Conceptual Graphs and First Order Logic

GIANNI AMATI[1,2] AND IADH OUNIS[2]

[1]*Fondazione Ugo Bordoni, v. B. Castiglione, 59, 00142 Rome, Italy*
[2]*Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK*
Email: gba@fub.it and *ounis@dcs.gla.ac.uk*

We study Sowa's conceptual graphs (CGs) with both existential and universal quantifiers. We explore in detail the existential fragment. We extend and modify Sowa's original graph derivation system with new rules and prove the soundness and completeness theorem with respect to Sowa's standard interpretation of CGs into first order logic (FOL). The proof is obtained by reducing the graph derivation to a question-answering problem. The graph derivation can be equivalently obtained by querying a Definite Horn Clauses program by a conjunction of positive atoms. Moreover, the proof provides an algorithm for graph derivation in a pure proof-theoretic fashion, namely by means of a slight enhancement of the standard PROLOG interpreter. The graph derivation can be rebuilt step-by-step and constructively from the resolution-based proof. We provide a notion of CGs in normal form (the table of the conceptual graph) and show that the PROLOG interpreter also gives a projection algorithm between normal CGs. The normal forms are obtained by extending the FOL language by witnesses (new constants) and extending the graph derivation system. By applying iteratively a set of rules the reduction process terminates with the normal form of a conceptual graph. We also show that graph derivation can be reduced to a question-answering problem in propositional datalog for a subclass of simple CGs. The embedding into propositional datalog makes the complexity of the derivation polynomial.

Received August 7, 1998; revised November 4, 1999

1. INTRODUCTION

In 1984, Sowa [1] introduced a knowledge representation model that can be considered as a compromise between a formal language and a graphical language, the *conceptual graph* formalism. Due to the graphical representation of knowledge, this model allows for the construction of user interfaces.

As mentioned by Gaines [2], conceptual graphs can be considered as a basis for an operational knowledge science and technology encompassing natural language, formal language, visual language, and a range of reasoning processes that may be based on them. As will be shown in this paper, conceptual graphs consist in a convivial graphical representation of logic.

In contrast to logic-based systems where semantics and proof theory are important to understand and derive knowledge, in conceptual graph theory two alternative notions were introduced: graph derivation and projection. It turns out that the basic rule for a graph derivation is the so called *restriction rule*: a concept C can be restricted to a subconcept C', in symbols $C' \leq C$, together with a suitable substitution of the existential quantifier for individuals. The knowledge base is thus organized by a lattice of concepts, which are in turn used to define relations. Relations can be seen as the cartesian product of a certain number of concepts. This knowledge base is called the *canon* of the conceptual graphs [1].

Since the main operation for deriving graphs is given by restriction, the idea of Sowa was to obtain the graph derivation $h \trianglelefteq g$, to be read as h derives from g, by projecting all concept nodes of the starting graph g onto the nodes of the final graph h, provided that restriction is soundly preserved.

Thus, both graph derivation and projection proceed in the same direction, but opposite to that of logical derivability. This fact seems to us to be a major source of confusion and makes the nature of derivation of conceptual graphs counterintuitive. However, this paper shows that semantics and proof theory can fully take their place when using conceptual graphs. The graph derivation interpretation of CGs seems to have just a pure theoretical interest.

In particular, we will show how to use the FOL semantics for translating the graphs and FOL theorem provers for obtaining:

(i) graph derivations, and
(ii) a projection algorithm.

Notwithstanding the logical nature of CGs, most of the research work on conceptual graphs has focused on graph theory and graph algorithms [3, 4, 5, 6], which have been considered as the core of conceptual graph theory [7]. Nevertheless, there have been some attempts to provide a sound and complete graph derivation algorithm with respect to the FOL semantics and graph derivation [8, 9, 10, 11]. As for the

existential fragment, our characterization is different from the previous proposals. First, we give a proper extension of graph rules. Second, we give a new definition of a *normal* conceptual graph. In contrast to all previous syntactical definitions [12], our notion is based on the notion of graph derivation, that is a normal graph is obtained as a fixpoint of the application of a suitable set of graph derivation rules. Our normal form gives different results from the standard notion of normal form [9, 12]. Indeed, our normal form is a database, while their normal form is a graph. Third, this notion of normal form can be given just by introducing new graph derivation rules and, accordingly, by extending the FOL language and semantics with *witnesses*. Indeed, it is not possible to reduce a conceptual graph in our normal form by Sowa's set of derivation rules [1]. Finally, we directly embed the result of the completeness theorem for Definite Horn Clauses in our proof, which becomes easy to follow.

Up to now, the lack of a unique characterization of CGs in their general form has led to the belief that a CG is a formalism somehow different from the logical one, or possessing some complex logical characterization. For instance, many conceptual graph operators or extra notions, such as the maximal join [7], have been introduced without background semantics or explanation.

Our completeness results bring about other consequences. The logical deduction of the theorem prover automatically finds out which graph rules are needed and even their *right application order* in the graph derivation. Up to now there was no clear strategy on the application of graph rules and even those which make the system complete. For example, our rules of Split and Join$_\exists$ are sound with respect to FOL semantics (whatever the chosen translation of graphs into FOL) but they are no use in [8, 13] (though the authors claim completeness). We show a one-to-one mapping between graph rules and a resolution-based proof, but graph derivation rules are large in number, non-deterministic and cumbersome to use.

The second problem raised with conceptual graphs is the notion of *normalized forms*. Our proposal of the normal form of a graph is a graph in which some subset (to be specified) of the set of graph derivation rules cannot be further applied. By varying such a subset we may obtain different notions of the normal form. We will show that if we restrict the expressiveness of CGs, the logical derivation is polynomial. This result shows that if an underlying logic exists for such forms of conceptual graphs, it will then be much easier to build efficient algorithms for the conceptual graph interface. Hence, one can construct usable systems based on CGs. In other words, the purpose of our work is to provide a logic-based proof theory for conceptual graphs. As far as we know there is no pure logic-based conceptual graph derivation.

We hereby summarize the most important results of the paper.

(i) Using Sowa's original interpretation of CGs, we extend the interpretation of CGs to treat both existential and universal markers (see Section 4).

(ii) We then study the existential fragment. We introduce a technique (the witness technique) which allows us to define our notion of the *normal* form of a conceptual graph. The normal form of a graph consists of a table where existential markers (a marker denotes a quantifier) are replaced by suitable new constants (which are called *witnesses* in logic) [14]. The witnesses are substituted to all existential quantifiers in the translation of the conceptual graph which must be derived; more precisely, a witness for each occurrence of an existential quantified concept in the graph. The graph to be derived (according to the graph derivation direction) becomes a set of propositional facts of the form $\{C_i(e^i_j)\} \cup \{r_k(t_1, \ldots, t_m)\}$, where $t_1, \ldots, t_m$ are constants in $\{e^i_j\}$. The projection algorithm as well as the graph derivation is then obtained as a variant of the PROLOG interpreter (i.e. a SLD resolution algorithm driven by relation names). The query is an existential quantified conjunction of concepts and relations. The projection algorithm just uses the table as a lookup table and tries to unify variables of the query with witnesses and individual constants of the starting graph. In the case of no existential markers in the derived graph (the query), the algorithm is polynomial. The general case is obviously NP-complete. We must warn the reader that we do not assert that the witnessed graph translation is logically equivalent to its original translation. On the contrary, we assert that the implication between a witnessed graph translation and a query not containing such witnesses is equivalent to the implication between its original graph translation and the query. If we used skolemization, we would get only one direction of the above equivalence (see [15, Proposition 2.28]). Therefore, we need to plug special rules for witnesses in the graph derivation system. Also, we need extra logical axioms in the semantics in order to prove both directions of the equivalence. As a consequence, witnesses cannot be treated as ordinary constants when using graph derivation systems.

(iii) We then show that our deduction algorithm is sound and complete with respect to FOL semantics, graph derivation rules and projection (up to our normal forms).

(iv) Since we show that deduction provides a neat and clear algorithm for deriving graphs for the existential fragment, for the sake of completeness we give a semantics and a set of graph derivation rules for the CGs with universal quantifiers case. However, it is beyond the scope of our paper to provide completeness of our conceptual graph derivation rules for the CGs' extension with the universal marker (a different approach is given in [16]). It would be a long exercise to show how to rename variables under the scope of the quantifiers when deriving the formulas graphically.

