Introduction of Viewpoints in Conceptual Graph Formalism

Myriam Ribière, Rose Dieng
INRIA (project ACACIA)
2004, route des Lucioles BP.93
06902 Sophia-Antipolis Cedex
E-mail: {Myriam.Ribiere,Rose.Dieng}@sophia.inria.fr

Abstract:
To represent knowledge in a context of several experts, it is interesting to model the multiple perspectives that different experts may have on the objects handled in their reasoning. The concept of a particular perspective called viewpoint, is already used in the field of object oriented representation or in the field of databases conception. Taking inspiration of research made in object oriented representation, we propose in this paper an extension of the conceptual graph formalism to integrate viewpoints in the support and in the building of conceptual graphs. We also study mechanisms for managing viewpoints.

Keywords: *Knowledge Representation*, Conceptual Graphs, Viewpoints.

1. Introduction

In knowledge acquisition and modelling, it is often to encounter multi-expertise problem. Indeed, each expert, involved in knowledge acquisition, can have a particular perspective on a handled object, depending on his/her position in the enterprise organization, domain (working context) and the knowledge learned from experience (know-how, competence). Therefore it is difficult to elaborate a unique model taking into account the different terminologies used by the different experts. The notion of viewpoints on the handled object, helps to model knowledge by factorizing information. Viewpoints are also efficient for knowledge extraction; there are viewpoint-based mechanisms allowing accessibility to a subset of information of a common model.

Viewpoints for knowledge acquisition are particularly used in the field of object oriented representation. The interest is based on manipulation of a complex system in terms of accessibility and dynamic knowledge representation. After describing previous work in object oriented representation, we propose an extension of the conceptual graph formalism to support the viewpoint management: we first describe the different steps to integrate viewpoints in support and in conceptual graph, then we explain how to manage viewpoints.

In conclusion, we describe our perspectives and we discuss on the interest of our work in comparison with previous work related to views or viewpoints in conceptual graph formalism.

2. Viewpoint Notion

Research on viewpoint notion was carried out in the seventies. Several interpretations of this notion are possible. The first one is a spatial interpretation [9]: viewpoints correspond to the different perceptions of an object with respect to the observer's position. The second interpretation is a knowledge domain one: viewpoints correspond to the different ways to translate knowledge with respect to the social position, know-how and competence of an expert. In this interpretation, a viewpoint includes a context, and the perception of a person or a group of persons.
This second interpretation underlies several systems developed since 1977. The first one, KRL [1] is dedicated to knowledge representation. LOOPS [12] inherits the approach of KRL. SHOOD [10], ROME [2], TROPES [7] are the second generation of those systems. The definition of the viewpoint interpretation is more restricted in those systems: they consider the different ways by which an expert can see a knowledge base, but they don't explain explicitly what is a viewpoint. In most applications with such systems, viewpoints are restricted to a knowledge domain or a competence domain.

We describe here two example models for the introduction of viewpoint in knowledge representation and for the management of multiple viewpoints. We focus on these models because they are sure as basis of our reflection about introducing viewpoints in conceptual graphs.

2.1 TROPES

TROPES [7] notion of multiple perspectives allows to structure a knowledge base according to different objectives. TROPES is based on the definition of a set of basic concepts, which can be described in a class arborescence [fig. 1], according to different viewpoints.

