Reasoning with type definitions

M. Leclère

IRIN – IUT de Nantes
2, rue de la Houssinière - BP 92208
44 322 Nantes cedex 03 - France
Ph.: +33 2 40 37 49 01 Fax: +33 2 40 37 49 70
E-mail: leclere@irin.univ-nantes.fr

Abstract. This article presents an extension of the basic model of conceptual graphs : the introduction of type definitions. We choose to consider definitions as sufficient and necessary conditions to belong to a type. We extend the specialization/generalization relation on conceptual graphs to take advantage of these definitions. Type contractions and type expansions are clearly defined. We establish the correspondence with projection by use of the atomic form of conceptual graphs. Finally, we give a logical interpretation of type definitions and prove than the correspondence between logical deduction and generalization relation is maintained.

Keywords: type definitions, contraction, expansion, atomic form, projection, logical interpretation.

1 Introduction

The aim of knowledge representation is to provide formal model which allow to modelize different kinds of knowledge and allow to reason with this knowledge. In conceptual graphs (CGs), the knowledge is split into several levels. The terminological level defines the conceptual vocabulary. It contains the concept type lattice and the relation type poset. These two taxonomies are simply composed of strings, representing concept types or relation types, which are ordered by a specific/generic relation. At the assertional level, one describes some facts by CGs constructed with the conceptual vocabulary.

However, we often dispose of general knowledge which is relevant to terminological level but one can't represent it with a simple specific/generic relation. J. Sowa proposes the formalism of abstractions to answer this need. In this paper, we study a kind of complex terminological knowledge: the type definition.

We start from the basic model of CGs, introduced by J. Sowa in [Sow84] and clarified by M. Chein and M.L. Mugnier in [CM92, MC96], and extend it to take into account type definitions when we are reasoning at assertional level. This extension includes contraction and expansion operations in the specialization rules. The connection with the projection is established once again. After giving a logical interpretation for definition, one demonstrates that the specialization relation always corresponds to a complete and sound set of inference rules.

In section 2, we present the different semantics that we can give to type definitions and we remind the type definition syntax. Then we choose to consider a definition as a set of sufficient and necessary conditions for belonging to a type. Section 3 introduces the four new specialization rules allowing to manipulate the type definitions at assertional level: contraction and expansion of concept types or relation types. Then we show in section 4 how projection allows to compute the new specialization relation. The section 5 is devoted to logical interpretation of type definitions and contraction/expansion rules.

2 Type definition formalism

In the basic model of CGs, the meaning of a type is given by position which hold into the taxonomy of types. The specific/generic relation is thus the only definitional mechanism. Such a representation of meaning for concept types and concept relations is very poor. Often one has some generic information on types which are not expressible by this single mechanism. The mechanism of type definition address this problem by associating a formal description to a type. This description is constructed with atomic types (a type without description) and already defined types. The semantics given to the type/description association determine the reasoning technics to use with this form of knowledge representation.

In this work, we consider that descriptions represent a set of characteristics, properties, attributes which can be shared by objects of the domain of representation. A description is thus the intensional representation of a set of objects. It remains to define the semantics given to the link between a type and its description. In general, two semantics are used:

- the description represents a set of sufficient and necessary conditions to belong to its type. Any object recognized by the description must belong to the type and any instance of type owns the attributes of the description. The description is thus the intensional representation of type. These semantics are given to defined concepts in the KL-ONE derived systems [WS92]. Such semantics are called *definition*;
- the description only represents a set of necessary conditions to belong to its type. These are semantics that is used for natural kinds. In KL-ONE systems, they are given to primitive concepts. These semantics are sometimes called *partial definition*.

In CGs, the descriptions are represented by abstractions (cf. [Sow84] section 3.6). They are conceptual graphs which one or several generic concepts are considered as formal parameters. In the following, we study description mechanism to do some definitions and not to do partial definitions.

Definition 1 *A concept type definition asserts an equivalence between a concept type and a monadic abstraction. We denote* $t_c(x) \stackrel{def}{\Leftrightarrow} D(x)$ *the definition of type* t_c *with* x *the variable of formal parameter. The formal parameter concept vertex*

of $D(x)$ is called the **head** *of t_c. In the aristotelician approach, the genus of new type is the type of the head concept and $D(x)$ represents the differentiae from t_c to its genus.*

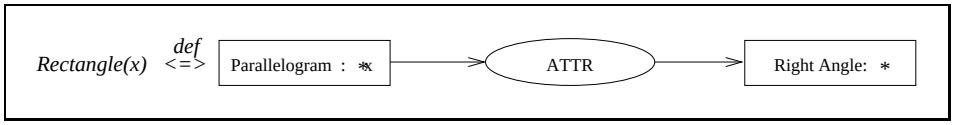

Fig. 1. Definition of a rectangle as a parallelogram with right angles..