(v) For the sake of clarity we present here a summary of how we prove the completeness theorem. Let A be a projection up to the normal form, B be the graph derivation, and C be the logical derivation.

We will prove $A \Rightarrow B \Rightarrow C \Rightarrow A$ to show that A, B and C are equivalent. $B \Rightarrow C$ is the soundness. $C \Rightarrow A$ is given for free by the completeness of resolution. $A \Rightarrow B$ is easily obtained. Going back and forth from the original graph to its normal form is obtained by two terminating algorithms which preserve soundness.

(vi) There is another case when the derivation algorithm is polynomial, that is when each existential quantified concept may occur at most once. So if we restrict the expressiveness of conceptual graphs to this form we get a tractable model of conceptual graphs. Therefore, we add a new rule (Split$_\exists$ rule) to the pure existential fragment with the original Sowa's graph operations and modify the semantics. We provide a new logical reading of CGs which shows that projection can be reduced to a propositional question-answering problem relatively to a set of ground facts together with a set of simple ground implications (one atom as antecedent and one as consequent). This can be seen as a restricted form of a propositional Definite Horn Program, where queries are ground conjunctions of atoms. This proves that the derivation between CGs is polynomial in time [17, 18, 19, 20].

2. CONCEPTUAL GRAPHS

We chose to change some terminology of CGs. The reason is that some notions like generic, universal, existential markers have been in conflict in the CGs literature. We decided to use the logical symbol $\forall$ to denote both the generic ($\star$ as in [1]) and the standard universal quantifier, and $\exists$ to denote the existential quantifier.[3]

Let us introduce conceptual graphs. A conceptual graph is a bipartite, finite and directed graph of *concept nodes* and *conceptual relation nodes*. In the graphs, concept nodes represent classes of individuals, and conceptual relation nodes show how the concept nodes are related [1]. A *type* is associated to each (concept or relation) node. We do not require CGs to be connected.

We may use the term 'concept' or 'concept type' for 'the type of a concept node' unless confusion arises. All concepts are related by a partial ordering relation $\leq$, a *specialization/generalization relation*. If $C_1 \leq C_2$, then we say that C_1 is a *restriction* of C_2; also $[C : a] \leq [C]$ ($[C]$ is called *existential* (i.e. $[C] = [C : \exists]$) and $[C : a]$ is an *individual concept*). $g' \trianglelefteq g$ denotes the derivation of g' from g by using the following standard operators:

- **Copy:** the copy g' of a conceptual graph g is a conceptual graph.
- **Restriction:** a concept type is replaced by a subtype, in particular an existential concept is instantiated to a 'conforming referent' (an individual in the denotation concept type).
- **Simplification:** if concept nodes are linked by two identical relation nodes, then one occurrence can be deleted.
- **Join:** Two graphs having a 'common' concept node (either existential or individual) can be joined to form one graph by sharing this common concept.

[3]The marker $\star$ is no longer considered in Sowa's new book [21].

DEFINITION 1. (Canon) *A* canon *is a* 4*-tuple* $\langle \mathcal{T}_c, \mathcal{M}, \Sigma_r, Conf \rangle$ *made up by:* (i) *a lattice* $\leq$ *on a set* $\mathcal{T}_c$ *of* concept types*;* (ii) *a set* $\mathcal{M} = \zeta \cup W \cup \{\forall, \exists\}$ *of* quantifier markers *for individuals. The marker* $\forall$ *stands for a universal individual in the domain, the marker* $\exists$ *stands for a selected individual in the domain, while the extra sets* W*, called the set of* witnesses*, and* ζ*, called the set of* individual markers*, are both used to denote individuals in a concept type domain. The operational interpretations of the set* W *will be explained in Sections* 4 *and* 5*;* (iii) *a* signature $\Sigma_r = (r, n, C_1, \ldots, C_n)$ *for each* r *with arity* n*. We denote by* $\Sigma_i(r)$ *the* i*th concept type* C_i *in the signature of* r*;* (iv) *a* conformity relation *Conf on* $\mathcal{M}$ *and* $\mathcal{T}_c$*: it tells us whether* $C(d)$ *or* $C(w)$ *can be interpreted as true. We use the unique name assumption for the markers of* ζ*; and* (v) *a lattice* $w = \exists > a > \forall$ *on the set of markers, where* $a \in \zeta$ *and* $w \in W$.

We use $w = \exists > a > \forall$ instead of $w = \exists < a < \forall$ in order to follow the graph direction instead of the logical derivation. We will use the marker lattice for substitutions of markers in the graph derivation. If w is a witness not occurring in a graph g then we may substitute a marker $\exists$ by the markers $a, \forall, w$, which are less than or equal to $\exists$ in the lattice. Similarly, all occurrences of a witness w may be simultaneously substituted by the markers $a, \forall$, but we can substitute $\exists$ for w in a node only if w occurs exclusively in that node. An individual marker can be substituted by $\forall$. Note that if $\forall$ appears in a concept we cannot perform the marker substitution anymore. We should obviously verify that the conformity relation still holds after the substitution. All details will be explained in Section 4.

DEFINITION 2. (Conceptual graph) *A* conceptual graph $g = (\mathcal{R}, \mathcal{C}, \mathcal{E}, ord, label)$ *is a bipartite (we recall that they are not necessarily connected) and finite graph with* $\mathcal{C} \neq \emptyset$*.* $\mathcal{R}$ *and* $\mathcal{C}$ *denote its relation and concept nodes.* $\mathcal{E}$ *is the set of edges, and the edges adjacent to each relation node* r *are totally ordered by the function ord. The* i*th neighbour node of* r *in* g *is denoted by* $g_i(r)$*. Every concept node in the conceptual graph has a label defined by the mapping label. A label of a concept type* $C \in \mathcal{C}$ *is a pair label* $(C) = (c, m(c))$*, with* $c \in \mathcal{T}_c$ *and* $m(c) \in \mathcal{M}$.

Concepts and relations nodes of the conceptual graph must be well formed according to the canon in the obvious way. Single concepts nodes are considered well formed. Given a canon and a set of well formed concepts and relations, one can build an infinite set of well formed graphs.

3. INTERPRETING GRAPHS IN FOL

Sowa's interpretation of a graph into first order logic [1] is given as follows:

DEFINITION 3. (Existential fragment) *Let* $g = (\mathcal{R}, \mathcal{C}, \mathcal{E}, ord, label)$ *be a graph not containing the universal quantifier*

marker $\forall$. *We associate to each concept type a unary predicate, which we denote by the same name. Similarly, we associate with each n-relation type an n-ary predicate, which we denote by the same name. Finally, all individual markers are treated as constants of FOL.*

C. For each node $[C] \in \mathcal{C}$ *the translation is:*
- $C(x)$, *with* x *a new variable if the label of* $[C]$ *is an existential quantifier;*
- $C(a)$, *if the node is* $[C : a]$ *with* a *an individual marker.*

We denote by $C(t)$ *the translation of* $[C]$ *in FOL, where* t *can be either a variable or a constant.*

R. If $C_1, \ldots, C_n$ *are all the adjacent concept nodes to a relation node* r *we associate a formula* $\tau(r)$ *as follows: with the ordering given by ord, then* $\tau(r) = C_1(t_1) \wedge \ldots \wedge C_n(t_n) \wedge r(t_1, \ldots, t_n)$. *We put* $r(t_1) =$ *true if* r *is empty* (C_1 *is an isolated node).*

G. The translation of the graph in FOL is $\Phi(g) = \exists x_1 \ldots \exists x_k \bigwedge_{r \in \mathcal{R}} \tau(r)$, *where* $x_1 \ldots x_k$ *are all the free variables in* $\bigwedge_{r \in \mathcal{R}} \tau(r)$.

EXAMPLE 1. Let h be a graph made up of a binary relation r on the same concept node $[C : \exists]$ and of an isolated node $[C : a]$. Then the translation of h is $\exists x.C(x) \wedge r(x, x) \wedge C(a)$. Let g be the graph h without the isolated node $[C : a]$. There is a logical derivation from the translation of h to that of g and we see in the next section that there is a projection from g to h.

