see also [Mug00]). In [Ker01] simpler examples were exhibited and it was shown that projection remains complete in a very particular case (briefly when positive and negative relations are separated into distinct connected components). [Kli05] gave examples of problems related to the introduction of negation on relations (including equality) in the framework of protoconcept graphs (which can be translated into the conceptual graphs considered in the present paper). Moreover, as far as we know the problem of atomic negation in relationship with query problems had never been explored.

Paper organization. Section 2 introduces atomic negation into SGs, leading to *polarized* SGs. In section 3, several meanings of negation are discussed and related with the projection notion. Algorithm schemes for solving all the deduction and query problems are provided. In section 4, the results are extended to the processing of inequality. Due to space limitations, proofs are not included in this paper. The reader is referred to [ML05].

2 Polarized SGs

In this paper, we define *atomic negation* on relations only, but as explained below the results can easily be translated to concept types. Besides 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. As in [Ker01], we call *polarized* SGs (**PG**s) such SGs. A negative relation node with label $(-r)$ and neighbors $(c_1...c_k)$ expresses that "there is no relation r between $c_1...c_k$" (or if $k = 1$, "c_1 does not possess the property r"); it is logically translated by Φ into the literal $\neg r(e_1...e_k)$, where e_i is the term assigned to c_i. Let us consider the very simple example of figure 1. 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.

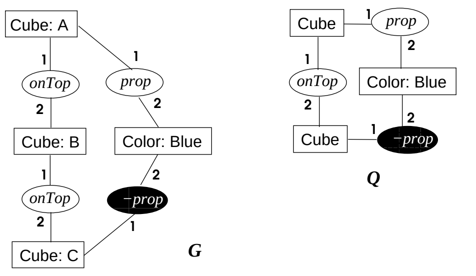

Fig. 1. Atomic negation

Projection on PGs is similar to projection on SGs with a simple extension of the order on relation node labels. The opposite type order is considered for negative labels: we set $-r_1 \leq -r_2$ if $r_2 \leq r_1$.

Definition 4 (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 $r_1 \geq r_2$.*

Since negation is introduced a PG can be inconsistent.

Definition 5 (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 1. For any PG G on a support $\mathcal{S}$, G is inconsistent iff $\Phi(\mathcal{S}) \cup \{\Phi(G)\}$ is (logically) inconsistent.

Negation on concept types. Negation in concept labels can be defined in a similar way. A concept node labeled by $-t$ (and a marker) is interpreted as *"there is an entity that is not of type t"*, and not as *"there is not 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 $(\sim t)$. A simple and uniform way of processing negation on concepts and relations thus involves applying the transformation sketched above, processing the obtained graphs with algorithms given in this paper and, if needed, applying the reverse transformation to present the results. Another solution is to adapt the algorithms, which is straightforward.

3 Different kinds of atomic negation

In this section we study three ways of understanding negation in relation with the notions of *query* and *answer*.

3.1 Closed-world assumption

A first way of understanding "not A" is "A is not present in the KB" (and more generally A cannot be obtained from the KB by 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 KB, we will not forbid a PG representing facts to contain negative relations. 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 KB but rather to its completed version.

Definition 6 (completed PG). *The* completed PG *of a PG G, denoted by completed(G), defined over a support $\mathcal{S}$, is the only 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 $-r(c_1...c_k)$.*

Definition 7 (CWA-PG deduction problem). *The PG deduction problem with closed-world assumption semantics takes two PGs Q and G as input and asks whether $Q \succeq$ completed(G).*

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

Property 2. Let Q and G be *PG*s defined on a support $\mathcal{S}$, with G being consistent. $Q \succeq$ *completed*(G) if and only if $\Phi(\mathcal{S}), \Phi(\textit{completed}(G)) \vdash \Phi(Q)$.

Definition 8 (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. We have to select the projections from Q^+ to G that do not lead to "map" a negative relation in Q to a contradictory positive relation in G.

Definition 9. *A negative relation $-r(c_1...c_k)$ in a PG Q is* satisfied *by a projection π from Q^+ to a PG G if G does not contain a positive node $+s(\pi(c_1)...\pi(c_k))$ with $s \leq r$.*

Property 3. Let Q and G be two PGs defined over a support $\mathcal{S}$, with G being consistent. There is a bijection from the set of projections from Q^+ to G satisfying all negative relations in Q to the set of projections from Q to *completed*(G).