Definition 2 *A relation type definition asserts an equivalence between a relation type and a n-ary abstraction. We denote $t_r(x_1, x_2...x_n) \overset{def}{\Leftrightarrow} D(x_1, x_2...x_n)$ the definition of type t_r with $x_1, x_2...x_n$ the variables of formal parameters. The formal parameter concept vertices of $D(x_1, x_2...x_n)$ are called the* **arguments** *of t_r.*

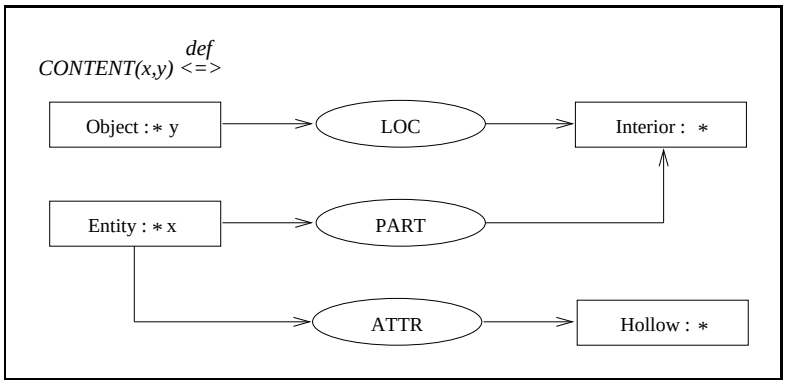

Fig. 2. Definition of content relation as a link between an object and an entity which contains the object.

Restriction: In the following, we assume there is neither direct nor undirect recursive type definition.

3 Extension of the model

The mechanism of type definition grows expressivity of the CG model. It remains to extend the specialization/generalization relation to take advantage of these definitions. We believe that in order to keep the specificity of the CG model,

such an extension must keep the triple correspondence: specialization rules / projection / logical inference. We need to modify some parts of CG model to perform that.

3.1 The canon

The canon contains the conceptual vocabulary of the domain of representation. It represents the terminological component of the CG model. The canon is shared in two taxonomies: T_c the concept type taxonomy and T_r the relation type taxonomy.

These two type posets are composed of atomic types or defined types. A type definition is attached to each defined type. For atomic types, the order relations $\leq_c$ and $\leq_r$ are simply given by the user (who is supposed to be a specialist of the domain of representation).

The concept type definitions extend the $\leq_c$ relation. Since a concept type is introduced by definition as subtype of its genus, the $\leq_c$ relation is modified:

Definition 3 *Let t_c be a defined concept type, we have $t_c \leq_c genus(t_c)$.*

The genus of a defined concept type may be atomic or defined. If the genus is defined then, by transitivity of t_c, the new type is also a subtype of the genus of its genus. As no recursive definition is permitted, one can define the smallest atomic super-type of a defined concept type that we call *atomic genus*.

Definition 4 *Let t_c be a defined concept type, the atomic genus of t_c denoted by $AG(t_c)$ is the atomic type obtained from t_c by successive applications of the genus function.*

We immediately get the following property:

Property 1 *Let t_c be a defined concept type and t'_c an atomic concept type, $t_c \leq_c t'_c$ if and only if $AG(t_c) \leq_c t'_c$.*

On the contrary, the relation type definitions does not extend the $\leq_r$ relation. This is because such a kind of definition does not derive from Aristotle's method. Each relation type owns a signature which fixes the arity and the concept authorized as maximal types for arguments of a relation of this type. The signature of atomic relation types is given by the user. For defined relation types, the signature is deduced from type definition in the following way.

Definition 5 *Let t_r be a defined relation type:*

- *the arity of t_r is equal to the arity of the abstraction which defines t_r;*
- *the concept maximal type of the i-th argument of t_r is the type of the i-th formal parameter of the abstraction.*

A conceptual graph must verify the canonicity property and the conformity property. The first property enforces the neighbour concepts of a relation vertex to accord with the signature of the relation type. The second property enforces the concepts to have a referent according with the type.

3.2 Extension of specialization relation

A specialization relation is defined on CGs by a set of specialization rules. The reverse relation, the generalization, corresponds to logical deduction. First part of this theorem (the soundness of the CG model) has been demonstrated by J. Sowa [Sow84]. The reciprocal part (the completeness) has been demonstrated by M. Chein and M.L. Mugnier [CM92, MC96] with CGs in *normal form* (normal form does not allow CGs to have several individual concepts with the same referent).