We now consider the translation of conceptual graphs with the universal quantifier $\forall$. Let us consider $[C_1 : \forall] \longrightarrow (r) \longrightarrow [C_2 : \forall]$ as an example. We propose the translation $\forall x_1 x_2.C_1(x_1) \wedge C_2(x_2) \rightarrow r(x_1, x_2)$. If we have in a graph both quantifiers (universal and existential) then we give the following interpretation:

DEFINITION 4. (Arbitrary graphs) *Let* $g = (\mathcal{R}, \mathcal{C}, \mathcal{E}, ord, label)$ *be a graph. Let us distinguish for convenience two sets of variables* X *and* Y. *We use the convention that if there is one or more isolated nodes, then their translation is either* $\forall x C(x)$ *or* $\exists x C(x)$ *or* $C(a)$ *according to the relative marker.*

C. For each node $[C] \in \mathcal{C}$ *the translation is:*
- $C(x)$, *with* $x \in X$ *a new variable if the label of* $[C]$ *is* $\forall$;
- $C(y)$, *with* $y \in Y$ *a new variable if the label of* $[C]$ *is* $\exists$;
- $C(a)$, *if the node is* $[C : a]$ *with* a *an individual marker.*

We denote by $C(t)$ *the translation of* $[C]$ *in FOL, where* t *can be either a variable or a constant.*

R. To each relation node r *we associate a formula* $\tau(r)$ *as follows: if* $C_1, \ldots, C_n$ *are all the concept nodes adjacent to the relation node* r *with the ordering given by ord, then* $\tau(r) = \bigwedge_{t_i \in Y \cup \zeta \cup W} C_i(t_i) \wedge [\bigwedge_{t_j \in X} C_j(t_j) \rightarrow r(t_1, \ldots, t_n)]$ *provided that* X *is not empty (otherwise use true as antecedent).*

G. The translation of the graph in FOL is $\Phi(g) = \exists y_1 \ldots \exists y_k \forall x_1 \ldots \forall x_h \bigwedge_{r \in \mathcal{R}} \tau(r)$, *where* $y_1 \ldots y_k, x_1 \ldots x_h$ *are all the free variables in* $\bigwedge_{r \in \mathcal{R}} \tau(r)$.

Note that all isolated nodes always have a simple translation that is either the logical sentence $C(a)$, $\forall C(x)$ or $\exists C(x)$.

3.1. A new interpretation of the existential fragment of CGs

We introduce a modification of the interpretation of the graph derivation with respect to the logical derivation. We give an example of this new interpretation before giving the formal definition, which is somewhat difficult to read. We allow for each concept type to have at most one existential node in a graph.

EXAMPLE 2. Let $\Phi(g)$ be the formula

$$\exists x_1 \exists x_1' \exists x_2 \exists x_2'.C(x_1) \wedge C(x_1') \wedge C(x_2) \wedge C(x_2') \wedge r(x_1, x_1', x_2, x_2').$$

We impose that the interpretation of $\Phi(g)$ is equivalent to the formula

$$\exists x_1 \exists x_2.C(x_1) \wedge C(x_1) \wedge C(x_2) \wedge C(x_2) \wedge r(x_1, x_1, x_2, x_2)$$

and thus to the formula

$$\exists x_1 \exists x_2.C(x_1) \wedge C(x_2) \wedge r(x_1, x_1, x_2, x_2).$$

The interpretation is that each concept type has at most one node with the existential marker.

DEFINITION 5. (Restricted existential fragment) *For all formulas*

$$\exists x_1 \ldots \exists x_k \ldots \exists x_1^n \ldots \exists x_m^n C(x_1) \wedge \ldots \wedge C(x_k) \wedge \ldots \wedge C_n(x_1^n) \wedge \ldots \wedge C_n(x_m^n) \wedge \phi(x_1, \ldots, x_k, \ldots, x_1^n, \ldots, x_m^n)$$

which are a translation $\Phi(g)$ *of some graph* g, *the formulas*

$$\Phi'(g) = \exists x_1 \ldots \exists x_1^n C(x) \wedge \ldots \wedge C_n(x_1^n) \wedge \wedge \phi(x_1, \ldots, x_1, \ldots, x_1^n, \ldots, x_1^n).$$

Then we add the set of equivalences $\Phi(g) \leftrightarrow \Phi'(g)$.

This definition amounts to the statement that in the translation of a conceptual graph each concept may use at most one variable for a concept.

We gave above a semantics for reducing a simple conceptual graph to a conceptual graph which has the property of having its FOL translation with at most one variable for each concept type. Alternatively, we may be already satisfied with the standard FOL translation of simple CGs, but we constrain the expressiveness to CGs having as translations those formulas which use at most one variable for each concept.

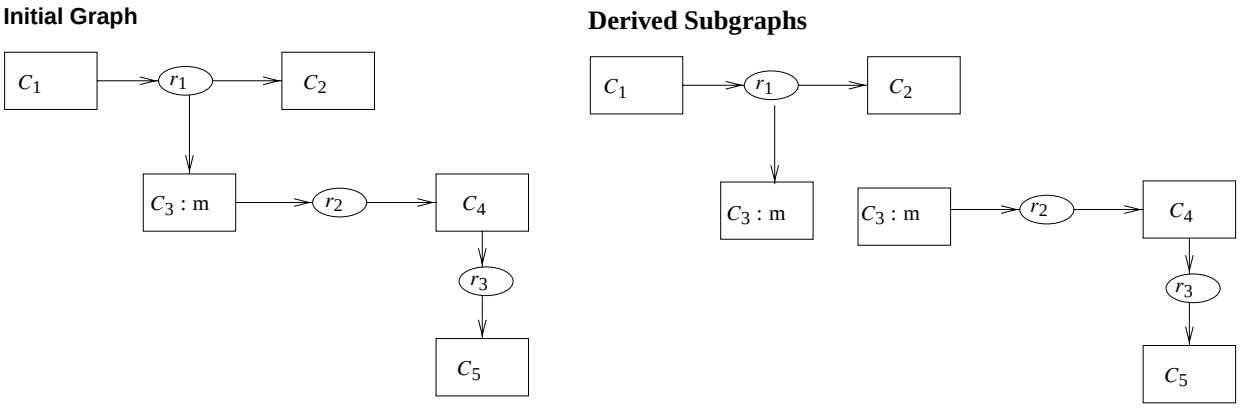

FIGURE 1. Application of the Split or Split$_\exists$ operator on the concept $[C_3]$.

We will show in Section 5.2 that Definition 5 captures a graph derivation system whose question-answering complexity is polynomial in time.

Notice that such a theory extension does not always preserve consistency. For example, if a binary relation r is irreflexive, i.e. $\forall x \neg r(x,x)$ and from the CG $\exists x_1 \exists x_2 r(x_1,x_2) \wedge C(x)$, then we get inconsistency because we deduce that there is an element such that $r(x,x)$. However, if we do use Definite Horn Clauses, then we always obtain a consistent theory. In the above example, irreflexivity cannot be expressed in Definite Horn Clauses, so we cannot end up with inconsistency.

Though we have in general an infinitary theory, this set of equivalences $\Phi(g) \leftrightarrow \Phi'(g)$ can be easily handled by witnesses. We just need to use at most one witness for each concept C, since each concept type could have at most one existential marker. Hence, this set of equivalences amounts to saying that the witness of a concept is *unique*, i.e. it is an *existential* concept node.

4. OPERATIONS ON GRAPHS

The following rules will be used in the paper. We divide the graph operation rules into three classes: rules which introduce nodes, rules which eliminate nodes and rules which restrict nodes (substitute a concept node with a subconcept node). These operators will be used in different combinations and will produce different systems of conceptual graphs.

These rules have been set up in order to have the easiest proof for the completeness theorem. For example, even the standard Join operator has been divided into the main rules: Join$_1$, Join$_\exists$. Duplication, Split are new and, obviously, witness restriction rules are new. Some readers can find similarities with other existing rules [8] sound or not with respect to FOL. However, this is not the main issue of this paper, because we actually want to show once and for all that a sound and complete graph derivation is possible, but standard theorem provers can make the derivation more efficiently and neatly.

4.1. Node introduction rules

The new introduction rules are the Split rules (see Figure 1) and Duplication.

