Laboratoire d'Informatique, de Robotique
et de Microélectronique de Montpellier

Unité Mixte CNRS - Université Montpellier II C 09928

RAPPORT DE RECHERCHE

Sound and Complete Forward and Backward Chainings of Graph Rules

Eric Salvat and Marie-Laure Mugnier

January 96

R.R.LIRMM 96 004

161, Rue Ada - 34392 Montpellier Cedex 5 - France
Tél: (33) 67 41 85 85 - Fax: (33) 67 41 85 00

Sound and Complete Forward and Backward Chainings of Graph Rules

Eric Salvat and Marie-Laure Mugnier

L.I.R.M.M. *(U.M.R. 9928 Université Montpellier II / C.N.R.S.)*
161 rue Ada, 34392 Montpellier cedex 5 - France
tel.: (33) 67 41 85 45 fax: (33) 67 41 85 00
e-mail: {salvat,mugnier}@lirmm.fr

Abstract. This paper presents graph operations for processing conceptual graph rules in forward chaining and backward chaining. In both cases the operations provide sound and complete procedures with respect to first-order logic deduction. First we present our framework : simple conceptual graphs, rules as couples of lambda-abstractions, knowledge base, logical semantics. Next we focus on forward chaining. In particular, using the notion of redundancy, we exactly characterize when the application of a rule to a graph enriches or not this graph with a "new" information. The forward mechanism is complete if the knowledge base is in normal form. Basic notions (cut points, pieces, compatible partitions, unification) for backward chaining are detailed. A parallel with previous works on backward chaining is done, in particular with the work of B.C. Ghosh and V. Wuwongse, which is close to ours. The main difference is that we do not split the goals into trivial subgraphs (a relation and its neighbours). Instead, we determine cut points, which define arbitrary complex subgraphs, called pieces, that can be processed as a whole.

1 Introduction

This paper presents graph operations for processing conceptual graph rules in forward chaining and backward chaining. In both cases the operations allow one to obtain sound and complete procedures with respect to first-order logic deduction.

The framework (simple CGs, rules, logical semantics, knowledge base) is briefly defined in part 2. Part 3 is devoted to forward chaining. In particular, we exactly characterize when the application of a rule to a graph enriches or not this graph with a "new" information. In part 4, basic notions and operations for backward chaining are detailed. A parallel with previous works is done. In particular, the work of [GW95] is close to ours in that it presents a sound and complete procedure for CG rules in backward chaining, using graph operations. But there is a main difference in the way of processing the goal. Perspectives are outlined in part 5.

2 The Framework

2.1 Rules

By *simple conceptual graphs* (CGs), we mean non-nested conceptual graphs, in particular without negation (as defined in [Sow84], Chapter III). But we do not impose the CGs be connected. This choice proved to be convenient for some definitions and properties [MC95] [CM95]. Notably, it allows to consider a set of connected CGs as one CG. It is a technical choice, without semantic implications. Both viewpoints are coherent with the following intuitive semantics: if graphs G_1 and G_2 are asserted, then the graph G_1+G_2 is asserted (where + denotes the disjoint sum), and reciprocally.

A rule $R : G_1 \Rightarrow G_2$ is composed of two CGs G_1 and G_2, respectively called *hypothesis* and *conclusion* of the rule. There may be co-reference links between concepts of G_1 and G_2. If CGs were connected, then G_1 and G_2 would be sets of CGs. This kind of rules could be described with negative contexts i.e. $\neg[G_1 \neg[G_2]]$. But we prefer not to use this notation which would be misleading, since we do not process other kinds of graphs with negation, in particular with negative context at depth more than two. A useful notion is that of *lambda-abstraction*. A lambda-abstraction $\lambda x_1 ... x_n G$, where $n \geq 0$, consists of a CG G and n distinguished generic vertices of G, $x_1 ... x_n$. A CG can be seen as a particular case of lambda-abstraction with $n = 0$.

Definition 1. *A rule* $R : G_1 \Rightarrow G_2$ is a couple of lambda-abstractions $(\lambda x_1 ... x_n G_1, \lambda x_1 ... x_n G_2)$. $x_1 ... x_n$ are called *connection points.* They correspond to n co-reference links between G_1 and G_2. When needed, the occurrence of an x_i in G_1 (respectively in G_2) is denoted by x_{i_1} (resp. x_{i_2}). x_{i_1} and x_{i_2} must appear with the same type.

There are no constraints on types for individual markers common to G_1 and G_2: the type actually considered in operations is the smallest type that can be associated with the marker.

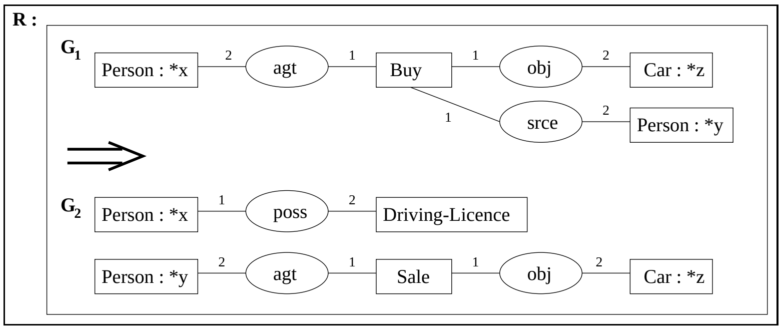


Fig. 1. A rule.

2.2 Logical semantics.

The classical function Φ associates to every lambda-abstraction $\lambda x_1 ... x_n G$ a formula of the first-order logic, where all variables are bound by an existential quantifier, except for the variables corresponding to the x_i, which are free. Given a rule $R : G_1 \Rightarrow G_2$, let $\Psi(R) = \Phi(\lambda(x_1...x_n)G_1) \rightarrow \Phi(\lambda(x_1...x_n)G_2)$, where $\rightarrow$ denotes the logic implication connector. Then, $\Phi(R)$ is the universal closure of $\Psi(R)$: $\Phi(R) = \forall x_1 ... \forall x_n \Psi(R)$.