Using type definitions allows to represent knowledge to different levels of abstraction. But generalization relation forgets some inferences. In figure 3 for instance, there is no possibility to derive H from G (or G from H) though these two CGs represent the same knowledge (according to the type definition of figure 1).

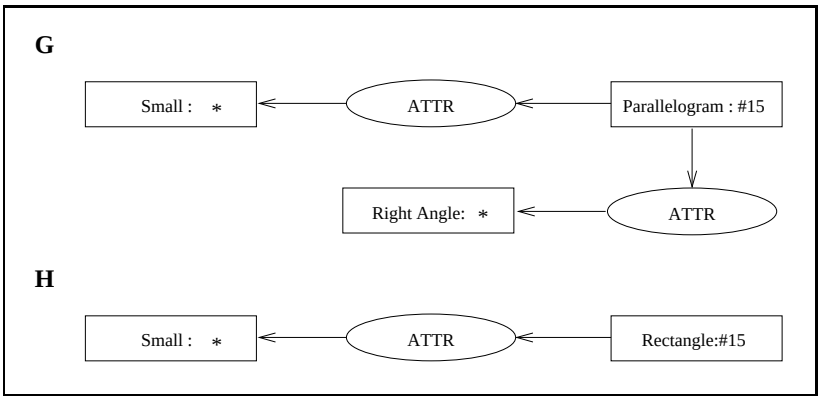

Fig. 3. Two equivalent graphs according to the type definition of Rectangle.

In order to preserve the truth of the generalization relation, we must add four new specialization rules: type contractions and type expansions. These operations are close to J. Sowa's operations [Sow84] but not identical. First we introduce these operations in order to extend the formal model of CGs when J. Sowa presents these operations as tools to simplify CGs. So we search to give a well-defined framework to manipulate type definitions rather than an algorithm of graph reduction. Secondly, J. Sowa's contraction (cf. section 3.6.5 of [Sow84]) does not compute an equivalent graph but a more general one than the original (e.g. some informations can be lost). As for his expansions, either it does not preserve truth, or the defined type is not replaced by its genus.

We state that our following operations preserve truth. We define them as reversible operations which replace a defined type with its definition and conversely replace a graph corresponding to a type definition with a defined type.

Definition 6 (concept type expansion) *Let G be a conceptual graph containing a concept c with a defined type t_c. Let $t_c(x) \stackrel{def}{\Leftrightarrow} D(x)$ be the definition. H is obtained by expansion in the following way:*

1. *replace t_c the type of c by its genus;*
2. *restrict the head of t_c by changing its referent to the referent of c;*
3. *join G and D on c and the head of t_c;*

This operation is permitted only if the resulting graph is canonical (e.g. c does not violate signature of neighbouring relation types).

Definition 7 (relation type expansion) *Let G be a conceptual graph containing a relation r with a defined type t_r. Let $t_r(x_1, ...x_n) \stackrel{def}{\Leftrightarrow} D(x_1, ...x_n)$ be the definition. H is obtained by expansion in the following way:*

1. *restrict each argument c_i of t_r by changing its type and referent to type and referent of the i-th neighbour of r;*
2. *perform n joins to identify the i-th neighbouring concept of r with the i-th argument of t_r;*
3. *delete r and all its edges from G.*

Definition 8 (concept type contraction) *Let G be a CG containing a subgraph G' isomorphic to the definition $D(x)$[1] of a defined concept type t_c (concept c in G' corresponding to formal parameter of $D(x)$ in isomorphism can have an individual referent yet). If moreover c is a cut point between G' and the other part of G (i.e. all the paths from one vertex of G' to one vertex of the rest of G go through the vertex c), H is obtained by contraction in the following way:*

1. *replace the type of c with t_c;*
2. *delete subgraph $G' \setminus \{c\}$ from G.*

This operation is permitted only if c in the resulting graph is conform.

Definition 9 (relation type contraction) *Let G be a CG containing a subgraph G' isomorphic to the definition $D(x_1, ...x_n)$ of a defined relation type t_r (concepts c_i ($i \in [1..n]$) in G' corresponding to formal parameters of $D(x_1, ...x_n)$ can have yet individual referents and types lesser than types of corresponding arguments of t_r). If moreover the concepts c_i ($i \in [1..n]$) allow to disconnect G' from G, H is obtained by contraction in the following way:*

1. *insert a relation having type t_r to G;*
2. *$\forall i \in [1..n]$ link the i-th edge of r to concept c_i of G';*
3. *delete subgraph $G' \setminus \{c_1, ...c_n\}$ from G.*

[1] The variable x in the formal parameter concept is though not considered for isomorphism.