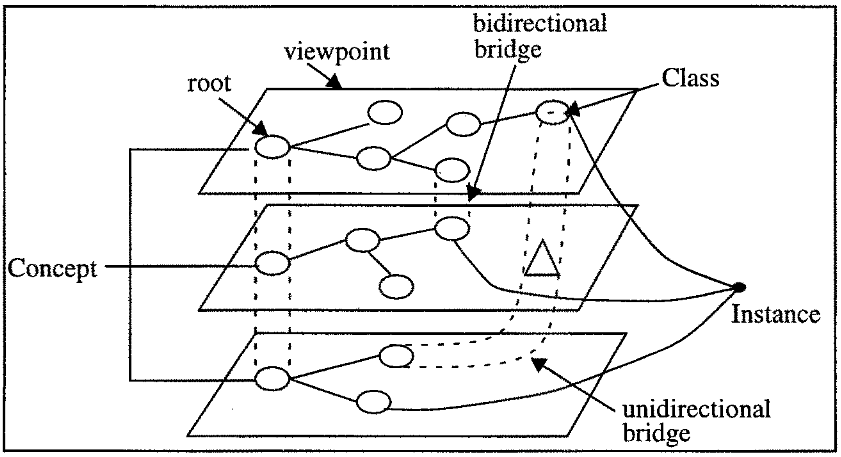


Fig. 1. The TROPES model

Bridges in the structure allow to build an inclusion or equivalence relation between two classes pertaining to different viewpoints. Each class belongs to only one viewpoint. The concept of multi-instantiation enables to instantiate an object with several classes.
This structuration is dedicated to the classification process allowing the manipulation of complex systems and, in particular, an easier access to information.

2.2 ROME

ROME [2] and FROME [3] (the extension of ROME for frames) are dedicated to modelling incomplete and evolving knowledge, in particular through the multiple representation according to multiple viewpoints. FROME uses a simple link for instantiation, but multiple links for representation. So an object [fig. 2] is identified by a main class, which keeps the identity of the object, and can have multiple representation links, which specialize the object through viewpoints.

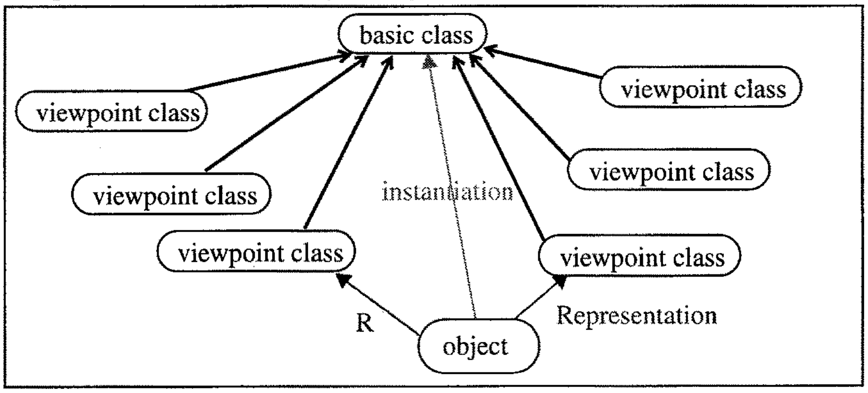


Fig. 2. The ROME model

This technique allows to add information by adding a new viewpoint to an object (by a new link of representation). FROME also uses an exclusion link to prevent an object from having incompatible representation links. In TROPES this exclusion link does not exist, since an object can be instantiated only by one class per viewpoint.

2.3 Objects and Conceptual Graphs

A natural correspondence can be made between formalisms for object oriented representation on the one hand and conceptual graph formalism on the other hand. Indeed, object oriented formalisms often rely on the notions of class and object (i.e. instance). A class is described by slots and methods, that may be inherited through a hierarchy of classes. An object is an instance of a class and can be manipulated in a knowledge base. So a class corresponds to a concept type (in CG formalism), and an object to a concept, having a concept type and a referent. The class hierarchy corresponds to the concept type lattice in the support.
Furthermore previous work already exists on bringing together oriented object language and conceptual graph formalism [5]. Therefore this correspondence convinced us that the integration of viewpoints in conceptual graph formalism can be inspired by research realized in object oriented representation.

3. Viewpoint Integration in Conceptual Graph

In object oriented formalisms, the integration process of viewpoints can be summed up by two phases:

- How to represent viewpoints inside the class hierarchy?
- How to identify an object with different viewpoints?

These phases are situated at two different levels: the first one concerns the class level, which corresponds to the support level in CG formalism. The second one deals with the instance level, which corresponds to the graph level in CG formalism. The figure 3 summarizes our approach. It shows the different entities that take place in the viewpoint integration.