Split: let $[C]$ be a concept node whose marker in the label is not the existential quantifier marker $\exists$ and r is a relation which has an adjacent edge in $[C]$. Introduce a new node $[C']$. Replace $[C] \rightarrow (r) \rightarrow [C_1]$ (or $[C_1] \rightarrow (r) \rightarrow [C]$) with $[C'] \rightarrow (r) \rightarrow [C_1]$ (or $[C_1] \rightarrow (r) \rightarrow [C']$), where C_1 is an arbitrary concept. Replace $[C']$ by $[C]$.

Split$_\exists$: let $[C]$ be a concept node whose marker in the label is the existential quantifier marker $\exists$ and r is a relation which has an adjacent edge in $[C]$. Split the node $[C]$ as above.

Copy: add a copy of any well formed graph to the graph.

Duplication: a node relation (r) can be duplicated together with the corresponding arrows.

4.2. Node elimination rules

The node elimination rules are:

Join$_1$. If two concept nodes have the same label with the same marker in $\zeta \cup W \cup \{\forall\}$ we can join them and obtain a single concept node.

Join$_\exists$. Let $[C : \exists]$ belong to two different graphs g and h, then these two nodes can be joined.

Simplification. As in Sowa's book.

Note that Join$_1$ and Split are inverse operations. Also Join$_\exists$, Split$_\exists$, Copy and Cut, Simplification and Duplication are inverse operations. Join$_1$ and Join$_\exists$ (Split and Split$_\exists$) do not have different conditions on their application and they could be unified in one single rule, but we need to keep them separate for the completeness theorem.

Notice that Split does not increase the number of relation nodes (r). Hence if a concept node has only one adjacent relation, then Split does not produce any modification in the graph. There is the ambiguity left in the formalism of CGs in the case a relation r is defined on two concept occurrences of the same C. In FOL this means $C(a) \wedge C(a) \wedge R(a,a)$, which is equivalent to $C(a) \wedge R(a,a)$ when treating individuals.

For universal quantifiers it is easy to see that if $x, \vec{y}$ and $x', \vec{y'}$ do not share variables, then $[\forall x(C(x) \wedge \phi(\vec{y}) \rightarrow \psi(x, \vec{y}))] \wedge [\forall x'(C(x') \wedge \phi'(\vec{y'}) \rightarrow \psi'(x', \vec{y'}))]$ is equivalent to $\forall x[C(x) \wedge \phi(\vec{y}) \rightarrow \psi(x, \vec{y})] \wedge [C(x) \wedge \phi'(\vec{y'}) \rightarrow \psi'(x, \vec{y'})]$. In such a case we may split and join indifferently. So Join_1 and Split are sound in both directions of logical consequences. In CG this means that they belong to different arcs. There is a small problem for $\forall x \forall x'(C(x) \wedge C(x') \rightarrow r(x, x'))$. This happens because x is not a free term for x' after substitution. It would be tedious now to give an ad hoc rule for such a case. We just say that if the concept node $[C : *]$ has more than one adjacent arrow with the same relation (r), then it is not a well formed graph.

4.3. Restriction on markers

The restriction rules on markers are:

Existential restriction. For existential concept nodes we may substitute for the existential marker in the label either an individual or a new witness marker (not already occurring in the graph) or the $\forall$ marker, that is by a marker m such that $\exists > m$.

Witness restriction. For concept nodes with a witness w we may substitute w uniformly in the graph by a marker m for which $w > m$. We may substitute $\exists$ for w only if w occurs only in that node. (This constraint is to prevent us first restricting the existential node to a witness node, then splitting this node, and finally having for each new split node a new existential node. This would be a sort of a maximal join.)

Individual restriction. For concept nodes with an individual a in the label we may substitute a uniformly in the graph with a marker m for which $a > m$, that is only by $\forall$.

4.4. Restriction on concept types

The restriction rules on concepts are:

Restriction for existential, witness and individual concept nodes. We may substitute a concept subtype for the concept type (i.e. provided that the conformity relation allows for it).

Restriction for universal concept nodes. We may substitute a concept supertype for the concept type (i.e. provided that the conformity relation allows for it).

4.5. Conceptual graph systems

We use the notation $h \trianglelefteq g$ if h can be obtained from g by a finite number of graph operations.

DEFINITION 6. (The family of $\mathcal{S}1$) *By $\mathcal{S}1$ we denote the system which has the following set of rules: Join, Split, $\text{Join}_\exists$, Copy, Simplification, Duplication, and all restriction rules. We denote the existential fragment by $\mathcal{S}1_\exists$.*

DEFINITION 7. (The family of $\mathcal{S}2$) *By $\mathcal{S}2$ we denote the system which has the same derivation rules of $\mathcal{S}1$ but the well formed graphs are obtained by applying the node elimination rules (that is Join, $\text{Join}_\exists$ and Simplification) until they cannot be further performed. We denote the existential fragment of $\mathcal{S}2$ by $\mathcal{S}2_\exists$. We call the conceptual graph in this form a conceptual graph in node elimination form, or shortly in NE-normal form.*

DEFINITION 8. (The system $\mathcal{S}3$) *By $\mathcal{S}3$ we denote the system which has the same derivation rules of $\mathcal{S}1$ but with $\text{Split}_\exists$ as an additional rule. We denote the existential fragment of $\mathcal{S}3$ by $\mathcal{S}3_\exists$.*

4.5.1. *Derived rules and remarks*

Note that:

(i) Split cannot be derived by the standard graph operators of $\mathcal{S}1$, as the example in Figure 2 shows.

(ii) Split and $\text{Split}_\exists$ always preserve the number of edges adjacent to a relation node (r), therefore when the initial graph is well formed we get a new well formed one.

(iii) In $\mathcal{S}3$ all graphs are equivalent to a graph in NE-normal form. In fact in the system all rules, except restrictions, have their inverse. Hence a concept in a graph can be split or joined indifferently, still obtaining an equivalent graph. Thus the graph $g = [C : \exists] \rightarrow (r) \rightarrow [C_1 : a] \leftarrow (r_1) \leftarrow [C : \exists]$ is equivalent to the graph g' obtained by joining the two node occurrences of $[C : \exists]$. Then $h = [C : b] \rightarrow (r) \rightarrow [C_1 : a] \leftarrow (r_1) \leftarrow [C : \exists]$ must be obtained by both g and g'. The semantics of such a system is given by Definition 5. The graph h instead cannot be derived from g' in the system $\mathcal{S}2$. However, the projection from g to h by using their NE-normal forms g' and h', requires one to reduce first g and h into these NE-normal forms g' and h'. In such an operation $\text{Join}_\exists$ is also applied, that is $g' \trianglelefteq g$ and $h' \trianglelefteq h$: then from a derivation between their NE-normal form $h' \trianglelefteq g'$ we may infer that $h \trianglelefteq g$ only if we may reverse $h' \trianglelefteq h$. This can be obtained in general only if $\text{Split}_\exists$ is applied. However, if we accept the rule $\text{Split}_\exists$ we need to find new semantics for the $\wedge$ operator, that is the semantics given in Definition 5.

4.6. Witnesses

Existential quantifiers can be easily eliminated in closed formulas when all existential quantifiers are prenex, that is if the formula is of the form $\exists \vec{x} \phi(\vec{x})$, where all variables in $\phi(\vec{x})$ are free. We may introduce a set of new constants w_C^i for each concept. Notice that in conceptual graphs the formula $\exists x C(x) \wedge C'(x) \wedge \phi$ cannot be expressed, that is in CGs we cannot express that two different concepts are satisfied by the same individual, unless $C = C'$.