Algorithms 1 and 2 take advantage of this property. In algorithm 2, $Ans(Q,\pi)$ is the PG obtained from Image(Q^+,π) by adding negative relations corresponding to negative relations in Q (i.e. for each $-r(c_1...c_k)$ in Q, one adds $-r(\pi(c_1)...\pi(c_k))$ to Image(Q^+,π)). 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, as the absence of a relation in an answer graph could come from the absence in the KB but also from its absence in the query. For instance, consider figure 2, which shows the only answer obtained by applying the query Q to the KB G in figure 1; 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 ← true;
1         forall negative relation $-r(c_1 ... c_k)$ in $Q$ do
              if there is $s(\pi(c_1) ... \pi(c_k))$ in $G$ with $s \leq r$ then
                  Good ← false ;   // $\pi$ is not good
                  exit this for loop ;
2         if Good then return true;
      return false;
  end
```

3.2 Open-world assumption

Let us now interpret the example in figure 1 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

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 ← true;
1       forall negative relation $-r(c_1 \dots c_k)$ in $Q$ do
            if there is $s(\pi(c_1) \dots \pi(c_k))$ in $G$ with $s \leq r$ then
                Good ← false ;   // $\pi$ is not good
                exit this for loop ;
2       if Good then $Answers \leftarrow Answers \cup \{Ans(Q, \pi)\}$;
    return $Answers$;
end
```

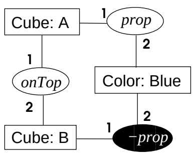

Fig. 2. Single result obtained by applying the CWA-PG query answering in figure 1

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 that there is 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 for 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 the preceding observations led us to distinguish two semantics for negative relations, namely according to *intuitionistic* logic and to classical logic. In intuitionistic logic, the law of excluded middle does not hold. In fact, this logic appears to br completely in line with the notion of answer, as detailed later: Q is intuitionistically deducible from G if and only if the set of answers to G is not empty. Note we do not claim that intuitionistic logic is the only logic suitable to our framework. Another candidate would have been 3-value logic, in which 3 truth vales are considered instead of 2: *true*, *false* and *undetermined*.

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. 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 $((A \rightarrow B) \rightarrow (\neg A \vee B))$. We denote by $\Vdash$ intuitionistic deduction (recall that $\vdash$ is classical deduction). The relationship between classical and intuitionistic logic in the logical fragment of PGs can be expressed as follows:

Property 4. For any predicate r with arity k, let $\mathcal{E}(r)$ be the formula $\forall x_1 \ldots x_k$ $(r(x_1, \ldots, x_k) \vee \neg r(x_1, \ldots, x_k))$. 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: $\Phi(\mathcal{S})$, $\mathcal{E}_{\mathcal{S}}$, $\Phi(G) \Vdash \Phi(Q)$ if and only if $\Phi(\mathcal{S})$, $\Phi(G) \vdash \Phi(Q)$.

Let us come back to the example in figure 1. According to intuitionistic logic, 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 proven, 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, that is the existence of a projection. This problem is equivalent to intuitionistic deduction checking, as shown by the next theorem.

Property 5. A polarized SG G defined on a support $\mathcal{S}$ is inconsistent iff $\Phi(\mathcal{S}) \cup \{\Phi(G)\}$ is intuitionistically inconsistent.

Theorem 1. *Let Q and G be two polarized SGs defined on a support $\mathcal{S}$, with G being 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 is completely in line with the notion of answer to a query.

Property 6. 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.

We are now able to define the intuitionistic deduction problem as well as the query answering problem in terms of projection.

Definition 10 (OWA-PG intuitionistic deduction problem). *The PG intuitionistic deduction problem takes two PGs Q and G as input and asks whether $Q \succeq G$.*

Definition 11 (OWA-PG query answering problem). *The OWA-PG query answering problem takes two PGs Q and G as input and asks for the set of image graphs of Q by all projections to G.*

Classical negation The classical semantic 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. The next definition specifies the notion of the completion of a PG relative to a support $\mathcal{S}$.

Definition 12 (Complete PG). *A* complete PG *on a support $\mathcal{S}$ is a consistent (normal) 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 7. 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 3.1) 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 13 (OWA-PG (classical) deduction problem). *The PG (classical) deduction problem with open-world assumption semantics takes two PGs Q and G as input and asks whether each complete PG G^c obtained from G is such that $Q \succeq G^c$.*

This problem is known to be Π_p^2-complete (Π_p^2 is co-NP^{NP}) whereas deduction on SGs is NP-complete. The following property expresses that the PG deduction is sound and complete with respect to the classical deduction in FOL.