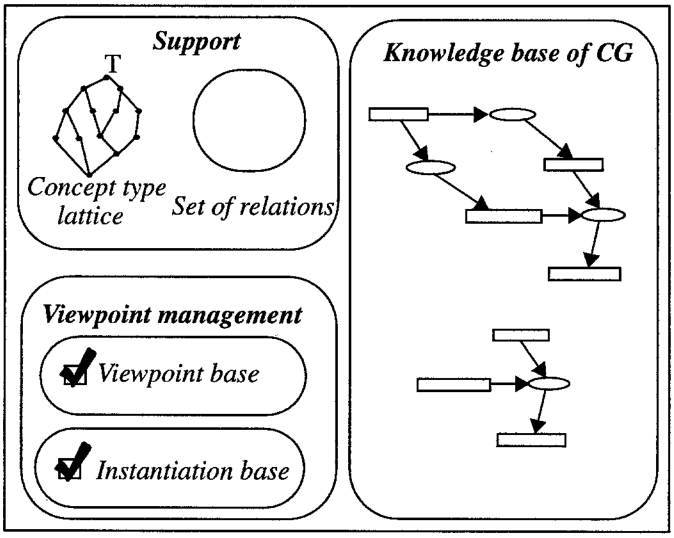


Fig. 3. Viewpoint integration in conceptual graph formalism

We present the different steps of viewpoint integration at different levels in next sections.

3.1 Viewpoint and Support

In this section we define the different elements that we introduced in the support and we describe a viewpoint knowledge base.

3.1.1 Concept Type Lattice

Viewpoints are already underlying in the concept type lattice but they are implicit. The information contained implicitly in the subtype link can be explicitly expressed by a viewpoint. When, in the classic support, a concept type has several fathers, it generally corresponds to the use of several implicit viewpoints for building the lattice: such viewpoints are implicit in the ordering relation upon which the concept type lattice relies.

Definition:
Let C, C' two concept types. If C' is a subtype of C, then a viewpoint P can be made explicit, such that C' is a subtype of C according to the viewpoint P. Once such a viewpoint made explicit, C is called a «basic concept type» and C' an «viewpoint oriented concept type» (noted v-oriented concept).

Note:
- C' itself can also be a basic concept type, if it is decomposed through at least one explicit viewpoint. In this case we must consider the two characteristics (v-oriented and basic) of the concept type for the viewpoint management.
- A v-oriented concept type can be subtype of several basic concept types.

A viewpoint is then the explicit expression of a particular subtype relation existing between two concept types.

3.1.2 Conceptual Relations

In the set of conceptual relations, we introduce the notion of «viewpoint relation», which expresses a more precise subtype link. This viewpoint relation has two concept types in its signature. Esch in [6] uses a superior level concept type TYPE, to manipulate second order concept types. So our viewpoint relation is a second order binary relation between two concept types. The signature of this relation type is (TYPE, TYPE).
In a previous work [11], we defined second order relations with an associated behaviour. A «viewpoint relation» is one of such relations, and its associated behaviour enables to manage consistency between the subtype link in the concept type lattice and the expression of a viewpoint between the same concept types. This behaviour is particularly realized during the process of creation or modification of a viewpoint (i.e. addition or removal of a concept type in the concept type lattice) which has an impact on the subtype link.
Let the following graph G:

$$\begin{tikzcd}\boxed{\text{TYPE: Basic\_type\_}t_1} \arrow[r] & (\text{VPt}) \arrow[r] & \boxed{\text{TYPE: v-oriented\_type\_}t_2}\end{tikzcd}$$

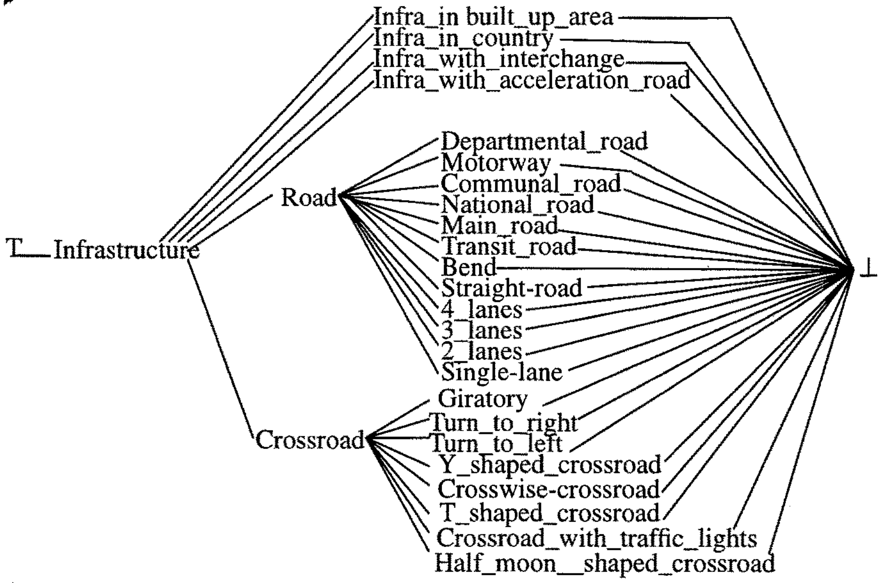
G is read: «the v-oriented type t_2 is a subtype of the basic type t_1 according to the viewpoint VPt»