If h is a graph and $\Phi(h)$ is its translation in FOL then we may consider the new axiom $\Phi(h) \leftrightarrow \Phi^*(h)$, where $\Phi^*(h)$ is obtained by substituting each variable in the open formula with a witness w_C^i not used before. It is a theorem of FOL (see [14]) that $\Phi(h)$ and $\Phi(h) \wedge (\Phi(h) \leftrightarrow \Phi^*(h))$ have the same set of models. Hence if a formula χ without

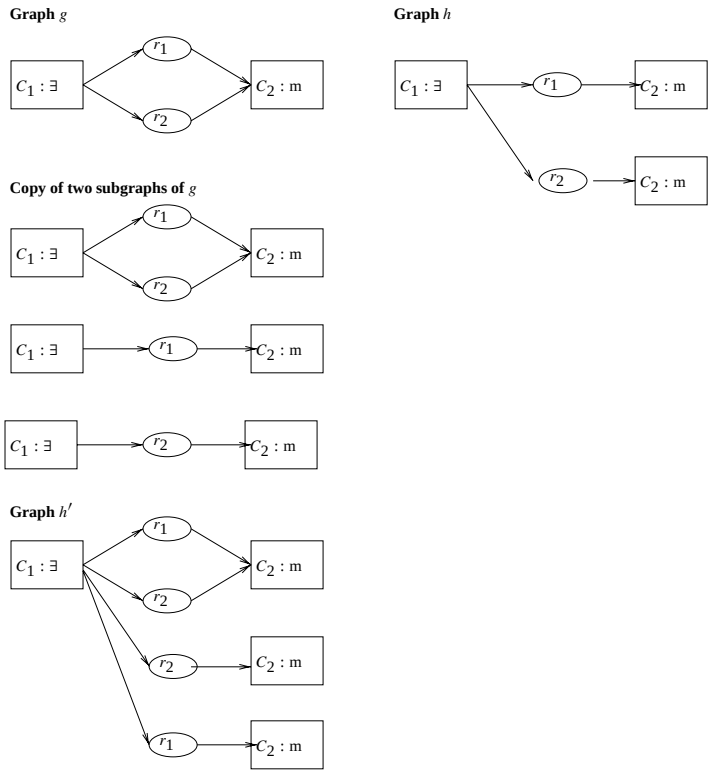

FIGURE 2. The graph h cannot be obtained from the graph g with classical graph operators. Hence, Split is necessary in order to show completeness.

witnesses is derivable from one theory it is also derivable from the other. Indeed:

(i) if $\Phi^*(h) \vdash \chi$ then $\Phi^*(h), \Phi(h) \vdash \chi$ by monotonicity (that is $\Phi^*(h), \Phi(h)$ is equivalent to the theory $\Phi(h) \wedge (\Phi(h) \leftrightarrow \Phi^*(h))$), hence $\Phi(h) \vdash \chi$;
(ii) *vice versa*, since $\Phi^*(h) \rightarrow \Phi(h)$, by the logical axiom $\phi(a) \rightarrow \exists x \phi(x)$, then $\Phi(h) \vdash \chi$ implies $\Phi^*(h) \vdash \chi$.

That is, $\Phi^*(h)$ and $\Phi(h)$ have the same expressive power when they derive formulas without witnesses. It is easy to check that all graph derivation rules with witnesses are sound with respect to this interpretation of substituting new witnesses for variables in concepts.

Thus if a graph h must be derived from a graph g, namely $h \trianglelefteq g$, then we translate h into propositional logic by means of the function $\Phi^*(h)$. g, the query, will have the translation $\Phi(g)$ as in Definition 3.

We get a set of propositional facts from $\Phi^*(h)$. Then we add the Definite Horn Clauses $C'(x) \leftarrow C(x)$ for each $C \leq C'$ in the lattice of concepts. We get a set of Definite Horn Clauses P together with a set of propositional facts. If we have to derive $h \trianglelefteq g$ we ask the query $\Phi(g)$ according to Definition 3 to the program P. The unification algorithm will provide a projection in case the query $\Phi(g)$ succeeds. Notice that the query $\Phi(g)$ does not contain any witnesses but possibly contains existential quantifiers. The enhanced PROLOG interpreter for CGs is given in Section 5. This will show that projection is sound and complete with respect to the semantics of Definition 3.

EXAMPLE 3. The graph h in Figure 2 with the marker $m = \exists$ gives the Horn program (suppose that $C_1 \leq C_2$):

$$\leftarrow C_1(w_1) \quad (1)$$
$$\leftarrow C_2(w_2) \quad (2)$$
$$\leftarrow C_2(w_3) \quad (3)$$
$$\leftarrow r_1(w_1, w_2) \quad (4)$$
$$\leftarrow r_2(w_1, w_3) \quad (5)$$
$$C(x_2) \leftarrow C(x_1). \quad (6)$$

The query g with the marker $m = \exists$ in Figure 2 becomes the conjunction:

$$?{-}r_1(x, y), C_1(x), C_2(y), r_2(x, y).$$

4.7. Tables of graphs

Let us consider $\mathcal{S}1$. We give now the notion of a normal form of a graph g: we call it the *table* of the graph.

DEFINITION 9. (Normal form) *A* table *or* normal conceptual graph *of a conceptual graph is obtained by the following* reduction algorithm.

(i) *We apply existential restriction to all existential nodes by replacing new witnesses for* $\exists$.
(ii) *We iterate Split until each node has exactly one adjacent edge.*

Note that Split does not change the graph when all nodes have exactly one adjacent edge. Also, all concept nodes

may be split once existential markers were removed by the existential restriction. Hence, this procedure terminates since at each step the total number of edges remains constant, but there is a concept whose number of adjacent edges decreases.

Each connected subgraph u of h obtained by the reduction algorithm is well formed and has only one (r) node occurring in it. Let r be of arity n and let $[C_1 : m_1], \ldots, [C_n : m_n]$ be the concept nodes occurring in a connected subgraph u of h in the order given by *ord*. We can represent in a compact way the connected subgraph u by $[C_1 : m_1], \ldots, [C_n : m_n].R(m_1, m_2, \ldots, m_n)$, where each m_i is different from the existential marker.

The table of h is the set of all connected subgraphs

$$[C_1 : m_1], \ldots, [C_n : m_n].R(m_1, m_2, \ldots, m_n)$$

of h obtained as above. We may go back to the initial graph by applying the inverse of the reduction algorithm, which reverses each single step of the derivation in the reduction algorithm.

(i) Reverse the step in which existential restriction was applied by applying witness restriction (and thus restore $\exists$).
(ii) Reverse the step with Split by using Join_1 on the split nodes.

This proves that the graph h is equivalent to its normal form h', that is $h' \trianglelefteq h$ and $h \trianglelefteq h'$.

The representation of graphs by tables allows for a direct way of deciding whether a graph h derives from another graph g, i.e. $h \trianglelefteq g$. We reduce h to its normal form h' and then apply an algorithm for defining a mapping from g in h'. If we succeed then $h \trianglelefteq g$. Note that g does not contain witnesses.

The projection mapping can be extended to consider universal quantifier markers. In this definition we may use restrictions on both concepts and relations.

We say that for two nodes $[C_1]$ and $[C_2]$ we have:

(i) $label(C_1) \leq label(C_2)$ iff
  (a) $c_1 \leq c_2$ and $m_1 \leq m_2$ and $m_2 \neq \forall$ according to the concept type lattice $\mathcal{T}_c$ and the marker lattice respectively;
  (b) $m_1 \leq m_2 = \forall$ and $c_2 \leq c_1$ according to the concept type lattice $\mathcal{T}_c$ (notice that the restriction on the universal reverses the ordering in the lattice);
(ii) $[C_1]$ is *a restriction* of $[C_2]$ iff $label(C_1) \leq label(C_2)$.

DEFINITION 10. (Projection) *Let g_1 to g_2 be two graphs without witnesses.*[4] *A mapping π from g_2 to g_1 is called a* projection *if π satisfies the following properties.*

(i) *For each concept node $[C]$ in g_2, $\pi([C])$ is a restriction of $[C]$ or is equal to $[C]$ in g_1.*
(ii) *For each conceptual relation r in g_2, $\pi(r) = r$.*
(iii) *If the ith arc of r is linked to a concept node $[C]$ in g_2, the ith arc of $\pi(r)$ must be linked to $\pi([C])$ in g_1.*

Soundness is tedious but easy, therefore we skip the proof.

THEOREM 1. (Soundness) *Let A be the set of axioms made up of all clauses $\forall x C_1(x) \rightarrow C_2(x)$, with $C_1 \leq C_2$ in the concept lattice of the canon. Let $h \trianglelefteq g$ then $A, \Phi(h) \vdash \Phi(g)$.*

As for completeness we give a circular proof in the next section.