Theorem 2. *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 $\Phi(\mathcal{S}), \Phi(G) \models \Phi(Q)$.*

Algorithm 3 presents a brute-force algorithm scheme for OWA deduction. Let us recall that the other OWA problems (intuitionistic deduction and query answering) are directly based on projection. An immediate observation for generating the G^c 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. Although algorithm improvments are beyond the scope of this paper, let us outline an improved way of checking deduction. Consider the space of graphs $\mathcal{G}$ leading from G to its completions (and ordered by subgraph inclusion). The question "is there a projection from Q to each $G^c \in \mathcal{G}$?" can be reformulated as "is there a set of (incomparable) SGs $\{G_1$,..., $G_k\}$ in this space, which covers $\mathcal{G}$, i.e. each $G^c \in \mathcal{G}$ has one of the G_i as subgraph, and such that there is projection from Q to each G_i?". The brute-force algorithm takes $\mathcal{G}$ as the covering set. A more efficient method consists in building a covering set by incrementally completing G, one relation node after another.

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. relation types in $Q$;
    forall $G^c \in \mathcal{G}$ do
        if there is no projection from $Q$ to $G^c$ then
            return false ;    // $G^c$ is a counter-example
    return true;
end
```

4 Equality and difference

In this section we extend previous framework to equality and inequality, also called *coreference* and *difference*. Equality is classically represented in conceptual graphs by a special feature called a coreference link. A coreference link relates two concept nodes and indicates that these nodes represent the same entity. See figure 3: coreference links are represented by dashed lines; in addition there is an implicit coreference link between two nodes with the same individual marker (here c_1 and c_5). Formally, coreference can be defined as an equivalence relation, denoted by *coref*, added to a SG, such that nodes with the same individual marker necessarily belong to the same equivalence class. As most knowlegde representation formalisms, conceptual graphs make the "unique name assumption" (UNA). Consequently, nodes with different individual markers necessarily belong to different equivalence classes. In addition, coreferent concepts must have *compatible* types (see the discussion in [CM04]). Let us point out that equality does not bring more expressiveness to SGs (at least in the context of UNA). Indeed a SG with coreference, say G, is logically equivalent to the normal SG $nf(G)$ obtained by merging all nodes belonging to the same coreference class (see figure 3). We present it for clarity reasons, since difference is naturally seen as the negation of coreference.

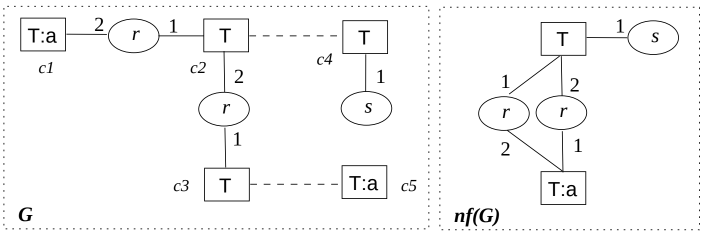

Fig. 3. Coreference: *coref*$= \{\{c_1,\ c_3,\ c_5\}, \{c_2,\ c_4\}\}$

Let us introduce inequality (or difference) as a special element of the SG syntax, called a *difference link*. A difference link between two nodes c_1 and c_2 expresses that c_1 and c_2 represent distinct entities. See Figure 4: difference links are represented by crossed lines. Due to the unique name assumption, there is an implicit difference link between nodes having distinct individual markers. Formally, difference is added to SGs as a symmetrical and antireflexive relation on concept nodes, called *dif*. In next definitions, we distinguish between the set of explicit coreference and difference links (E_{coref} and E_{dif}) and the relations ($coref$ and dif) obtained from explicit *and* implicit links.

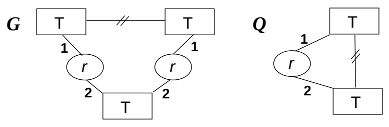

Fig. 4. PG$^{\neq}$: Q is classically deducible from G

Definition 14 ($\mathbf{PG}^{\neq}$). *A (polarized) SG with equality and inequality (notation $PG^{\neq}$) is a 6-tuple $(C,\ R,\ E,\ l,\ E_{coref},\ E_{dif})$ where:*

- *$(C, R,\ E,\ l)$ is a (polarized) SG;*
- *E_{coref} and E_{dif} are sets of specific edges between* distinct *nodes of C.*

Definition 15 ($coref$ relation). *The relation $coref$ on a $PG^{\neq}$ $G = (C,\ R,\ E,\ l,\ E_{coref},\ E_{dif})$ is the equivalence relation over C defined as the reflexive and transitive closure of the union of:*

- *the symmetrical relation induced by E_{coref} over C;*
- *implicit links due to multiple occurrences of individual markers: $\{\{c, c'\} \mid c, c' \in C,\ marker(c) \neq *$ and $marker(c) = marker(c')\}$;*

Before defining dif, we introduce a relation Dif on the *equivalence classes* of $coref$.

Definition 16 (Dif relation). *The relation Dif on a $PG^{\neq}$ $G = (C,\ R,\ E,\ l,\ E_{coref},\ E_{dif})$ is the symmetrical relation over equivalence classes of $coref$ defined as the union of:*

1. *the symmetrical relation induced by E_{dif}: $\{\{C_1, C_2\} \mid$ there are $c_1 \in C_1,\ c_2 \in C_2$ with $\{c_1, c_2\} \in E_{dif}\}$;*