Definition:
Let C_1, C_2 two concept types. If $C_1 < C_2$, then there may exist a «viewpoint relation» VPt such that C_1 is a subtype of C_2 according to VPt. VPt is a second order conceptual relation of signature (TYPE,TYPE). Then C_1 is a basic concept type and C_2 a v-oriented concept type.

Remark: According to Wermelinger's theory [12], there are several finite concept type lattices (one for each order) and a finite set of finite relational concept type lattices (one for each order).There are several relation type hierarchies: relation types are classified according to their arity and order. Each hierarchy contains all relations with the same signature (i.e. order, arity and arguments' orders).

So, as the relation type SUBTYPE, our «viewpoint relation type» VPt belongs to the hierarchy of dyadic relation types and of signature (TYPE, TYPE), in the lattice of second-order relation types. TYPE belongs to the second-order concept type lattice. In addition, VPt < SUBTYPE.

▷ a concept type lattice based on accidentology



▷ set of viewpoint relations: (VPt_curvature, VPt_administrative, VPt_lane_nb, VPt_administrative&town).

▷ a part of the viewpoint knowledge base:

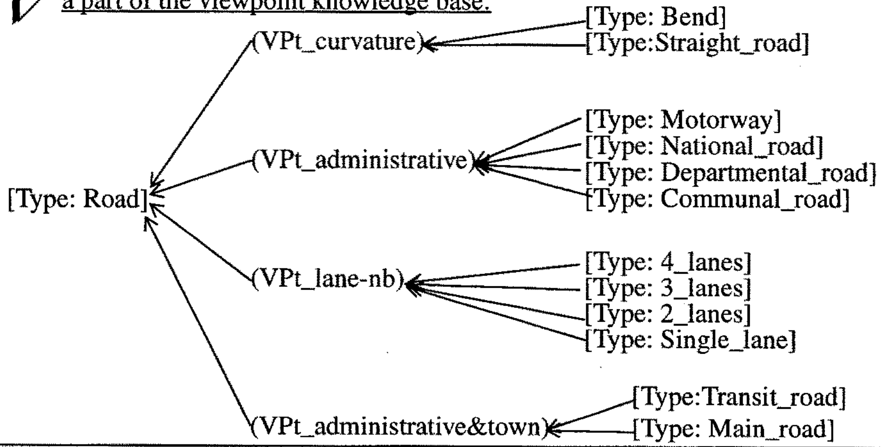


Fig. 4. Example of viewpoint expression in the domain of accidentology

3.1.3 Viewpoint Knowledge Base

Modifications and definitions brought in the two previous sections allow to build a knowledge base, where all subtype links, according to a viewpoint, are described. We call this base a «viewpoint base». We choose to represent viewpoints outside the support, in order to avoid to enforce a systematic use of viewpoints in conceptual graphs. Indeed a viewpoint must be used only if a subtype link contains information which can help in the information retrieval or modelling. This is how we introduce viewpoints in a conceptual graph base. We present an example [fig. 4] in the domain of accidentology to illustrate the aim of such a module [fig. 3].

This example shows that different criteria were used to define the subtypes of Road. The introduction of viewpoints makes these criteria explicit and allows the classification of the different kinds of subtypes. Those different viewpoints come from several experts having different competence domains. Each expert gives importance to some viewpoints on the definition of a road. The infrastructure expert gives an importance to the viewpoints: «VPt_administrative», «VPt_administrative&town» and «VPt_lane_nb», while the psychologist points to «VPt_curvature» or «VPt_lane_nb» that help him/her to make the driver's error explicit to be considered as a factor of the accident.