THEOREM 2. (Completeness) *Let g and h be two conceptual graphs.*

A. A projection exists from the table of g to that of h.
B. The graph h can be derived in $S1_\exists$ from g.
C. $\Phi(g)$ derives from $\Phi(h)$ up to the canon translation as in the soundness theorem.

We have to show $A \Rightarrow B \Rightarrow C \Rightarrow A$, in order to prove that A, B and C are equivalent. $B \Rightarrow C$ is Theorem 1, hence we must show $A \Rightarrow B$ and $C \Rightarrow A$.

Below we give a projection algorithm, and then use it to prove the graph derivation. By soundness we prove the logical derivation, and finally we show that we again derive a projection by the other direction of completeness obtained by the PROLOG interpreter.

5. A PROJECTION ALGORITHM FOR $\mathcal{S}1_\exists$

Let us consider g and h which do not have witnesses. Let h' be the table of h. We derive h' by the reduction algorithm. We obtain the table $h' = \{e^i\}_{i \in I} = \{[C_1^i : m_1^i], \ldots, [C_{k(i)}^i : m_{k(i)}^i].r^i(m_1^i, m_2^i, \ldots, m_{k(i)}^i)\}_{i \in I}$.

First we define a projection algorithm from g to h'. Then we exploit the inverse reduction algorithm to prove the completeness theorem. Finally we show that there is a projection from g to h if and only if $h \trianglelefteq g$.

In the following algorithm we treat g as in the FOL translation of Definition 3. Variables of g will be unified to constants and witnesses of h. This will provide the projection theorem. Notice that the witnesses of h are treated as constants, hence we allow substitutions only for variables belonging to g. Let $\{e^j\}_{j \in J}$ be the set of all subformulas of $\Phi(g)$ made by a relation and all its connected nodes or by isolated nodes. Let us denote the constant and the variables of e^j by m_s^j and the set of all variables in $\Phi(g)$ by $\vec{z}$, that is

$$\Phi(g) = \exists \vec{z} \bigwedge_{j \in J} C_1^j(m_1^j) \wedge \ldots \wedge C_{k(j)}^j(m_{k(j)}^j) \wedge r^j(m_1^j, m_2^i, \ldots, m_{k(j)}^j).$$

The projection algorithm is given in Table 1.

This procedure terminates for all finite graphs. In the stack there is the projection mapping of all nodes of g to those of

[4] We may define projection for graphs containing witnesses. However, witnesses are not introduced at the knowledge representation level but by the graph operators: graphs containing witnesses have to be considered as intermediate forms of final graphs.

TABLE 1. The projection algorithm of g in h.

(i) Stack =: empty.
(ii) Choose a new element e^j in g **if** any **else** stop.

(iii) Consider the set L of elements e^i in h such that $r^i = r^j$.
(iv) Consider an element in L not yet examined.
(Then there is a one-to-one mapping between the concepts nodes of e^i and e^j in the order given by *ord*.)
If there are no such elements left **then** BACKTRACK to step (ii).
else for such an element e_i

FOR all s ($1 \leq s \leq k(i)$)
E = Stack + new equation $m_s^j = m_s^i$
If $[C_s^i : m_s^i]$ is a restriction node of $[C_s^j : m_s^j]$ with respect to E by substituting m_s^i for the variables m_s^j of g (and thus no multiple association of a variable m_s^j to different elements of h may occur in E)
then continue the FOR
else EXIT the FOR and Goto step (iv).
end FOR
Stack:= E

(v) Goto step (ii).

h. If it is not complete then it is a failure, and this is due to the fact that the backtracking mechanism finished without associating all elements e_j of g to one of h.

We need the following theorem:

THEOREM 3. (Soundness of projection w.r.t. graph derivation) *Let g in h be two arbitrary conceptual graphs and h' the table of h. If a projection exists from g to h' then $h \trianglelefteq g$.*

Proof. We have applied Split and existential restriction in the reduction algorithm, only the restrictions rules in the projection algorithm, and Join$_\exists$ in the case that two witnesses of h are associated with the same variable of g, Join$_1$ and witness restriction in the inverse reduction algorithm.

Finally we copy the subgraph $h - g$ up to Restriction, we apply Join$_\exists$ on all possible existential nodes and then Join$_1$ (if any). This ends the proof and gives the implication $A \Rightarrow B$ of Theorem 2.

In order to prove the last implication $C \Rightarrow A$ of Theorem 2 we just need to prove that from $\Phi(h) \vdash \Phi(g)$ we can define a projection of a graph g in h'.

This is given by observing that our projection algorithm is an adaptation of the PROLOG interpreter. They differ in the selection function for the atoms in the query: it is not the leftmost but a different fair rule [22].

THEOREM 4. (The Embedding of CGs into PROLOG) *The projection algorithm is the PROLOG interpreter (up to a fair goal selection rule).*

Proof. Let $\Phi(h)$ and $\Phi(g)$ be the FOL translation of the graphs h and g. Consider the theory with witnesses $\Phi^*(h)$ of $\Phi(h)$. $\Phi^*(h)$ is a set of facts equivalent to $\Phi(h)$. Let $\Phi(g)$ be the query. Then each step used in the projection algorithm is a linear resolution of the query with either a fact (a unit resolution in such a case) or the head of a clause of the concept lattice. In both cases the unification algorithm is applied, that is when a concept node in the query is projected onto the graph g. The selection rule of the atomic literal in the query is fair. In fact, the number of relations and concepts strictly decreases when the element e^i of the table is processed, hence each atomic goal in the query must be processed in a finite number of steps.

The selection rule is fair, hence the projection algorithm is also complete with respect to FOL. This completes the proof of completeness. □

We need to make a small digression about this point. The term *fair* refers to the goal selection rule and not to the search in logic programming. As it is well known, resolution is a complete rule for FOL formulas in clausal form [23].

The simplest proof procedure based on resolution is, in Robinson's paper, a direct implementation of the proof of completeness. However, this procedure is quite inefficient, and Robinson concluded discussing several principles (called search principles). These must be used to design efficient proof procedures employing resolution as the basic logical process.

We may find several search strategies, many of them still complete (see for example [24]).

One of these rules is called *input resolution*, which adopts one of the two clauses used in the resolution rule is provided by a static set of clauses (program). A second search procedure is the *linear resolution* which always uses the last clause obtained by resolution instead. PROLOG strategy (the SLD resolution) is the combination of the input and linear resolution is complete for a fragment of FOL, the Definite Horn Clauses.

Simple CG is a fragment of the Definite Horn Clauses fragment of FOL, hence the PROLOG strategy is complete

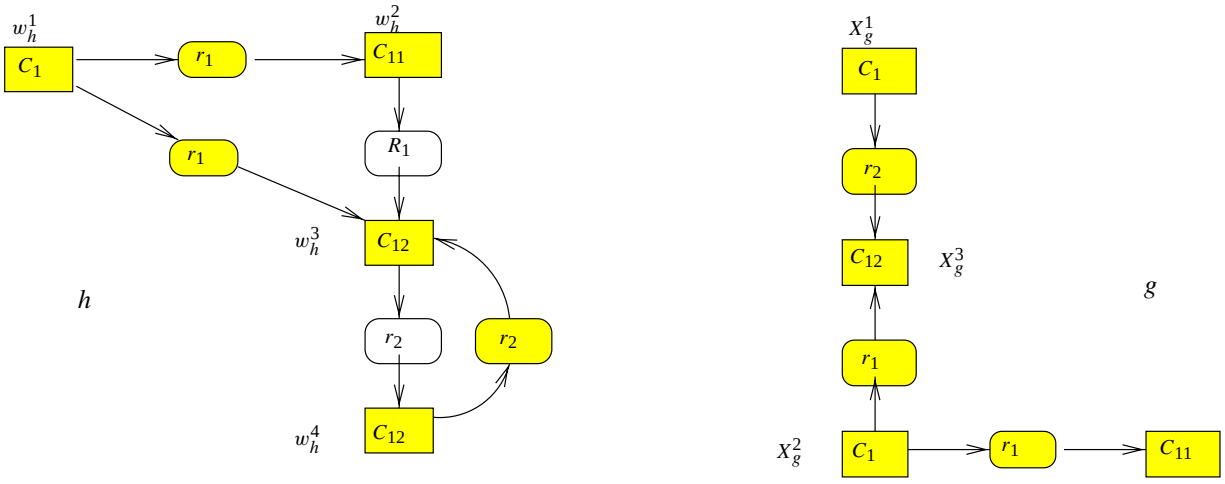

FIGURE 3. A sample projection case.

for the translation of simple CGs into FOL. We now come to the implementation problem: the way the linear and input strategy is actually implemented in PROLOG for Definite Horn Clauses. There are three separate issues: (i) the unification algorithm, (ii) the choice of the positive literal to be chosen in a query for the resolution rule, and (iii) the search on the space of solutions. The second issue (ii) is not a problem in CGs as we may use the so-called fair rule, that is, any atom in the query must be processed in a finite number of steps. For example, we can always use the topmost atom in the query seen as a queue of atoms. The first issue (i) may be time consuming with the occur check and usually this control mechanism is not implemented in most systems. But the occur check is an implementation issue and also it is needed once the Herbrand universe is infinite, namely when at least one functional symbol is in the language. We do not run into this problem since simple CGs do not have functional symbols. Finally the problem of the search. This is linked to the second issue. If there are no functional symbols and the initial program is finite then the Herbrand universe is finite, and thus any question-answering problem is decidable, independently from a breadth- or depth-first search for example.