Observe that concept type expansion and concept type contraction are not necessarily permissible due to the respect constraint of canonicity and conformity. There is no problem with relation contraction and relation expansion.

Let us recall the basic specialization rules before defining the specialization relation on CG model extended to type definitions. We consider the following four rules:

- concept restriction which allows to replace a type with a subtype or generic referent with an individual one (conformity must be preserved);
- relation restriction which allows to replace a type with a subtype (canonicity must be preserved);
- join which allows to merge two identical concepts (perform from two graphs or only one);
- simplification which allows to delete a duplicated relation.

Definition 10 *H is a specialization of G, written $H \leq_D G$, if and only if one can derive H from G using both the basic specialization rules and the new specialization rules. Respectively G is thus a generalization of H.*

Note that the new specialization relation produces equivalent classes greater than the equivalent classes of the basic specialization relation. This is due to the semantics of definition which enforce to consider the resulting graph of a contraction (or expansion) operation equivalent to the original graph.

4 Computing the specialization relation

In the basic model, the projection operator is used to detect the specialization relation between CGs. In our extended model, projection can still exhibit the specialization relation but it is not a complete operator to detect the specialization relation between CGs. For instance, there is no projection between conceptual graphs in figure 3 though there exists a mutual specialization relation between these two graphs.

Using projection operator to detect again specialization relation needs to define a transformation on conceptual graphs on which one computes projection. For that, we reuse the *atomic form* introduced in [CL94]. Indeed, we extend this form to take into account the defined relation types.

Definition 11 *Let G be a CG, we call atomic form of G, written $AF(G)$, the graph without defined types obtained by a succession of type expansions.*

Property 2 *Without recursive type definitions, one can compute the atomic form of a CG in a finite number of type expansions.*

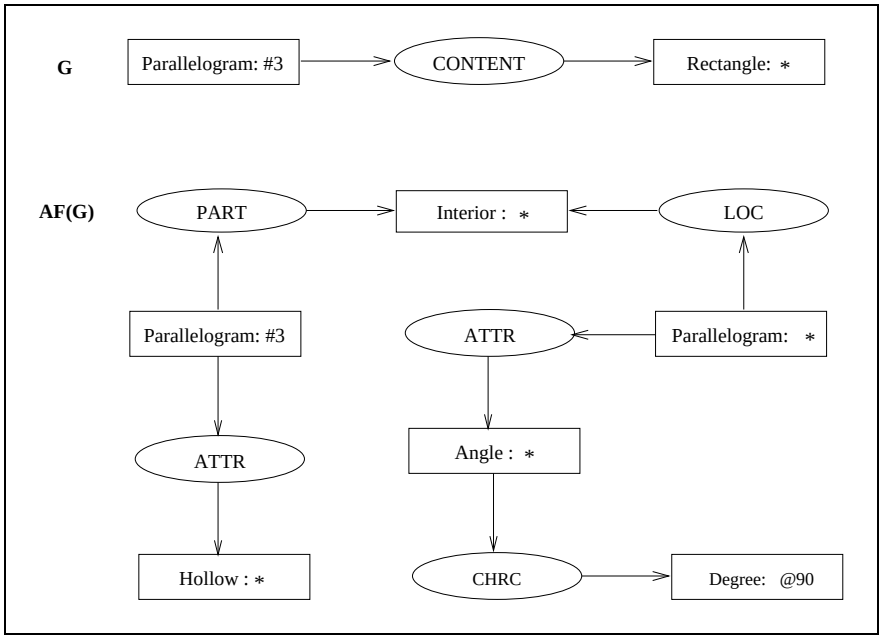

Fig. 4. G and its atomic form according to Rectangle definition, CONTENT definition and supposing that Right Angle is defined as an Angle of 90 degrees.

Proof

Let G be a CG containing vertices with defined types. As no constraint exists to perform relation type expansions, one can compute from G a graph G' without any relation having a defined type (we have just to perform relation type expansions while it remains relations with defined types). Since relation type definitions are not recursive, one can do it in a finite number of relation type expansions.
Let's prove that we can now perform concept type expansion from any concept c of G' having a defined type (e.g. the resulting graph is canonical). Since G' is canonical, any neighbouring relation r of c is such as: $type(c) \leq_c Sig_i(type(r))$ with c being the i-th neighbour of r and Sig_i representing the maximal concept type of the i-th argument of a relation. Moreover, the type of r is atomic according to the construction of G'. With property 1, we conclude that $AG(type(c)) \leq_c Sig_i(type(r))$. Thus $genre(type(c)) \leq_c Sig_i(type(r))$ and so the resulting graph will be canonical. Consequently, we can perform the concept type expansions from all the concepts with defined types of G'. Let G'' the resulting graph. The concept type definitions may contain vertices with defined types, we must redo the two preceding phases. But these definitions being no recursive, we will obtain a graph without defined types in a finite number of repetitions. □