E.g. for the rule R of Figure 1, $\Phi(R) = \forall x \forall y \forall z (\exists w (Person(x) \wedge Person(y) \wedge Car(z) \wedge Buy(w) \wedge agt(w,x) \wedge obj(w,z) \wedge srce(w,y)) \rightarrow \exists u \exists v (DrivingLicence(u) \wedge Person(x) \wedge poss(x,u) \wedge Sale(v) \wedge Person(y) \wedge Car(z) \wedge agt(v,y) \wedge obj(v,z)))$. Note that the following formula, where all variables of G_1 are bound by "global" universal quantifiers, is equivalent to the previous one: $\Phi(R) = \forall x \forall y \forall z \forall w ((Person(x) \wedge Person(y) \wedge Car(z) \wedge Buy(w) \wedge agt(w,x) \wedge obj(w,z) \wedge srce(w,y)) \rightarrow \exists u \exists v (Driving Licence(u) \wedge Person(x) \wedge poss(x,u) \wedge Sale(v) \wedge Person(y) \wedge Car(z) \wedge agt(v,y) \wedge obj(v,z)))$. This would correspond to an equivalent rule definition: $R : G_1 \Rightarrow G_2$ is a couple of lambda-abstractions $(\lambda x_1, ..., x_n, x_{n+1}, ..., x_p G_1, \lambda x_1 ... x_n G_2)$, where p is the number of generic vertices of G_1.

Let us point out differences between the logical formula associated to a rule and a clause. A clause is a disjunction of positive or negative literals, where all variables are universally quantified. A Horn clause has only one positive literal (such as rules in basic Prolog [Col83]: $P_1 \wedge P_2 ... \wedge P_n \rightarrow P_{n+1}$, where the P_i are positive atoms). One difference between a Horn clause and our rules is that the variables which are exclusive to the conclusion (P_{n+1}) are universally quantified, while they are existentially quantified in our formulas. Horn clauses can be seen as a special case of our rules only if they have no functions and all variables of the conclusion appear in the hypothesis. At these conditions, it becomes a very particular case of CG rule, since G_2 would be composed of only one relation whose all neighbours have an individual marker or a co-reference marker. Also note that a CG rule formula is not a particular case of clause.

2.3 Knowledge base.

We call a *support* the structure that represents the ontology of a specific domain application. It corresponds to the notion of canon in [Sow84]. Among others, it is composed a concept type lattice and a relation type set, which is partially ordered but is not necessarily a lattice. Two comparable relations must have the same arity. Our work still holds if the concept type set is only a partial order, but the unification step (see part 4) becomes more complex. A *knowledge base* is composed of a support S, a set of simple CGs, called *facts*, and a set of rules. The set of formulas associated to a knowledge base KB, denoted by $\Phi(KB)$, is the set of formulas associated to each component. For the support, we only need here the formulas associated to the partial orders on types: for all concept types t_1 and t_2, if t_2 is covered by t_1 (i.e. $t_2 < t_1$ and there does not exist a type t with $t_2 < t < t_1$; we only consider the edges of the transitive reduction), one has the formula: $\forall x \; t_2(x) \rightarrow t_1(x)$; for all relation types r_1 and r_2 with same arity p, if r_2 is covered by r_1 then: $\forall x_1 ... x_p \; r_2(x_1, ..., x_p) \rightarrow r_1(x_1, ..., x_p)$.

It is convenient to consider the empty CG (noted $G_\blacksquare$). Its logical formula is the valid formula denoted by $\blacksquare$. Then, a fact G can be seen a special rule, $R : G_\blacksquare \Rightarrow G$. It is immediate to check that $\Phi(G_\blacksquare \Rightarrow G) = \Phi(G)$. Therefore, facts can be processed as rules.

3 Forward Chaining

Forward chaining is typically used in order to explicitly enrich facts with knowledge which is implicitly present in the knowledge base (e.g. implicit knowledge associated with a concept type or rules of relation type composition). If G is a fact and R is a rule, if G fulfills the hypothesis of R, then the conclusion of R can be "added" to G. More precisely, the basic mechanism is as follows.

Definition 2 : Rule application. A rule $R : G_1 \Rightarrow G_2$, applies to a graph G if there is a projection, say Π, from G_1 to G. The resulting graph, denoted by $R(G)$, is built from G and G_2 by merging each x_{i_2} of G_2 with $\Pi(x_{i_1})$, the image of x_{i_1} by Π. In terms of canonical formation rules, the merging consists of the following steps:

1. In G_2, restrict the label of each connection point x_{i_2} to the label of $\Pi(x_{i_1})$.
2. In G_2, join all connection points having the same image into a single vertex.
3. Join each x_{i_2} with $\Pi(x_{i_1})$ in G.

Figure 2 illustrates a rule application with a non-injective Π.

3.1 Soundness and completeness of the forward mechanism.

Let us first recall some results about the logical semantics of CGs. Given two CGs G and H, $G \leq H$ means that G can be derived from H by the canonical formation rules, or "specialization rules" [Sow84]. Sowa proved that given two CGs G and H, if $G \leq H$ then $\Phi(G), \Phi(S) \vdash \Phi(H)$, where S is the support. In other words, the inverse rules (say generalization rules) constitute a sound set of

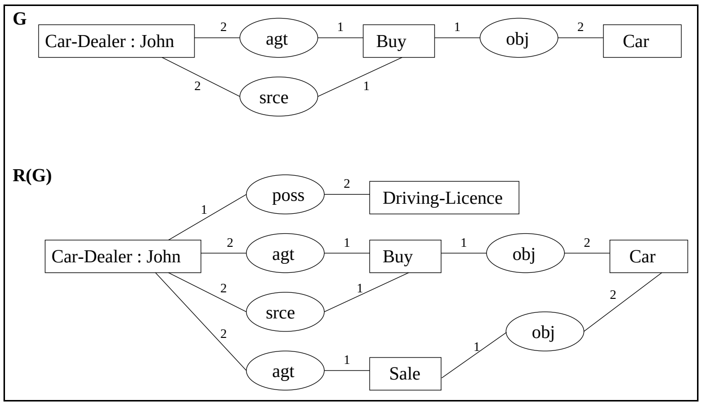
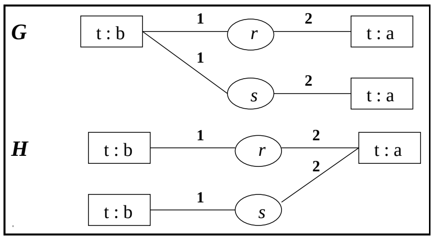


Fig. 2. G and $R(G)$ (where R is the rule of figure 1).

inference rules on CGs. He also proved that if $G \leq H$ then there is a projection from H to G. [CM92] proved the reciprocal results. If there is a projection from H to G, then $G \leq H$ ([MC93] extends this result to canonical graphs). And the generalization rules constitute a *complete* set of inference rules. In order to obtain the completeness property, CGs must be in *normal form*: all individual vertices of a graph must have different markers. If CGs are not normal, then some logical deductions can not be simulated (Figure 3 [1]). Let us point out that this notion corresponds to the normal form concurrently elaborated by [GW95].