2. *implicit links due to unique name assumption: $\{\{C_1, C_2\} \mid$ there are $c_1 \in C_1,\ c_2 \in C_2$ with $marker(c_1) \neq *$, $marker(c_2) \neq *$ and $marker(c_1) \neq marker(c_2)\}$;*
3. *implicit links due to incompatible types (in the sense of [CM04]): $\{\{C_1, C_2\} \mid$ there are $c_1 \in C_1,\ c_2 \in C_2$ such that $type(c_1)$ and $type(c_2)$ are incompatible $\}$;*
4. *implicit links due to contradictory relations: $\{(C_1, C_2) \mid$ there are $c_1 \in C_1,\ c_2 \in C_2$ and $r_1 = +t(d_1...d_q),\ r_2 = -s(e_1...e_q) \in R$ such that $t \leq s$ and for all $k \in \{1..q\}$, one has $\{d_1, e_k\} \in coref$ except for exactly one value of k.*

Sets 2, 3 and 4 could be deduced from set 1 and knowledge in the support but we prefer adding them explicitely in Dif. Set 4 is illustrated by figure 5: the relation nodes have opposite labels and coreferent neighborhood except for c and c'; making c and c' coreferent would lead to an inconsistent graph.

Definition 17 (dif relation). *The relation dif on a $PG^{\neq}$ $G = (C,\ R,\ E,\ l,\ E_{coref},\ E_{dif})$ is the symmetrical relation over C defined as the cartesian product of all pairs of coref classes belonging to Dif (i.e. if $\{C_1, C_2\} \in Dif$ then for all $c_1 \in C_1$ and $c_2 \in C_2$, $\{c_1, c_2\} \in dif$).*

For a $PG^{\neq}$ in normal form, one has $dif = Dif$. A PG$^{\neq}$ is consistent if it does not contain contradictory information, i.e. it is consistent *as a PG* and $coref$ and dif are not contradictory ($coref \cap dif = \emptyset$). Note that dif and Dif are then antireflexive.

The FOL formula assigned to a PG$^{\neq}$ translates coreference by assigning the same term (variable or constant) to coreferent nodes, or equivalently, by adding an atom $e_1 = e_2$ for each pair of coreferent nodes with assigned terms e_1 and e_2.

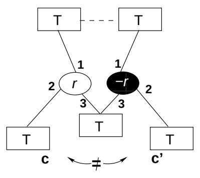

Fig. 5. Implicit *dif* link

dif is translated by $\neq$. Every consistent PG$^{\neq}$ possesses a normal form which is obtained by merging all concept nodes of the same *coref* class (note that this is a generalization of the normal form of *SG*s, where *coref* is implicitly defined, two nodes being in the same *coref* class if and only if they have the same individual marker). The obtained PG$^{\neq}$ is logically equivalent to the original one.

Let us come back to (positive) SGs and consider the processing of coreference and/or difference. A projection π from Q to G has to respect coreference: for all nodes c_1 and c_2 of Q, if $\{c_1, c_2\} \in coref_Q$ then $\{\pi(c_1), \pi(c_2)\} \in coref_G$ (note $\pi(c_1)$ and $\pi(c_2)$ can be the same node). As for positive SGs without coreference, completeness is obtained only if the target SG is in normal form. Recall that projection can be replaced by coref-projection to ensure completeness without this condition. If difference is added to SGs, projection has to respect the *dif* relation as well: for all nodes c_1 and c_2 of Q, if $\{c_1, c_2\} \in dif_Q$ then $\{\pi(c_1), \pi(c_2)\} \in dif_G$. Concerning completeness, the same discussion as for negative relations about the use of the law of excluded middle can be brought as illustrated by figure 4. The formulas assigned to G and Q are respectively $\Phi(G) = \exists x \exists y \exists z\ (r(x,\ z) \wedge r(y,\ z) \wedge \neg(x = y))$ and $\Phi(Q) = \exists x \exists y\ (r(x,\ y) \wedge \neg(x = y))$. $\Phi(Q)$ can be (classically) deduced from $\Phi(G)$, using the law of excluded middle for $x = z$ and/or $y = z$, while there is no projection from Q to G. As for PGs, projection is sound with respect to classical deduction and intuitionistic deduction, and it is complete for intuitionistic deduction only.

The extension to PG$^{\neq}$s of algorithms designed for PGs is easy. In the CWA case, nodes of the KB G not known as being coreferent are considered as being connected by a *dif* link. Thus a projection π from Q^+ to G has to satisfy: for all nodes c_1 and c_2 in Q, if $\{c_1, c_2\} \in dif_Q$ then $\{\pi(c_1), \pi(c_2)\} \notin coref_G$ (or simply $\pi(c_1) \neq \pi(c_2)$ if G is in normal form). The algorithms 1 (deduction) and 2 (query answering) are extended by insertion of a new step checking this condition between steps 1 and 2. In the OWA case, no changes are to be done for query answering (algorithm **??**) and intuitionistic deduction (algorithm **??**). The fact that projection preserves coreference and difference is sufficient. Concerning classical deduction, case-based reasoning has to be done as for negative relations. Algorithm 4 is a brute-force algorithm computing all *dif*-complete PG$^{\neq}$ (i.e. forall c, c' distinct concept nodes, either $\{c, c'\} \in dif$ or $\{c, c'\} \in coref$). Computing completions incrementally during deduction checking would of course be more efficient. Note that case 1 updates $coref$ but may also involve updating dif

(due to potential contradictory relations as illustrated in figure 5), while case 2 updates dif only.