Theorem 1. *The atomic form of a CG is unique.*

Proof

We must demonstrate that whatever succession order chosen to perform the type expansions, we obtain isomorphic graphs. We use the proof schemes proposed by G. Chaty and M. Chein in [CC81] for studying the graph reductions. These proof schemes based on the Church-Rosser's systems take advantage of properties on the confluent reductions demonstrated by G. Huet [Hue80].
Firstly, we define the $\mathcal{R}$ relation on set of the isomorphic CG classes by: $C_g \mathcal{R} C_h$ *iff* $\exists G \in C_g$ *and* $\exists H \in C_h$ *such as* H *is obtained from* G *by a type expansion.*
Secondly, we demonstrate $\mathcal{R}$ is locally confluent that is $\forall C_g, C_{g_1}, C_{g_2} \in \mathcal{G}$, *if* $C_g \mathcal{R} C_{g_1}$ *and* $C_g \mathcal{R} C_{g_2}$ *then* $\exists C_h \in \mathcal{G}$ *such as* $C_{g_1} \mathcal{R}^* C_h$ *and* $C_{g_2} \mathcal{R}^* C_h$ *with* $\mathcal{R}^*$ *is the transitive closure of* $\mathcal{R}$.
With the different properties on confluent reductions, we conclude that atomic form of a graph is unique. A detailed proof is given in [Lec95]. □

Atomic form of CG being obtained by type expansions, we immediately have the following property:

Property 3 *Atomic form of a graph is equivalent to the original graph according to the specialization relation. We thus have* $AF(G) \leq_D G$ *and* $G \leq_D AF(G)$.

In order to establish the link between projection operator and specialization relation in the extended model, we first demonstrate that two CGs belong to specialization extended relation if and only if their atomic forms belong to specialization basic relation. In the following, specialization basic relation is written $\leq$ while the specialization extended relation is written $\leq_D$.

Theorem 2. *If* $AF(H) \leq AF(G)$ *then* $H \leq_D G$.

Proof

Since $AF(H) \leq AF(G)$ then we have $AF(H) \leq_D AF(G)$ because $\leq_D$ is defined as an extension from $\leq$. With the property 3 and considering the transitivity of $\leq_D$ we immediately conclude that $H \leq_D G$. □

In order to demonstrate the reciprocal lemma, we show that to each specialization rule allowing to derive H from G in the extended model, there corresponds a sequence of specialization elementary rules allowing to derive AF(H) from AF(G) (a detailed correspondence is established in [Lec96]):

- for a join on concepts with atomic types there corresponds a similar join on atomic forms;
- for a join on concepts with defined types there corresponds a sequence of joins and simplifications;
- for a simplification with relations having atomic types there corresponds a similar simplification;

- for a simplification with relations having defined types there corresponds a sequence of joins and simplifications;
- for a relation restriction which necessarily replaces an atomic type with an atomic subtype there corresponds a similar restriction;
- for a concept restriction which replaces a generic referent with an individual referent there corresponds a similar restriction;
- for a concept restriction which replaces an atomic type with an atomic subtype there corresponds a similar restriction;
- for a concept restriction which replaces an atomic type with a defined subtype there corresponds a concept restriction and an external join (to connect the atomic form of the subtype definition);
- for a concept restriction which replaces a defined type with a defined subtype there corresponds an external join and a sequence of internal joins and simplifications (to merge the two type definitions);
- for a contraction or expansion there corresponds the empty sequence of specialization rules on atomic forms.

Theorem 3. *If $H \leq_D G$ then $AF(H) \leq AF(G)$.*

Proof

For each rules of specialization extended relation allowing to derive H from G, we have exhibited a sequence of rules of specialization basic relation allowing to derive $AF(H)$ from $AF(G)$. Since $H \leq_D G$, we obtain, by composition of specialization rules, $AF(H) \leq AF(G)$. □

Consequently, any sequence of specialization rules allowing to derive H from G in the extended model can be reorganized in:

1. a sequence of expansions to derive $AF(G)$ from G;
2. a sequence of specialization elementary rules to derive $AF(H)$ from $AF(G)$;
3. a sequence of contractions to derive H from $AF(H)$.