Theorem 1. *[CM92][MC95]. Let $\Phi(S)$ be the set of formulas associated with a support S. Let g and h be well-formed formulas associated with normal CGs G and H. If g, $\Phi(S) \vdash h$, then $G \leq H$ (there is a projection from H to G).*



$\Phi(G) = t(b) \wedge t(a) \wedge t(a) \wedge r(b,\ a) \wedge s(b,\ a)$.
$\Phi(H) = t(b) \wedge t(b) \wedge t(a) \wedge r(b,\ a) \wedge s(b,\ a)$.
$\Phi(G)$ and $\Phi(H)$ are equivalent, while G and H are not comparable for $\leq$.
Fig. 3. The need for normal form

These results are now extended to rules.

Theorem 2 Soundness of forward chaining. *Let KB be a knowledge base. If a CG G is obtained from KB by a sequence of rule applications, then $\Phi(KB) \vdash \Phi(G)$.*

[1] From A. Preller, private communication, June 94.

Lemma 1. *Let KB be a knowledge base. Let H and R be respectively a fact and a rule of KB. If a CG G is obtained from H by one application of R, then $\Phi(H), \Phi(R), \Phi(S) \vdash \Phi(G)$. (A fortiori, $\Phi(KB) \vdash \Phi(G)$).*

Proof. The inconsistency of $\{\Phi(H), \Phi(R), \Phi(S), \neg\Phi(G)\}$ is proved with the resolution method. To that end, we build a resolution ending with $\Box$ (the empty clause), which erases the predicates of $\neg\Phi(G)$. First, the formulae are put into Skolem form.
$\Phi(H)$ is of type: $\exists x_1 ... \exists x_l \; P_1 \wedge ... \wedge P_n$, where $x_1, ..., x_l$ are variables and $P_1, .., P_n$ are the predicates associated with the vertices of H. The variables, which are existentially quantified, are replaced by Skolem constants, say $a_1, ..., a_l$. n clauses of type $P_i[a_1, ..., a_l]$ $(1 \leq i \leq n)$ are obtained[2]. Let $R : G_1 \Rightarrow G_2$. $\Phi(R)$ is of type:

$$\forall x_1 ... \forall x_i (\exists x_{i+1} ... \exists x_{i+j} \; Q_{1.1} \wedge ... \wedge Q_{1.m_1} \rightarrow \exists y_1 ... \exists y_k \; Q_{2.1} \wedge ... \wedge Q_{2.m_2}),$$

where $x_1, .., x_i$ are the variables associated with the connection points of G_1, $x_{i+1}, .., x_{i+j}$ and $y_1, .., y_k$ are the variables associated with the other generic vertices of respectively G_1 and G_2. $Q_{1.1}, ..., Q_{1.m_1}$ and $Q_{2.1}, ..., Q_{2.m_2}$ are respectively the predicates associated with the vertices of G_1 and G_2. This formula is first put into normal conjunctive prenex form:

$$\begin{aligned} \forall x_1 ... \forall x_i ... \forall x_{i+j} \exists y_1 ... \exists y_k \; & (\neg Q_{1.1} \vee ... \vee \neg Q_{1.m_1} \vee Q_{2.1}) \wedge ... \\ & \wedge (\neg Q_{1.1} \vee ... \vee \neg Q_{1.m_1} \vee Q_{2.i}) \wedge ... \\ & \wedge (\neg Q_{1.1} \vee ... \vee \neg Q_{1.m_1} \vee Q_{2.m_2}) \end{aligned}$$

In order to obtain the Skolem form, the variables $y_1, ..., y_k$ are replaced by Skolem functions: $f_1(x_1, ..., x_{i+j}), ..., f_k(x_1, ..., x_{i+j})$. Then, m_2 clauses of form: $(\neg Q_{1.1} \vee ... \vee \neg Q_{1.m_1} \vee Q_{2.i})[x_1, ..., x_{i+j}, f_1(x_1, ..., x_{i+j}), ..., f_k(x_1, ..., x_{i+j})]$ where $1 \leq i \leq m_2$, are obtained. $\neg\Phi(G)$ is of type: $\neg(\exists z_1 ... \exists z_h R_1 \wedge ... \wedge R_p)$, which gives: $(\neg R_1 \vee ... \vee \neg R_p)[z_1, ..., z_h]$.

Step 1. Since G is built by the application of R on H, let Π be the associated projection from G_1 to H. From [Sow84] [CM92] $\Phi(G_1)$ is a logical consequence of $\Phi(H)$ and $\Phi(S)$: $(\Phi(H), \Phi(S) \vdash \Phi(G_1))$. So $\{\Phi(H), \Phi(S), \neg\Phi(G_1)\}$ is inconsistent. There is a resolution from this set to $\Box$, that corresponds to Π. $\neg\Phi(G_1)$ is $\neg Q_{1.1} \vee ... \vee \neg Q_{1.m_1}$, then, the previous resolution can be applied to the clauses from $\Phi(R)$ with the clauses from $\Phi(H)$ and $\Phi(S)$. The produced clauses are of type $Q_{2.i}[a_1, ..., a_l, f_1(a_1, ..., a_l), ..., f_k(a_1, ..., a_l)]$ where $1 \leq i \leq m_2$.

Step 2. G is obtained by joining H and a specialization of G_2. Then, all predicates of $\Phi(H)$ are also predicates of $\Phi(G)$. For each clause $P_i[a_1, ..., a_l]$ from $\Phi(H)$, there is a predicate $R_{i'}$ in $(\neg R_1 \vee ... \vee \neg R_p)[z_1, ..., z_h]$ such that $P_i = R_{i'}$. The resolution method allows one to erase all the $R_{i'}$ of this type in $\neg\Phi(G)$.

Step 3. Now, in $\neg\Phi(G)$ only negative predicates from G_2 remain. At this point the predicates corresponding to connection points of G_2 have been erased. Indeed, when building G, connection points of G_2 are the vertices on which the join is done. Then, in $\Phi(G)$, such vertices correspond to predicates from $\Phi(H)$, and they have been erased during step 2. For each remaining negative predicate, there is a corresponding positive predicate built in the first step from $\Phi(H)$ and $\Phi(R)$. Since, in step 2 the predicates from G_1 are erased with respect to the projection Π, then the substitution used is compatible with a substitution used for erasing predicates from H in G.