Therefore, in what sense is our system a variant of the PROLOG interpreter? Instead of using a query as a queue and the topmost element of the queue for resolution, we split atoms into separate sets: relations (binary) and concepts (unary). Then we mainly use the set of relations as a queue and look up corresponding suitable concepts. We first pop from the relation queue, we then look for concepts and this is a fair rule.

5.1. An illustration of how the algorithm works

Consider the graphs g and h of Figure 3. We assume that all the concepts in this figure are existentially quantified, i.e. $\forall[C], [C] = [C : \exists]$. By applying the reduction algorithm to the graph h, we introduce the witnesses w_h^i as mentioned in Figure 3. We also apply the same reduction algorithm to the graph g. The variables of g are denoted by X_g^j. On the other hand, for the graph h, we obtain the following table:

$[C_1 : w_h^1], [C_{11} : w_h^2].r_1(w_h^1, w_h^2)$

$[C_{11} : w_h^2], [C_{12} : w_h^3].r_1(w_h^2, w_h^3)$

$[C_{12} : w_h^3], [C_{12} : w_h^4].r_2(w_h^3, w_h^4)$

$[C_1 : w_h^1], [C_{12} : w_h^3].r_1(w_h^1, w_h^3)$

$[C_{12} : w_h^4], [C_{12} : w_h^3].r_2(w_h^4, w_h^3)$

For convenience, we denote the graph g as follows:

$[C_1 : X_g^1], [C_{12} : X_g^2].r_2(X_g^1, X_g^2)$

$[C_1 : X_g^2], [C_{12} : X_g^3].r_1(X_q^2, X_g^3)$

$[C_1 : X_g^3], [C_{11} : X_g^4].r_2(X_g^3, X_g^4)$

To state if there is a projection of g in h, we apply our projection algorithm on the previous graphs. The following simplified trace is obtained:

Begin

1. $X_g^1 = w_h^3$, $X_g^2 = w_h^4$, remove by backtracking
2. $X_g^3 = w_h^2$, $X_g^2 = w_h^3$, remove by backtracking
3. $X_g^3 = w_h^1$, $X_g^2 = w_h^3$, remove by backtracking
4. $X_g^1 = w_h^4$, $X_g^2 = w_h^3$,
5. $X_g^3 = w_h^2$, $X_g^2 = w_h^3$, remove by backtracking
6. $X_g^3 = w_h^1$, $X_g^4 = w_h^2$, remove by backtracking
7. $X_g^3 = w_h^1$, $X_g^2 = w_h^3$,
8. $X_g^3 = w_h^1$, $X_g^4 = w_h^2$

End

During the execution of the algorithm, we always choose the first occurrence e^i in the table of h such that $r^i = r^j$. By 'remove by backtracking' we mean that according to our

algorithm, the corresponding substitutions (the equations) are removed from the stack, and that another e^i in h must be chosen.

In our example there is a projection of g in h. The one-to-one mapping from the g variables to h constants is given by the following correspondences:

4. $X_g^1 = w_h^4$, $X_g^2 = w_h^3$,

7. $X_g^3 = w_h^1$, $X_g^2 = w_h^3$,

8. $X_g^3 = w_h^1$, $X_g^4 = w_h^2$

Going back to conceptual graph considerations, we note that the previous equalities correspond to a mapping π from the nodes of g associated to the variables X_g^j to those of h corresponding to the witnesses w_h^i (see Definition 10).

5.2. The existential fragment $\mathcal{S}3_\exists$ with Split$_\exists$

Let us consider the $\exists$ fragment $\mathcal{S}3_\exists$ of Section 3. We use the semantics given in Definition 5.

Even though the set of axioms in Definition 5 can be expressed by infinitely many axioms, it turns out that in this fragment the question-answering problem is polynomial. The hypothesis is to keep both graphs g and h variables. In the previous section, the projection problem was NP-complete (the algorithm shows that it is in NP; the hard part can be shown by considering the three colours problem in a graph) even when the graphs h and g were kept fixed ('constant' according to the terminology in [17]). Suppose for the moment that we have just a single relation and a single concept for defining both graphs, and that we want to show that $h \preceq g$ and consider the full projection algorithm. Let n_e^c and m_e^c denote the number of existential nodes associated to the single concept C in g and h respectively and n_c and m_c the number of constant nodes occurring in C in g and h respectively. Then n_e^c grows arbitrarily and thus the possible candidates for projections are $(m_e^c + m_c)^{n_e^c}$ plus $n_c \times m_c$ identity tests. To find a projection we need for each node mapping a number $n_r \times m_r$ (they are the number of arcs in the graphs) of confrontations. This growth instead is bounded in the fragment with Split$_\exists$ because $n_e^c = m_e^c = 1$. Moreover, the restriction does not increase the number of nodes. It comes out that verifying whether $c \leq c'$ for two arbitrary concepts c and c' is polynomial in time (no matter if there are cycles or not [25, 26]).[5] If we have different relations the same considerations apply to each subgraph. The set of arches related to the same relation r (it is not important if it is connected or not). Since $m_e^c + 1 \leq Ind + 1$, where *Ind* is the cardinality of the set of individuals in the conformity relation (which must be finite), then $(Ind + 1)^k$ is the complexity of the algorithm, where k is the number of concepts in the query. As for $\mathcal{S}3_\exists$ we can see that the projection of $\mathcal{S}2_\exists$ is not more suitable (see the example in the remark of Section 4.5; indeed the node $[C : \exists]$ can be split and then projected). We exhibit here a set of propositional Definite Horn Clauses which represent the graph h and the ontology on the graph. We reduce the question-answering problem to the derivation of a conjunction of propositional literals corresponding to the graph g (propositional *datalog*). Hence the complexity is polynomial [17, 18, 19, 20].

Let $C_{k+1}(a_{k+1}) \wedge \ldots \wedge C_n(a_n) \wedge \exists x_1 \ldots \exists x_k C_1(x_1) \wedge \ldots \wedge C_k(x_k) \wedge r(x_1, x_2, \ldots, x_k, a_{k+1}, \ldots, a_n)$ be the translation of a single well formed part of a conceptual graph (up to a permutation of the terms), where r is of arity n. We suppose that the translation conforms to the canon. Let us introduce for each concept C_i a *unique* witness p_i.

- We substitute the above translation with a conjunction of literals, that is $C_{k+1}(a_{k+1}) \wedge \ldots \wedge C_n(a_n) \wedge C(p_1) \wedge \ldots \wedge C_k(p_k) \wedge r(p_1, p_2, \ldots, p_k, a_{k+1}, \ldots, a_n)$.
- We add to the translation of g the set of propositional formulas $C_i(a) \rightarrow C(p_i)$ for all concepts C and constants a in the conformity relation w.r.t. C. This encodes the valid formula $C(a) \rightarrow \exists x C(x)$.
- For all rules $\forall x C_1(x) \rightarrow C_2(x)$ in the canon we add $C_1(p_1) \rightarrow C_2(p_2)$.
- We define a relation $t \leq t'$ on the set of witnesses and constants as follows: $p_i \leq p_j$ iff $C_i(p_i) \rightarrow C_j(p_j)$ and $a \leq p$ if $C(a) \rightarrow C(p)$. Let r be defined in the canon with respect to $C_1, \ldots, C_n$ and let it occur in the graph h. Then we add $r(t_1, \ldots, t_n) \rightarrow r(t'_1, \ldots, t'_n)$, for all $t_i \leq t'_i \leq p_i$, $i = 1, \ldots, n$.