Let us recall the complete correspondence between projection and specialization basic relation established by M. Chein and M.L. Mugnier in [CM92]:

Theorem 4. *$H \leq G$ iff there exists a projection from G to H.*

The three previous theorems yield:

Theorem 5. *$H \leq_D G$ if and only if there exists a projection from $AF(G)$ to $AF(H)$.*

5 Logical interpretation

We extend the operator ϕ to take into account type definitions in the logical interpretation of CG model. A type definition represents an equivalence between

a type and its description, we must preserve these semantics in logical interpretation. Let $t(x_1, ...x_n) \stackrel{def}{\Leftrightarrow} D(x_1, ...x_n)$ be a relation type definition or a concept type definition (in this case $n = 1$). The logical formula associated with this definition, written $\phi(Def(t))$, is determined by the following construction:

- with the type t, associate a n-adic predicate having the same name;
- with the left hand of the definition, associate the atomic formula $t(x_1, ...x_n)$;
- with the right hand of the definition, associate the formula $\phi(D(x_1, ...x_n))$ computed in the same way as for a normal CG but without existential closure of variables $x_1, ...x_n$;
- connect the two previous formula with an equivalence logical operator;
- universally quantify the variables $x_1, ...x_n$ of the constructed formula.

For instance, the formulae associated with type definitions in figures 1 and 2 are:

$\forall x(Rectangle(x) \leftrightarrow \exists y(Parallelogram(x) \wedge RightAngle(y) \wedge Attr(x, y)))$

$\forall x \forall y(Content(x, y) \leftrightarrow \exists w \exists z(Object(y) \wedge Interior(w) \wedge Entity(x) \wedge Hollow(z)$
$\wedge Loc(y, w) \wedge Part(x, w) \wedge Attr(x, z)))$

With this extension of ϕ, we can show that type expansions and type contractions preserve the semantics of the formulae associated with CGs before and after the operation.

Property 4 *Let H be a CG derived from G by a type expansion (or G derived from H by a type contraction) of a defined type t, we have $\phi(Def(t)) \vdash \phi(G) \leftrightarrow \phi(H)$.*

Proof

Suppose that H is obtained from G by a concept type expansion on c. Let $t(x) \stackrel{def}{\Leftrightarrow} D(x)$ be the type definition of the type of c. We have $\phi(Def(t)) = \forall x(t(x) \leftrightarrow F_D[x])$ ($F_D[x]$ is the formula associated with $D(x)$).
If c is a generic concept then $\phi(G) = EC(t(x) \wedge A[x])$ and $\phi(H) = EC(F_D[x] \wedge A[x])$ (EC represents the existential closure). Furthermore $A[x]$ and $F_D[x]$ do not contain any common variable except x. Since $\phi(Def(t))$ is valid, we also have $\phi(G) \leftrightarrow \phi(H)$ is valid.
If c owns an individual referent m then $\phi(G) = t(m) \wedge EC(A)$ and $\phi(H) = EC(F_D[m] \wedge A)$ with $EC(F_D[m]) = Subst_{[x/m]}(EC(F_D[x]))$. Furthermore $A[x]$ and $F_D[m]$ do not contain any common variable. Since $\phi(Déf(t))$ is valid, we also have $\phi(G) \leftrightarrow \phi(H)$.

Suppose that H is obtained from G by a relation type expansion on r. Let $t(x_1, ...x_n) \stackrel{def}{\Leftrightarrow} D(x_1, ...x_n)$ be the type definition of type of r. We have $\phi(Def(t)) = \forall x_1 ... \forall x_n(t(x_1, ...x_n) \leftrightarrow F_D[x_1, ...x_n])$ ($F_D[x_1, ...x_n]$ is the formula associated with $D(x_1, ...x_n)$).

$\phi(G) = EC(t(t_1, ...t_n) \wedge A[t_1, ...t_n])$ and $\phi(H) = EC(F_D[t_1, ...t_n] \wedge A[t_1, ...t_n])$ and $t_1, ...t_n$ are constants or variables according to referents of neighbouring concepts of r.
$EC(F_D[t_1, ...t_n]) = Subst_{[x_1/t_1, ...x_n/t_n]}(EC(F_D[x_1, ...x_n]))$. Furthermore, $A[t_1, ...t_n]$ and $F_D[t_1, ...t_n]$ do not contain any common variable except possible variables in $t_1, ...t_n$. Since $\phi(Def(t))$ is valid, we also have $\phi(G) \leftrightarrow \phi(H)$ is valid. □

We denote $\phi(\mathcal{S})$ the set of formulae associated with the relations $\leq_c$ and $\leq_r$ defined on the atomic type sets. Let us recall that with each pair (concept or relation types) such as $t \leq t'$ is associated the formula $\forall x_1 ... \forall x_n (t(x_1, ...x_n) \rightarrow t'(x_1, ...x_n))$. We denote $\phi(\mathcal{D})$ the set of formulae associated with type definition set. We now demonstrate that generalization rules (the reverse specialization rules) still corresponds to a set of inference rules.