[2] The notation $F[x_1, ..., x_k]$, where F is a formula, means that the terms of F (variables/constants) are $x_1, ..., x_k$

A resolution ending by $\Box$ has been built. The inconsistency of the set $\{\Phi(H), \Phi(R), \Phi(S), \neg\Phi(G)\}$ is proved. Then $\Phi(H), \Phi(R), \Phi(S) \vdash \Phi(G)$.

Proof of the theorem. By induction on the number of rule applications.

Note that any knowledge base can be seen as a set of rules and a unique fact graph. Indeed, a set of graphs is equivalent to a unique non-connected graph obtained by performing the disjoint sum of all the graphs in the set. A knowledge base is said to be in *normal form* if:

- for each rule $R : G_1 \Rightarrow G_2$, G_1 and G_2 are in normal form.
- the set of facts considered as one global graph is in normal form (i.e each individual marker occurs at most once in the facts).

In order to ensure that only normal graphs are produced in forward chaining, we add the following step to a rule application:

4. Join the vertices of G_1 and G with same individual marker.

Property 1 *Let G be a normal graph and R be a normal rule. If R is applied on G with the four steps then $R(G)$ is in normal form.*

Theorem 3 Completeness of forward chaining. *Let KB be a normal knowledge base. Let g be a well-formed formula associated with a normal CG G. If $\Phi(KB) \vdash g$, then there is a CG H that can be obtained from KB by a sequence of rule applications, such that $H \leq G$ (i.e. $\Phi(H) \vdash g$).*

Let us first define the notion of an *S-substitution* (from [CM92]).

Definition 3. Let f and g be two well-formed formulae associated with normal CGs. An *S-substitution* from g to f is a mapping σ which, to every atom of g, associates a term or atom of f, in the following way:

- if a is a constant of g: $\sigma(a) = a$ (so a must be a constant of f),
- if x is a variable of g: $\sigma(x)$ is a variable or a constant of f,
- if t is the unary predicate associated with the concept type t: $\sigma(t) = t'$ where $t' \leq t$,
- if r is the predicate associated with the relation r :$\sigma(r) = r'$ where $r' \leq r$,
- in addition, the image by σ of every atom $v(x_1, ..., x_k)$ of g is an atom $\sigma(v)(\sigma(x_1), ..., \sigma(x_k))$ of f.

The following property holds (the $\Leftarrow$ part is not true if G and H are not normal CGs): there is a projection from G to H if and only if there is an S-substitution from $\Phi(G)$ to $\Phi(H)$.

Proof of theorem 3. Let GF be the fact graph of KB and Re be the set of rules of KB. Since $\Phi(KB) \vdash g$, the set $\{\Phi(KB), \neg g\}$ is inconsistent. The resolution method allows one to produce the empty clause from $\{\Phi(KB), \neg g\}$. We build a graph H, such that $H \leq G$, by following a resolution from $\{\Phi(KB), \neg g\}$ to $\Box$. As previously (proof of lemma 1), the formulae are put in Skolem form.

1. $\Phi(GF)$ is of type: $P_1 \wedge ... \wedge P_n[a_1, ..., a_l]$, where $P_1 ..., P_n$ are the predicates from GF and $a_1, ..., a_l$ are Skolem constants. Then, n clauses of type $P_i[a_1, ..., a_l]$, with $1 \leq i \leq n$, are built.
2. Let $Re = \{R_1, ..., R_t\}$, then $\Phi(Re) = \Phi(R_1) \wedge ... \wedge \Phi(R_t)$. For each $(R : G_1 \Rightarrow G_2)$ in Re, let m_1 and m_2 be respectively the number of vertices of G_1 and G_2. Then, m_2 clauses of type $(\neg Q_{1.1} \vee ... \vee \neg Q_{1.m_1} \vee Q_{2.m})[x_1, ..., x_{i+j}, f_1(x_1, ..., x_{i+j}), ..., f_k(x_1, ..., x_{i+j})]$ with $1 \leq m \leq m_2$ are built. $Q_{1.1}, ..., Q_{1.m_1}$ are the predicates from G_1, the $Q_{2.m}$ are the predicates from G_2, $x_1, ..., x_{i+j}$ are the variables from G_1 and $f_1(x_1, ..., x_{i+j}), ..., f_k(x_1, ..., x_{i+j})$ are the Skolem functions of the clauses.
3. $\neg g$ is of type: $(\neg R_1 \vee ... \vee \neg R_p)[z_1, ..., z_h]$, $R_1, ..., R_p$ are the predicates of g and $z_1, ..., z_h$ are its variables.

If $\Box$ can be obtained without any clause from a rule (i.e. with clauses from $\{\Phi(GF), \neg g\}$ only) there is an S-substitution from g to $\Phi(GF)$, then a projection from G to GF, and $H = G$.

Otherwise, at least one clause from $\Phi(Re)$ is used in the resolution. Each predicate of $\neg g$, which is a disjunction of negative predicates, has to be erased with a positive predicate. The only available predicates are the ones from $\Phi(GF)$ and the predicates $Q_{2.m}$ in the clauses from $\Phi(Re)$. To obtain a predicate $Q_{2.m}$, the clause $(\neg Q_{1.1} \vee ... \vee \neg Q_{1.m_1} \vee Q_{2.m})$ where it appears, has to be emptied. We know that the resolution uses at least one of these predicates. Let $(\neg Q_{1.1} \vee ... \vee \neg Q_{1.m_1} \vee Q_{2.m})$, obtained from a rule $R : G_1 \Rightarrow G_2$, be the first clause of this type used in the resolution. Emptying this clause is necessarily done by using the clauses from $\Phi(GF)$, which are the only available positive predicates. This resolution sub-sequence defines an S-substitution from $\Phi(G_1)$ to $\Phi(GF)$. The S-substitution defines a projection from G_1 to GF, thus a rule application on GF.