```
Algorithm 4: AllCompleteGraphsForDif
Data: a PG$^{\neq}$ $G$
Result: the set of all (normal) dif-complete PG$^{\neq}$ obtainable from $G$
begin
    $CompleteSet \leftarrow \emptyset$;
    CompleteRec($G$);
    return $CompleteSet$;
end
```

```
Procedure CompleteRec($G$)
Data: a PG$^{\neq}$ $G$
Result: this procedure computes recursively the completion of $G$; it has access
        to all data of the main algorithm (alg. 4)
begin
    if $dif \cup coref$ is complete then
        // all pair of distinct concept nodes are in $dif$ or $coref$
        $CompleteSet \leftarrow CompleteSet \cup \{G\}$;
    else
        Choose two (distinct) nodes $c, c'$ in $G$ such that $\{c,\ c'\} \notin dif \cup coref$;
        // case 1: make them coreferent
        let $G_1$ be obtained from $G$ by adding $\{c, c'\}$ to $E_{coref}$
            CompleteRec($G_1$);
        // case 2: make them ``different''
        let $G_2$ be obtained from $G$ by adding $\{c, c'\}$ to $E_{dif}$
            CompleteRec($G_2$);
end
```

5 Perspectives

In this paper we study separately three kinds of negation. In practice it may be useful to combine them. An interesting approach in this perspective is that of G. Wagner [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. A kind of negation corresponds to each kind of predicate. The proposed logic for distinguishing between these three kinds of predicates is a *partial logic* with three truth values (true, false and undefined). 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. Similar 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 (or coref-projection), combining them is not difficult.

References

[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.

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

[CM04] M. Chein and M.-L. Mugnier. Types and Coreference in Simple Conceptual Graphs. In *Proc. ICCS'04*, volume 3127 of *LNAI*, pages 303–318. Springer, 2004.

[Dau03] Frithjof Dau. *The Logic System of Concept Graphs with Negation And Its Relationship to Predicate Logic*, volume 2892 of *LNCS*. Springer, 2003.

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

[Ker01] G. Kerdiles. *Saying it with Pictures: a Logical Landscape of Conceptual Graphs.* PhD thesis, Univ. Montpellier II / Amsterdam, Nov. 2001. http://turing.wins.uva.nl/˜kerdiles/.

[Kli05] J. Klinger. Local Negation in Concept Graphs. In *Proc. of ICCS'05*, volume 3596 of *LNAI*, pages 209–222. Springer, 2005.

[ML05] M.L. Mugnier and M. Leclère. On Querying Simple Conceptual Graphs with Negation. Research Report LIRMM 05-051, revised version to appear in Data and Knowledge Engineering, 2006, July 2005.

[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.

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

[Wag03] G. Wagner. Web Rules Need Two Kinds of Negation. In *Proc. 1st International Workshop PPSW3*, volume 2901 of *LNCS*, pages –. Springer, 2003. 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.