Theorem 6. *If $G \leq_D H$ then $\phi(\mathcal{S}), \phi(\mathcal{D}), \phi(G) \vdash \phi(H)$.*

Proof

Using $\phi(\mathcal{S})$, M. Chein and M.L. Mugnier demonstrated that elementary rules of generalization correspond to inference rules [CM92]. Using $\phi(\mathcal{D})$, I have demonstrated that type contractions and type expansions correspond to equivalence rules (cf. previous property). By composition of different rules, we obtain $\phi(G) \rightarrow \phi(H)$ is valid. □

Moreover, we have immediately:

Property 5 *Let G be a CG, $\phi(\mathcal{D}) \vdash \phi(G) \leftrightarrow \phi(AF(G))$.*

The normal form restriction is not still sufficient to demonstrate the completeness of the CG model. Suppose that a normal form graph G contains a concept with the individual referent m and another concept with a defined type t. And suppose that the type definition of t contains a concept with the same individual referent m. Then the atomic form of G will not be in a normal form and so the specialization relation can miss some deductions. For instance, the graph G in figure 5 can not be derived from H but the logical formula associated with G implies the logical formula associated with H.

Theorem 7. *Let G and H be two CGs with atomic forms in normal form. $\phi(\mathcal{S}), \phi(\mathcal{D}), \phi(G) \vdash \phi(H)$ if and only if $G \leq_D H$.*

Proof

Since $\phi(\mathcal{S}), \phi(\mathcal{D}), \phi(G) \vdash \phi(H)$, property 5 entails $\phi(\mathcal{S}), \phi(\mathcal{D}), \phi(AF(G)) \vdash \phi(AF(H))$, and then the set $\{\phi(\mathcal{S}), \phi(\mathcal{D}), \phi(AF(G)), \neg\phi(AF(H))\}$ is inconsistent. The resolution method allows to derive the empty clause from clauses generated by this set.

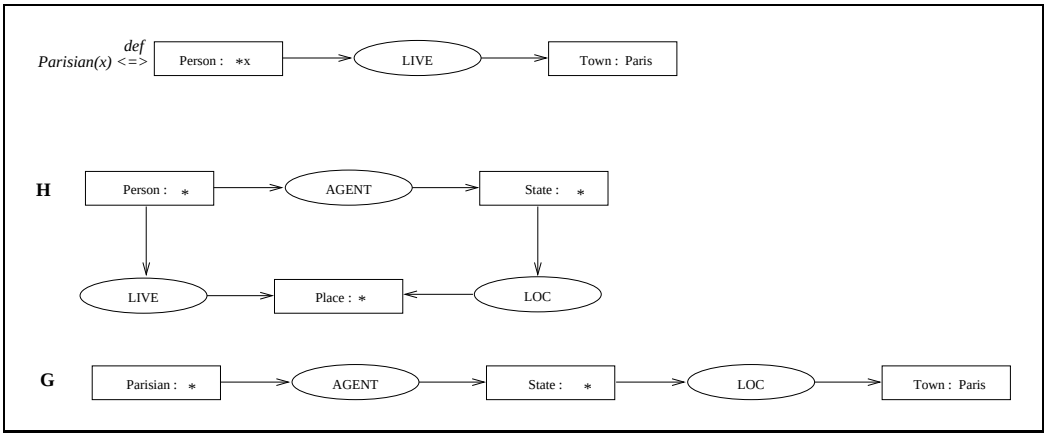

Fig. 5. An example with normal form graphs such as $G \not\leq_D H$ while $\phi(\mathcal{S}), \phi(\mathcal{D}), \phi(G) \vdash \phi(H)$.

Each formula in $\phi(\mathcal{S})$ is of type $\forall x_1 ... \forall x_n (t(x_1, ...x_n) \rightarrow t'(x_1, ...x_n))$ and generates a clause of type: $\neg t(x_1, ...x_n) \vee t'(x_1, ...x_n)$.