Then, the set of available positive clauses are the ones of $\Phi(R(GF))$. Indeed, the S-substitution can be applied to each clause from $\Phi(R)$, which produces m_2 predicates of type $Q_{2.m}[a_1, ..., a_l, f_1(a_1, ..., a_l), ..., f_k(a_1, ..., a_l)]$ corresponding to the specialization of G_2 in the application of R. These predicates can be used for erasing some predicates of $\neg g$. This process is repeated for each clause from a rule occurring in the resolution: the S-/substitution from the clause to the current fact define a rule application. According to the resolution emptying $\neg g$ and as described previously, a set $R_1, ..., R_n$ of rule applications is built such that $\{R_1(...(R_n(GF))...), \neg g\}$ is inconsistent. We obtain $H = R_1(...(R_n(GF))...)$.

3.2 Adding non redundant information

Obviously it is not valuable to apply a rule to a graph if this rule does not bring any new information. A trivial case is that, if a rule can be applied once, then it can be applied again to the same graph in the same way (i.e. following the same projection) indefinitely. More generally, we say that two graphs G_1 and G_2 are equivalent if $G_1 \leq G_2$ and $G_2 \leq G_1$ (or there is a projection from G_2 to G_1 and one from G_2 to G_1; or $\Phi(G_1)$ is equivalent to $\Phi(G_2)$); the application of a rule R brings a new information to a graph G if $R(G)$ and G are not equivalent. In the opposite case, note there may be another projection from G_1 to G giving another image to at least one x_i, that brings new information).

A CG G may itself contain redundant information: G is said to be *redundant* if it is equivalent to one of its strict subgraphs, in other words if there is a projection from G to one of its strict subgraphs. Otherwise, G is said to be *irredundant.*

Property 2 *Let G be an irredundant CG and $R : G_1 \Rightarrow G_2$ be a rule. Let Π_1 be the projection from G_1 to G, used to build $R(G)$. $R(G)$ is equivalent to G if and only if there is a projection from G_2 to G, say Π_2, such that for each connection point $x_i, \Pi_2(x_{i_2}) = \Pi_1(x_{i_1})$.*

Proof. $\Rightarrow$ Suppose there is a projection from $R(G)$ to G. ([CG95], property 6) proves the following property for connected CGs: if H is a redundant graph and K is an irredundant subgraph of H equivalent to H, then there is a projection from H to K which keeps every vertex in K invariant. Such a projection is called a "folding" from H to K. This property is also true for non-connected CGs. Let Π be a folding from $R(G)$ to G. In particular $\Pi(x_i) = x_i$, for each vertex x_i of $R(G)$ which results from the merging of a connection point x_{i_2} and a concept $\Pi_1(x_{i_1})$ of G. Then the restriction of Π to G_2 is a projection which keeps each x_i invariant.

$\Leftarrow$ If there is such a projection Π_2, then there is a projection from $R(G)$ to G, built as follows: each vertex coming from G_2 is projected onto $\Pi_2(G_2)$; each other vertex is projected onto itself. Since there is also a projection from G to $R(G)$ (the identity mapping), G and $R(G)$ are equivalent.

The part $\Rightarrow$ of the property is not true if G is a redundant graph.

An interesting feature would be the ability to apply just the part of a rule (i.e. a part of G_2) that brings new information, in order to obtain: if G is irredundant, then $R(G)$ also is.

4 Backward Chaining

4.1 Pieces

Consider a request (or a goal) to be proved. One basic guideline of our work is the following. Instead of splitting the request into trivial subgraphs composed of one relation and its arguments (as in [FLDC86] [GW95] and similarly to Prolog [Col83]), we want to determine subgraphs as large as possible, that can be processed as a whole. Such subgraphs are called pieces.

Definition 4. Let $R : G_1 \Rightarrow G_2$ be a rule. A *cut point* of G_2 is either a connection point (i.e. a generic concept shared with G_1) or a concept with an individual marker (which may be common with G_1 or not). A *cut point* of G_1 is either a connection point of G_1 or a concept with an individual marker which is common with G_2. The *pieces* of G_2 are obtained as follows: remove from G_2 all cut points; one obtains a set of connected components; some of them are not CGs since some edges have lost an extremity. Complete each incomplete edge with a concept with same label as the former cut point. Each connected component is a piece. Equivalently, two vertices of G_2 belong to the same piece if and only if there exists a path from one vertex to the other that does not go through a cut point.

E.g. in figure 4 c_1 and c_2 are cut points of G_1. G_2 has cut points c_4, c_5 and c_7. G_1 is a piece, and G_2 has three pieces respectively delimited by c_3 and c_4, c_4 and c_5, c_5 and c_7.

As mentioned in ([GW95], figure 4), projection can not be used as the basic operation if one wants a *complete* mechanism. Instead, a *unification* between a request and a rule is defined. If there is a unification between a request Q and a rule R then there is a CG G such that $R(G)$ (the application of R on G in forward chaining) is a specialization of Q.

4.2 Preliminary definitions

Definition 5. We call *greatest lower bound (glb)* of two concepts c_1 and c_2 (noted $c_1 \wedge c_2$), if it exists, the concept c labelled $(t : m)$ such that:

- $t = type(c_1) \wedge type(c_2)$, if this minorant is not $\perp$ (the absurd type)
- $m = *$ if $referent(c_1) = referent(c_2) = *$, or,
  $m = m'$ (individual marker) if $referent(c_1) = m'$ and $referent(c_2) = *$, or $referent(c_1) = *$ and $referent(c_2) = m'$, or $referent(c_1) = referent(c_2) = m'$.

Such a lower bound does not exist when the lower bound of the concept types is $\perp$, or the label $(t : m)$ does not respect conformity, or c_1 and c_2 are individual concepts with distinct referents.

Definition 6. The glb of a concept set C, such that $|C| \geq 2$, noted $\bigwedge C$, is recursively defined as follows:

- if $|C| = 2$, i.e. $C = \{c_1, c_2\}$, then $\bigwedge C = c_1 \wedge c_2$.
- else, let $c_1 \in C$ and $c_2 \in C$ then $\bigwedge C = \bigwedge(C \backslash \{c_1, c_2\} \cup \{c_1 \wedge c_2\})$.

Definition 7. Let C_1 and C_2 be two concept sets. Let $P_1 = \{p_{1.1}, p_{1.2}, ..., p_{1.n}\}$ and $P_2 = \{p_{2.1}, p_{2.2}, ..., p_{2.n}\}$ be two partitions respectively of C_1 and C_2. P_1 and P_2 are said to be *compatible partitions* if for all $i \in [1, n]$ $\bigwedge(p_{1.i} \cup p_{2.i})$ exists.