Obviously, the program grows polynomially. Notice the query corresponding to g is a conjunction $C'_{s+1}(a'_{s+1}) \wedge \ldots \wedge C'_m(a'_m) \wedge C'_1(p'_1) \wedge \ldots \wedge C'_s(p'_s) \wedge r(p'_1, p'_2, \ldots, p'_s, a'_{s+1}, \ldots, a'_m)$. This is a reduction of the original problem to this propositional form. This ends the proof. □

6. CONCLUSIONS

We have proved that the simple CGs are a fragment of the Definite Horn Clauses. We also showed that the standard graph derivation rules were incomplete and we added the missing rules. The enhanced graph derivation system is shown to be sound and complete. We have given a new notion of a table for CGs, for which projection is sound and complete with respect to FOL semantics. We proved that the projection algorithm is indeed an optimization of the standard resolution proof provided by the PROLOG procedural interpretation.

The proofs are all constructive and effective. Our completeness proof shows how to pass step-by-step from the graphical derivation, to projection to the resolution proof and the reverse.

A FOL interpretation of conceptual graphs with both a universal and existential quantifier was given.

We studied a polynomial fragment of CGs. This system has been implemented in an image retrieval system [27].

ACKNOWLEDGEMENTS

The first version of this paper was done while Iadh Ounis was affiliated to the CLIPS-IMAG Laboratory of

[5] Actually the algorithm is linear in time by expanding polynomially the clauses defining the lattice.

the University of Grenoble, France. Gianni Amati's work was partially supported by a Research Fellowship from the Department of Computing Science of Glasgow, and is carried out in the framework of the agreement between the Italian PT Administration and the FUB. We would like to thank Georg Gottlob for important suggestions on the complexity theory issues of this paper and Maurizio Lenzerini for a profitable discussion. Particular thanks to Yves Chiaramella who gave a significant contribution to this research and funded this work at CLIPS-IMAG of Grenoble, where most of this work was done. Also particular thanks to Marie-France Bruandet for her interest and her comments on the first version of this paper and Marcos Theophylactou for having carefully read the last version of the paper.

REFERENCES

[1] Sowa, J. F. (1984) *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley Publishing Company, Reading, MA.

[2] Gaines, B. R. (1993) Representation, discourse, logic and truth: situating knowledge technology. In Mineau, G., Moulin, W. and Sowa, J. F. (eds), *Proc. 1st Int. Conf. on Conceptual Structures, ICCS'93 (Lecture Notes in Artificial Intelligence, vol. 699)*, Quebec, Canada, pp. 36–63. Springer-Verlag.

[3] Mineau, G. W., Moulin, B. and Sowa, J. F. (eds) (1993) *Proc. 1st Int. Conf. on Conceptual Structures, ICCS'93 (Lecture Notes in Artificial Intelligence, vol. 699)*, Quebec, Canada. Springer-Verlag.

[4] Tepfenhart, W. M., Dick, J. P. and Sowa, J. F. (eds) (1994) *Conceptual Structures: Current Practices. Proc. 2nd Int. Conf. on Conceptual Structures, ICCS'94 (Lecture Notes in Artificial Intelligence, vol. 835)*, College Park, USA. Springer-Verlag.

[5] Ellis, G., Levinson, R., Rich, W. and Sowa, J. F. (eds) (1995) *3rd Int. Conf. on Conceptual Structures, ICCS'95 (Lecture Notes in Artificial Intelligence, vol. 954)*, Santa Cruz, CA. Springer-Verlag.

[6] Eklund, P. W., Ellis, G. and Mann, G. (eds) (1996) *4th Int. Conf. on Conceptual Structures, ICCS'96 (Lecture Notes in Artificial Intelligence, vol. 1115)*, Sydney, Australia. Springer-Verlag.

[7] Chein, M. and Mugnier, M. L. (1995) *Conceptual Graphs are also Graphs*. Research report 95-004, LIRMM.

[8] Chein, M. and Mugnier, M. L. (1992) Conceptual graphs: fundamental notions. *Revue d'Intelligence Artificielle*, **6**, 365–406.

[9] Ghosh, B. C. and Wuwongse, V. (1995) A direct proof procedure for definite conceptual graphs programs. In Ellis *et al.* [5], pp. 158–172.

[10] Wermelinger, M. (1995) Conceptual graphs and first-order logic. In Ellis *et al*. [5], pp. 323–337.

[11] Prediger, S. (1998) Simple concept graphs: a logic approach. In Mugnier, M. L. and Chein, M. (eds), *Proc. 6th Int. Conf. on Conceptual Structures (Lecture Notes in Artificial Intelligence)*, Montpellier, France, pp. 225–239. Springer-Verlag.

[12] Salvat, E. and Mugnier, M. L. (1996) Sound and complete forward and backward chainings of graph rules. In Eklund, P. W., Ellis, G. and Mann, G. (eds), *Proc. 4th Int. Conf. Conceptual Structures, ICCS'96 (Lecture Notes in Artificial Intelligence, vol. 1115)*, Sydney, Australia, pp. 248–262. Springer-Verlag.

[13] Mugnier, M. L. (1995) On generalization-specialization for conceptual graphs. *J. Exp. Theor. Artif. Intell.*, **7**, 325–344.

[14] Chang, C. C. and Keisler, H. J. (1990) *Model Theory* (3rd edn). North-Holland Publishing Co., Amsterdam.

[15] Mendelson, E. (1987) *Introduction to Mathematical Logic*. Wadsworth & Brooks, Monterey, California.

[16] Cao, T. H. and Creasy, P. N. (1997) Universal marker and functional relation: semantics and operations. In Delugach, H. S., Lukose, D., Keeler, M., Searle, M. L. and Sowa, J. F. (eds), *5th Int. Conf. on Conceptual Structures, ICCS'97 (Lecture Notes in Artificial Intelligence, Subseries of Lecture Notes in Computer Science, vol. 1257)*, Seattle, WA, pp. 416–430. Springer-Verlag.

[17] Dantsin, E., Eiter, T., Gottlob, G. and Voronkov, A. (1997) Complexity and expressive power of logic programming. In *Proc. 12th Ann. IEEE Conf. on Computational Complexity (CCC'97)*, Ulm, Germany.

[18] Jones, N. and Laaser, W. (1977) Complete problems in deterministic polynomial time. *Theor. Comput. Sci.*, **3**, 105–1177.

[19] Vardi, M. (1982) Complexity of relational query languages. In *Proc. 14th STOC Conf.*, San Francisco, pp. 137–146.

[20] Immerman, N. (1986) Relational queries computable in polynomial time. *Inf. Control*, **68**, 86–104.

[21] Sowa, J. F. (1999) *Knowledge Representation, Logical, Philosophical, and Computational Foundations*. Cole Publishers, Pacific Grove, CA.

[22] Lloyd, J. W. (1987) *Foundations of Logic Programming* (2nd edn). Springer-Verlag, Berlin.

[23] Robinson, J. A. (1965) A machine-oriented logic based on the resolution principle. *J. Assoc. Comput. Mach.*, **12**, 23–41.

[24] Chang, C. L. and Lee, R. C. T. (1973) *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York.

[25] Lenzerini, M. (1990) Class hierarchies and their complexity. In Banchilhon, F. and Boolemann, P. (eds), *Advances in Databases Programming Languages*. ACM Press, New York.

[26] Lenzerini, M. (1991) Careful closure of inheritance networks. In Lenzerini, M., Nardi, D. and Simi, M. (eds), *Inheritance Hierarchies in Knowledge Representation and Programming Languages*, pp. 97–112. John Wiley & Sons, Chichester, UK.

[27] Ounis, I. and Pasca, M. (1998) RELIEF: Combining expressiveness and rapidity into a single system. In *The ACM SIGIR'98 Conf.*, Melbourne, Australia, pp. 266–274.