The consequence is that if an expert only wants to see a part of the concept type lattice depending on his/her competence domain and/or an other competence domain dependent on an other expert, the adequate information to be shown, can be filtered through viewpoints. It may be very efficient for a complex system, having a wide concept type lattice.
Each user (expert) can work with only the concept types that are relevant for his/her expertise. The concept type lattice plays the role of a common concept type lattice, from which several concept type sets, dedicated to the intended user, can be extracted. Such sets are partly organized in a hierarchy but not in a lattice structure.

3.2 Viewpoint and Conceptual Graphs

After having defined how to introduce viewpoints in the support, let us now define how to construct a concept and a conceptual graph with viewpoints.

3.2.1 Instantiation of a Concept Type

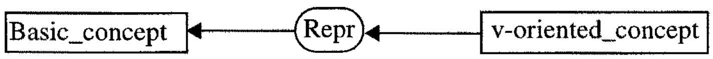
If a concept type is a «basic concept type», then it has several representations through the different viewpoints.

Definitions:
- Let Tc a basic type concept, and let r an individual or generic referent. [Tc:r] is an instantiation of the «basic concept type» Tc. It is called «basic concept».
- Let To a v-oriented concept type, and let r an individual or generic referent. [To:r] is an instantiation of the «v-oriented concept type» To. It is called «v-oriented concept».

Each representation of a given basic concept describes the same object as this basic concept, but with different levels of precision due to the specialization characterized by the considered viewpoint. So if a basic concept type is instantiated, then it can be described by the instantiation of any of its subtypes according to the different viewpoints characterizing this object.

Here, we introduce a new relation, called «Repr», which links a concept to one of its representations.

Let the graph R:



R can be read: The concept «v-oriented_concept» is a representation of the concept «basic_concept».

Repr is a first order relation. It can be expressed only if the two concept types are linked through a viewpoint relation in the viewpoint knowledge base. Two concepts linked by a «representation relation» describe the same object (cf. fig.5).

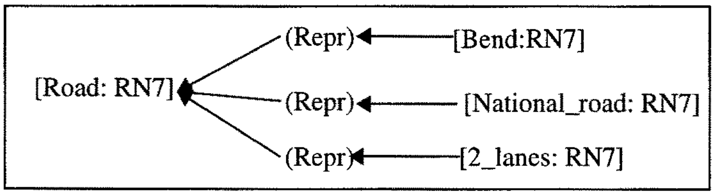


Fig. 5. Example of instantiation graph

3.2.2 Instantiation Knowledge Base

Using this relation of representation, we can now build a knowledge base containing each instantiation of each basic concept type . This base (called «instantiation knowledge base» [fig. 3]) complements the viewpoint knowledge base. In section 5, we detail the interest of both bases.

Two methodologies for building this instantiation base can be thought out:

- We can oblige the user to declare all the instantiations of a concept type in the instantiation knowledge base, before using them in a conceptual graph. In this case, the v-oriented concept types connected by Repr relations with the basic type instantiated, must have the same referent.
- Otherwise, when a concept type C is instantiated, we must verify if it is a subtype of a basic concept type. In that case, we can build an instantiation graph of the basic concept type with the referent proposed for the instantiation of the v-oriented concept type (with verification of the conformity relation). If C is a basic concept type, the adequate instantiation graph must be built.

We can notice that two representations of the same concept can be incompatible (e.g. [National_road:RN7] and [Communal_road:RN7]). Therefore we will introduce in the next section the exclusion relation between two v-oriented concept types.

3.2.3 Additional Relation for Instantiation Management

In this section we detail different second order relations, which can help to manage the representation relation or to make deductions. Those relations depend on the representation relation. They can be expressed in the viewpoint knowledge base [fig. 3].

Several definitions presented in this section will use the following links [fig. 6] (based on the IF...THEN graphs defined by Sowa in [14]).