Definition 8. Let G_1 and G_2 be two CGs with concept sets C_1 and C_2. Let P_1 and P_2 be two compatible partitions defined on $C'_1 \subseteq C_1$ and $C'_2 \subseteq C_2$. The *specialization of* G_1 *according to compatible partitions* P_1 *and* P_2 is the CG built from G_1 as follows:

- For every $c_i \in C'_1$, let $p_{1.j} \in P_1$ such that $c_i \in p_{1.j}$: the label of c_i is specialized into $\bigwedge(p_{1.j} \cup p_{2.j})$.
- then, for all $p_{1.i} \in P_1$, join together the concepts of $p_{1.i}$.

The specialization of G_2 according to compatible partitions P_1 and P_2 can be defined in the same way.

Definition 9. Let G_1 and G_2 be two CGs and P_1 and P_2 be two compatible partitions on subsets of G_1 and G_2 concept sets. The *join of* G_1 *and* G_2 *according to compatible partitions* P_1 *and* P_2 is the CG obtained as follows:

- let G'_1 and G'_2 be the specializations of G_1 and G_2 according to P_1 and P_2.
- for each $p_{1.i} \in P_1$ (resp. $p_{2.i} \in P_2$), let $c_{1.i}$ (resp. $c_{2.i}$) be the vertex obtained by merging all vertices in $p_{1.i}$ (resp. $p_{2.i}$). Then, for every $p_{1.i} \in P_1$, join $c_{1.i}$ and $c_{2.i}$.

4.3 Basic operations

Let us now define a unification between a request Q and a rule $R : G_1 \Rightarrow G_2$.

Definition 10. Let Q denote a CG request and $R : G_1 \Rightarrow G_2$ be a rule. There is a *unification* between Q and R, say $u(Q, R) = \{P_{G_2}, P_Q, \Pi_u\}$ if there are:

- two compatible partitions P_{G_2} and P_Q respectively defined on a subset of cut points of G_2 and a subset of concepts of Q; let $u(Q)$ and $u(G_2)$ be the specializations of Q and G_2 according to P_{G_2} and P_Q.
- and a projection Π_u from some pieces (at least one) of $u(Q)$ to $u(G_2)$. This projection has to respect the compatibility of P_{G_2} and P_Q, i.e. the image of each concept (of $u(Q)$) built from $p_{1.i} \in P_Q$ must be the corresponding concept (of $u(G_2)$) built from $p_{2.i} \in P_{G_2}$.

Once a unification $u(Q, R)$ between Q and R has been found, a new request (say Q') is built according $u(Q, R)$. Q' is such that $R(Q')$ is a specialization of Q.

Definition 11.

1. Construct P'_{G_2} and P'_Q from P_{G_2} and P_Q including only vertices of a piece on which Π_u is defined (i.e. cut points of G_2 having an image by Π_u and their image by Π_u).
2. Let C'_1 be the set of G_1 cut points which correspond to cut points of P'_{G_2}. Then P_{G_1} is the partition of C'_1 with the same structure as P'_{G_2}: any two cut points of C'_1 are in the same element of P_{G_1} if and only if the corresponding vertices in G_2 are in the same element of P'_{G_2}.
3. Build Q' and G'_1 the specializations of Q and G_1 according to P_{G_1} and P'_Q.
4. Erase the pieces of Q' which project to G_2; then join Q' and G'_1 according to P_{G_1} and P'_Q.

A resolution of a request Q is a sequence of unifications. It ends successfully if the last produced request is $G_\blacksquare$. The underlying principle is to build Q' such that if one applies in forward chaining and in reverse order the rules successively used in the unifications, one produces a specialization of the initial request Q. Then a "proof" of Q can be built from a "proof" of Q' ant the application of R on Q'.

4.4 An example of Backward Chaining

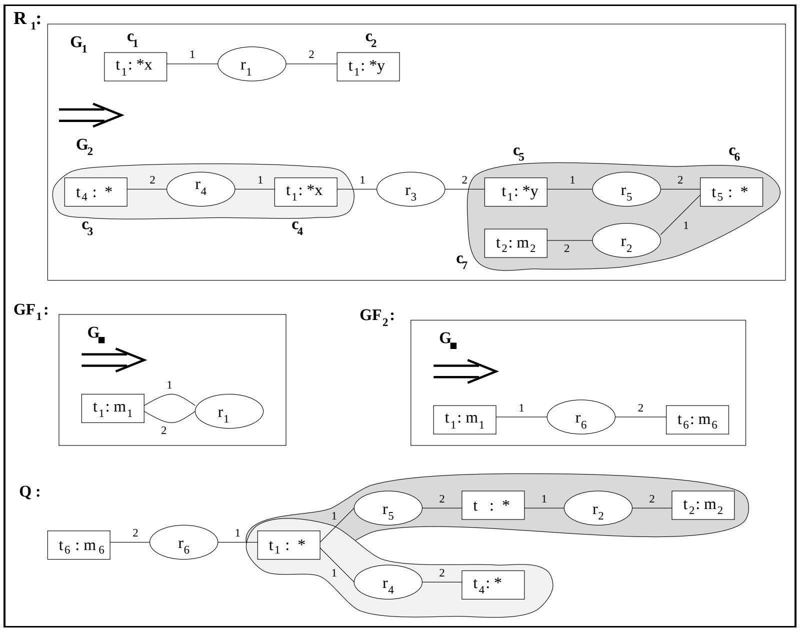


Fig. 4. A knowledge base with a request.

Figure 4 presents a knowledge base with one rule R_1, two facts GF_1 and GF_2 which are described as rules with hypothesis $G_\blacksquare$, and a request Q. Suppose a unification between Q and R_1 is first found: the dark grey part of Q is unified with the dark grey one of R_1, and the same is done for light grey parts. We obtain the compatible partitions $\{[t_1 : *]\}$ for Q and $\{[t_1 : *x], [t_1 : *y]\}$ for G_2. Remark that two distinct cut points of G_2 are unified with the same vertex of Q. According to the operations described previously, a new request Q' is built: the grey parts of Q are erased, labels of cut points do not change, and G'_1 and Q are joined to build the new request Q' (figure 5-left).