Each formula in $\phi(\mathcal{D})$ is of type:
$\forall x_1 ... \forall x_n (t_d(x_1, ...x_n) \leftrightarrow \exists y_1 ... \exists y_m (P_1[x_1, ...x_n, y_1, ...y_m] \wedge ... P_k[x_1, ...x_n, y_1, ...y_m]))$
with t_d the defined type predicate and P_i ($i \in [1..k]$) predicates associated with other vertices of the definition. The argument terms are either defined type variables $x_1, ...x_n$, or variables $y_1, ...y_m$ and constants associated with other concepts. Each formula generates k clauses of type:

$$\neg t_d(x_1, ...x_n) \vee P_i[x_1, ...x_n, f_1(x_1, ...x_n), ...f_m(x_1, ...x_n)]$$

(f_i functions result from the skolemisation of variables y_i), and a clause of type: $t_d(x_1, ...x_n) \vee \neg P_1[x_1, ...x_n, y_1, ...y_m] \vee ... \neg P_k[x_1, ...x_n, y_1, ...y_m]$.

The formula $\phi(AF(G))$ is of type: $\exists x_1 ... \exists x_n (Q_1[x_1, ...x_n] \wedge ... Q_m[x_1, ...x_n])$ with Q_i ($i \in [1..m]$) predicates associated with vertices of $AF(G)$. It generates m clauses of type: $Q_i[a_1, ...a_n]$ (with a_i the Skolem constants).

The formula $\neg\phi(AF(H))$ is of type: $\neg \exists x_1 ... \exists x_n (R_1[x_1, ...x_n] \wedge ... R_l[x_1, ...x_n])$ with R_i ($i \in [1..l]$) predicates associated with vertices of $AF(H)$. It generates a clause of type: $\neg R_1[x_1, ...x_n] \vee ... \neg R_l[x_1, ...x_n]$.

The empty clause is necessarily produced in emptying the clause associated with $\neg\phi(AF(H))$. The clauses generated by the type definitions are needless in the resolution process. Using these clauses will introduce some new predicates (the predicate associated with defined type). Since to delete these predicates one should have to re-introduce previous deleted predicates. Thus the empty clause can be produced from the other clauses and thus

$\{\phi(\mathcal{S}), \phi(AF(G)), \neg\phi(AF(H))\}$ is inconsistent.

Then the completeness of the basic model yields $AF(G) \leq AF(H)$ and with the theorem 2 we can conclude $G \leq_D H$. The reciprocal is immediate with theorem 6. □

6 Conclusion

We have defined an extension of the CG model allowing to enrich the terminological knowledge representation. This extension clarify the type definition mechanism proposed by J. Sowa in stating precisely the semantics of definitions and providing a framework to exploit type definitions. We have introduced type contractions and type expansions as new specialization rules and consequently extend the specialization relation. We have showed the correspondence between the new specialization relation and projection operator on atomic forms of CGs. Finally we have demonstrated the soundness and completeness of the extended model.

This extension allows to define a type classifier in a KL-ONE-style [Lip82] which was introduced in [CL94]. We are currently working to take into account partial definitions in order to provide terminological knowledge representations for natural types.

This theoretical work must be completed by a study of good algorithms for computing specialization relation. Totally compute the atomic forms to test projection does not seem to be the best solution !

References

[CC81] G. Chaty and M. Chein. Réduction de graphes et systèmes de Church-Rosser. *R.A.I.R.O.*, 15(2):109–117, 1981.

[CL94] M. Chein and M. Leclère. A cooperative program for the construction of a concept type lattice. In *Supplement Proceedings of the Second International Conference on Conceptual Structures*, pages 16–30, College Park, Maryland, USA, 1994.

[CM92] M. Chein and M.L. Mugnier. Conceptual graphs : fundamental notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.

[Hue80] G. Huet. Confluent reductions : Abstract properties and applications to term rewriting systems. *J.A.C.M.*, 27:797–821, 1980.

[Lec95] M. Leclère. *Le niveau terminologique du modèle des graphes conceptuels : construction et exploitation.* PhD thesis, Université Montpellier 2, 1995.

[Lec96] M. Leclère. Raisonner avec des définitions de types dans le modèle des graphes conceptuels. Rapport de recherche 143, IRIN, Nantes, France, Novembre 1996.

[Lip82] T. Lipkis. A KL-ONE classifier. In J.G. Schmolze and R.J. Brachman, editors, *Proceedings of the 1981 KL-ONE Workshop*, pages 126–143, Jackson, New Hampshire, 1982. The proceedings have been published as BBN Report No. 4842 and Fairchild Technical Report No. 618.

[MC96] M.-L. Mugnier and M. Chein. Représenter des connaissances et raisonner avec des graphes. *Revue d'Intelligence Artificielle*, 10(1):7–56, 1996.

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

[WS92] W.A. Woods and J.G. Schmolze. The kl-one family. In F. Lehmann, editor, *Semantic Networks in Artificial Intelligence*, pages 133–177, Pergamon Press, Oxford, 1992.

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