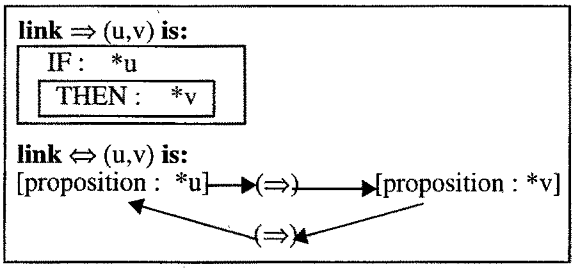


Fig. 6. Definition of implication and equivalence links

3.2.3.1 Exclusion Relation

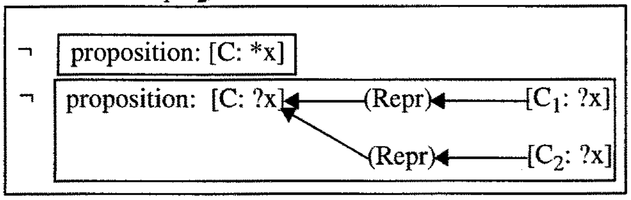
An exclusion is only an intra-viewpoint relation.

Definition of excl (C,C_1,C_2):

Let C a basic concept type, C_1 and C_2 two v-oriented concept types both subtypes of C. If «Repr» connects the v-oriented concept c_1=[C_1:ref] to the basic concept c=[C:ref] then «Repr» cannot connect the v-oriented concept type c_2=[C_2:ref] to c.

Graph representation:

relation excl (C,C_1,C_2)



This relation allows to avoid incompatible instantiations. We precise the basic concept type C, because an exclusion relation cannot exist between two v-oriented concept types subtyping two different basic concept types.

Relation behaviour:

Monotonic reasoning: before adding in the instantiation base a new «Repr» relation between a new v-oriented concept Co and a basic concept, it is checked whether none of the types of the v-oriented concepts already representing this basic concept in the instantiation base has any «excl» relation with the type of Co. If one such exclusion relation is found, the new «Repr» relation with Co is rejected.

3.2.3.2 Equivalence Relation

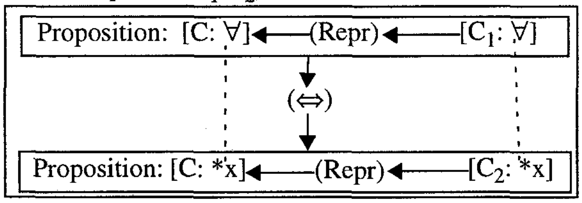
Equivalence relation is an inter-viewpoints relation.

Definition of equiv (C,C_1,C_2):

Let C a basic concept type and C_1 and C_2 two v-oriented concept types both subtypes of C. If «Repr» connects the v-oriented concept c_1=[C_1:ref] to the basic concept c=[C:ref], then «Repr» necessarily connects the v-oriented concept type c_2=[C_2:ref] to c and vice versa.

Graph representation:

relation equiv (C,C_1,C_2):


Proposition: [C: ∀]←(Repr)←[C_1: ∀]
(⇔)
Proposition: [C: *x]←(Repr)←[C_2: *x]

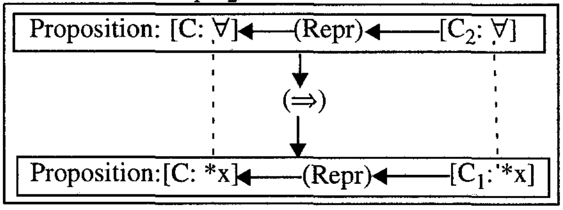
Relation behaviour:
If in a viewpoint graph, there exists an «equiv» relation between, on the one hand C_1 the type of c_1=[C_1:ref], a v-oriented concept connected with Repr to a basic concept c=[C:ref], and on the other hand C_2, then a v-oriented concept type c_2=[C_2:ref] is created if needed and a «Repr» relation connecting c_2 and c=[C:ref] is created.

3.2.3.3 Inclusion Relation

Definition of incl(C,C_1,C_2):
Let C a basic concept type and C_1 and C_2 two v-oriented concept types both subtypes of C. If «Repr» connects the v-oriented concept type c_2=[C_2:ref] to the basic concept c=[C:ref], then «Repr» necessarily connects c_1=[C_1:ref] to c (but not vice versa).