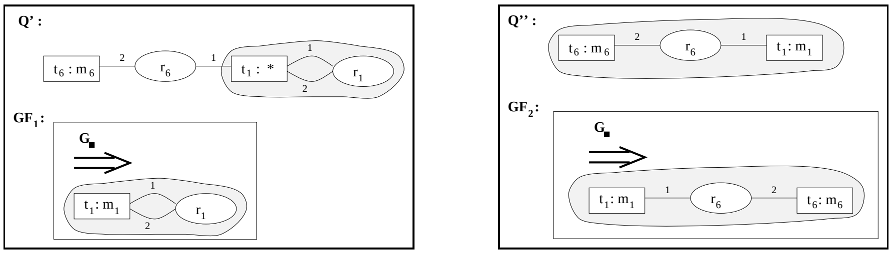


Fig. 5. New requests built by unifications.

As shown in the first part of figure 5, there is a unification between Q' and GF_1 materialized by the grey part. The label of the cut point is specialized into $(t_1 : m_1)$. The hypothesis of GF_1 being $G_{\blacksquare}$, there are no joins to do, only pieces are to be erased.

Finally, the last unification between Q'' and GF_2 produces an empty request. The resolution ends successfully. Then keeping trace of previous unifications, a specialization of the initial request Q can be built (figure 6).

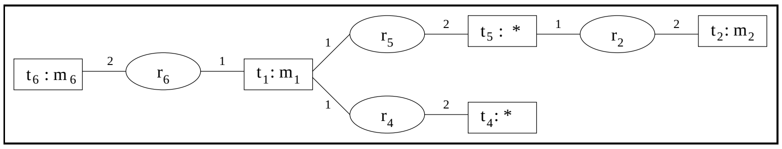


Fig. 6. A specialization of the initial request.

4.5 Soundness and completeness of backward chaining

The proofs of soundness and completeness of the backward mechanism are based on the results about the forward mechanism. Let KB be a knowledge base and Q be a request. Then, for each specialization of Q built in backward chaining there exists a forward chaining application of rules of KB producing this specialization. And, if a specialization of Q can be produced in forward chaining on KB, then it can be built in backward chaining.

Theorem 4 Soundness of backward chaining. *Let KB be a knowledge base and Q be a request. If a resolution*[3] *of Q in backward chaining on KB ends successfully, then $\Phi(KB) \vdash \Phi(Q)$.*

Lemma 2. *Let Q be a request and R be a rule. If Q' is the new request built by a unification of Q and R, then $\Phi(Q'), \Phi(R), \Phi(S) \vdash \Phi(Q)$.*

Proof of the lemma. We prove that R can be applied on Q' (in forward chining), and $R(Q') \leq Q$. Then $\Phi(R(Q')) \vdash \Phi(Q)$. Using lemma 1, $\Phi(Q'), \Phi(R), \Phi(S) \vdash \Phi(R(Q'))$. Then lemma 2 is proved. Q' is built from a specialization of G_1. There is a projection from G_1 to Q'. We choose the one that is naturally associated with Q' building. Now $R(Q')$ is built by joining Q' and a specialization of G_2 which is exactly the $u(G_2)$ of definition 10 (i.e. the specialization of G_2 according to the compatible partitions of the unification leading to Q'). Indeed, $u(G_2)$ is obtained from G_2 by specializing connection points only. These points are specialized as their images by the projection to Q' . Now a projection from Q to $R(Q')$ can be built extending the partial projection from Q to $u(G_2)$. Indeed, the parts of Q which do not have an image by this projection are also parts of $R(Q')$, because they are kept during the building of Q'. Since there is a projection from Q to $R(Q')$, then $\Phi(R(Q')), \Phi(S) \vdash \Phi(Q)$. In addition, $\Phi(Q'), \Phi(R), \Phi(S) \vdash \Phi(R(Q'))$ due to the soundness of forward chaining. Then: $\Phi(Q'), \Phi(R), \Phi(S) \vdash \Phi(Q)$

Proof of the theorem. By induction on the number of unifications.

Theorem 5 Completeness of backward chaining. *Let KB be a knowledge base and Q be a request. If $\Phi(KB) \vdash \Phi(Q)$, then there is a resolution of Q in backward chaining on KB that ends successfully.*

Lemma 3. *Let G, R and Q be respectively a CG, a rule and a request such that $R(G) \leq Q$ and fulfils the property: at least one vertex of Q projects to a vertex of $R(G)$ issued from G_2, which is not a connection point. Then there is a unification of Q and R producing Q' such that $G \leq Q'$.*

Proof of the lemma. Let Π_1 be the projection from G_1 to G used to apply R on G. As $R(G) \leq Q$, there exists a projection Π from Q to $R(G)$ fulfilling the property of the lemma 3. If no concepts of Q project to a cut point of $R(G)$, then $\pi(Q)$ is included in a piece of $R(G)$. As at least one vertex of Q projects to a vertex coming from G_2, this piece comes from G_2. So, $G_2 \leq Q$. Empty compatible partitions allow us to define a unification between Q and G_2 (Q consists of one piece which projects to G_2). The new request is $Q' = G_1$, which can be projected to G (by Π_1).

In the other case, let us call "cut points of Q" the vertices of Q whose image by Π is a cut point of $R(G)$. Then, pieces of Q can be defined. As at least one vertex of Q projects to a vertex coming from G_2, there is at least one piece of Q which projects to a piece from G_2. Then, a unification between Q and $R(G)$ can be defined. The compatible partitions are defined as follows. In Q, the partition is defined on the set of cut points. All cut points of Q which have the same image by Π in $R(G)$, and which are members of a piece which projects to a piece coming from G_2, are members of the same set of the partition. In G_2, each set of the partition contains all the cut points

[3] The term "resolution" means here "sequence of unifications"

of G_2 which are joined in the building of $R(G)$. The projection of the unification is defined on the pieces of Q which project by Π to a piece of $R(G)$ coming from G_2. A new request can be built, by joining a specialization of G_1 (only on cut points) and the pieces of Q which project by Π to pieces coming from G. Schema 7 presents the differents projections. Full-lines are projections. Π' is the projection from Q' to G we want to build. And dash-lines link Q' and the graphs which participate in its building.

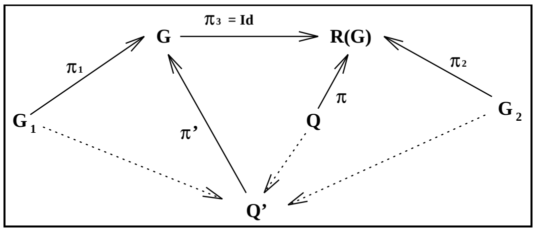


Fig. 7. Schema of the projections.

The projection Π' from Q' to G is built as follows. Pieces of Q' issued from Q have the same image as by Π. Indeed, these pieces are the ones which project by Π to pieces of $R(G)$ coming from G. The other pieces of Q' come from G_1, their image by Π' is the same as the one they had by Π_1. Π' is totally defined.

Proof of the theorem. The proof is done by induction on the number of rule applications building G in forward chaining. The induction hypothesis is the following. If the forward chaining mechanism on KB builds $G \leq Q$ in n or less rule applications, then there is a resolution of Q in backward chaining on KB that ends successfully. The hypothesis is true for $n = 0$. Indeed, $n = 0$ means G is a fact of KB. Since $G \leq Q$, there exists a unification between Q and the rule ($G_\blacksquare \Rightarrow G$) which builds as new request $G_\blacksquare$. Suppose the induction hypothesis is true at range n. Now, consider a CG $G \leq Q$ built in $n+1$ rule applications. The n first rule applications in forward chaining build a CG H such that, if the last rule application is noted R_{n+1}, $R_{n+1}(H) \leq Q$. If, for all projections Π from Q to $R_{n+1}(H)$, $\Pi(Q)$ is a subgraph coming from H, one considers $R_1, ..., R_n$ only (which is correct since $H \leq Q$). Otherwise, from lemma 3, there is a unification of Q and R_{n+1} producing Q' such that H is a specialization of Q'. By induction hypothesis, the resolution of Q' in backward chaining ends successfully. Thus the unification of Q and R_{n+1} produces a request whose resolution ends successfully.

Let us point out differences between our work and previous works [FLDC86], [RF87] and [GW95]. [FLDC86] and [RF87] present systems based on Prolog-like resolution. There are no theoretical results on soundness and completeness. A closer work is [GW95] which presents a sound and complete procedure.

One main difference lies in the notion of pieces. The underlying idea is to process as much as possible the graph as a whole. In [GW95] the request is put into "anti-normal form": it is splitted into trivial subgraphs restricted to a relation and its neighbours. Then, the unification is performed on each trivial subgraph.

Pieces are an alternative to the maximal split of the request. In addition, pieces allow one to perform splits which are guided by the rule. Indeed, during the unification, we only compute very specific kind of common specialization to the request and the rule conclusion. First note, in forward chaining the only vertices of the rule which may be specialized are connection points. Pieces are joined to the fact graph on which the rule is applied, without any modification except for connection points. Considering pieces in backward chaining, the unification takes into account the previous property. Then, a greatest lower bound is computed only between cut points of the rule and corresponding concepts of the request. This consists of searching a common specialization of the request and the conclusion G_2 of R where cut points are the only concepts of G_2 which may be specialized. Finally, as shown in the example of section 4.4, it is not necessary to put the knowledge base in normal form as in [GW95]. Considering individual referents as cut points in CG facts the successive unifications find one after other the pieces linked to them. Note that CGs in rules have to be in normal form because of the projections searched in the unification process.

5 Conclusion.

In this paper, operations for processing CG rules in forward and backward chaining are precisely defined. Both mechanisms are sound and complete relative to the deduction in first-order logic.

Concerning forward chaining, we exactly characterize when the application of a rule to a graph adds a "new" information to this graph. But a more interesting result would be the characterization (and the efficient computation) of which part of the rule brings a "new" information and which does not, in order to partially apply the rule. Another problem is to determine which conditions on the knowledge base and which order(s) on rule firing could ensure that the process stops.

Concerning backward chaining, a close work is [GW95]. The essential difference between their operations and ours is that we do not split the conclusion part of the rule into subgraphs restricted to one relation and its neighbours. We determine cut points, which define arbitrary complex subgraphs, called pieces, that can be processed as a whole. Beside the interest of a new sound and complete method, the question of whether we gain complexity in processing pieces instead of trivial subgraphs remains to be explored.

We are currently involved in the design of data structures and algorithms for implementing both chainings. They will be integrated into Corali, a development environment for knowledge-based applications using exclusively conceptual graphs [CBC+95].

References

[CBC+95] M. Chein, J. Bouaud, J.P. Chevallet, R. Dieng, B. Levrat, and G. Sabah. Graphes Conceptuels. In *Actes des 5emes Journees Nationales du PRC-GDR Intelligence Artificielle*, pages 179–212, 1995.

[CG95] O. Cogis and O. Guinaldo. A Linear Descriptor for Conceptual Graphs and a Class for Polynomial Isomorphism Test. In G. Ellis, R. Levinson, W. Rich, and J.F. Sowa, editors, *Lecture Notes in Artificial Intelligence, 954, Proceedings of ICCS'95*, pages 263–277. Springer Verlag, 1995.

[CM92] M. Chein and M.L. Mugnier. Conceptual Graphs: fundamental notions. *Revue d'Intelligence Artificielle*, 6(4), 1992. En anglais.

[CM95] M. Chein and M.L. Mugnier. Conceptual graphs are also graphs. Research Report 95-004, LIRMM, Jan. 1995. 17 pages.

[Col83] A. Colmerauer. Prolog in ten figures. In *Proceedings IJCAI'83, Vol. 1*, pages 487–497, 1983.

[FLDC86] J. Fargues, M.C. Landau, A. Dugourd, and L. Catach. Conceptual Graphs for Semantics and Information Processing. *IBM Journal of Research and Development*, 30(1):70–79, 1986.

[GW95] B.C. Ghosh and V. Wuwongse. A direct Proof Procedure for Definite Conceptual Graph programs. In G. Ellis, R. Levinson, W. Rich, and J.F. Sowa, editors, *Lecture Notes in Artificial Intelligence, 954, Proceedings of ICCS'95*, pages 158–172. Springer Verlag, 1995.

[MC93] M.L. Mugnier and M. Chein. Characterization and Algorithmic Recognition of Canonical Conceptual Graphs. In J.F. Sowa G.W. Mineau, B. Moulin, editor, *Lecture Notes in Artificial Intelligence, 699, Proceedings ICCS'93*, pages 294–311. Springer Verlag, 1993.

[MC95] M.L. Mugnier and M. Chein. Représenter des connaissances et raisonner avec des graphes. Research Report 95-003, LIRMM, Jan. 1995. 52 pages. To appear in Revue d'Intelligence Artificielle.

[RF87] A.S. Rao and N. Foo. Conceptual Graph Reasoning System. In *Proceedings of the 3rd Conference on Artificial Intelligence Applications*, pages 87–92. IEEE Computer Society Press, 1987.

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

This article was processed using the LaTeX macro package with LLNCS style