Graph representation:

relation incl (C,C_1,C_2):


Proposition: [C: ∀]←(Repr)←[C_2: ∀]
(⇒)
Proposition:[C: *x]←(Repr)←[C_1: *x]

Relation behaviour:
If in the instantiation base, there exists c_2=[C_2:ref] a v-oriented concept connected through Repr to its basic concept c=[C:ref], and if in the viewpoint base, there exists an «incl» relation between C_1 a v-oriented concept type, and the v-oriented concept type C_2, then the v-oriented concept c_1=[C_1:ref] is created if needed in the instantiation base and a «Repr» relation connects c_1 to c.

4. Viewpoint Management in Conceptual Graph Formalism

The two bases described in the previous section allow to have an independent viewpoint management through the conceptual graph formalism. A user can choose to use or not the viewpoint management. So we call «viewpoint management» the set created by the two bases. [fig. 7] shows the different interactions between the different entities. There are two sorts of interactions:

•interaction for consistency management

•interaction for construction of conceptual graph.

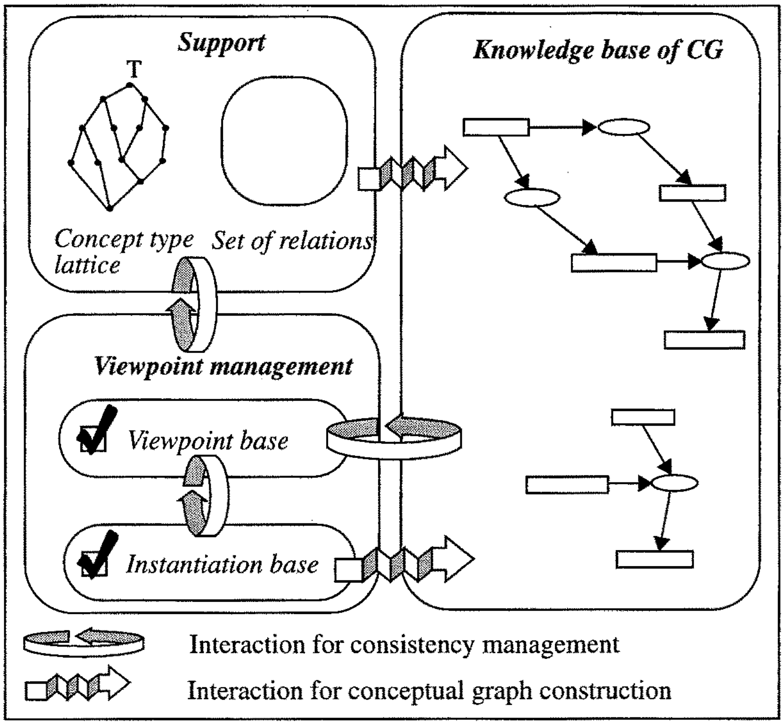


Fig. 7. Viewpoint management in conceptual graph formalism

4.1 Interaction for Consistency Management

There are three kinds of interactions for consistency management:

•The interaction between the support and the viewpoint base is to manage consistency between the concept type lattice (which expresses all the subtype links between concept types) and the viewpoint base (which expresses all the subtype links according to viewpoints). We must manage at the same time the creation, suppression, and modification of concept types in the two parts.

This is an example of property that must be satisfied:
If [TYPE:T_1]←(VPt)←[TYPE:T_2] in the viewpoint base then $T_2 < T_1$ in the concept type lattice (even more, T_2 is a direct subtype of T_1 in the concept type lattice).

- •The interaction between the two bases of the viewpoint management, with the different relations (exclusion, inclusion and equivalence). We must check also during the creation of a «Repr» relation, if the involved v-oriented concept type is subtype of the basic concept type, according to a viewpoint. The conformity relation must also be satisfied.
- •The interaction between the viewpoint management module and the knowledge base of CG allows the verification of the relation signatures. Indeed if a conceptual relation has a v-oriented concept type as maximal concept type in its signature, the other arguments must be in the common viewpoints. It allows also consistency management between information in viewpoint management module and conceptual graph base [fig. 8].

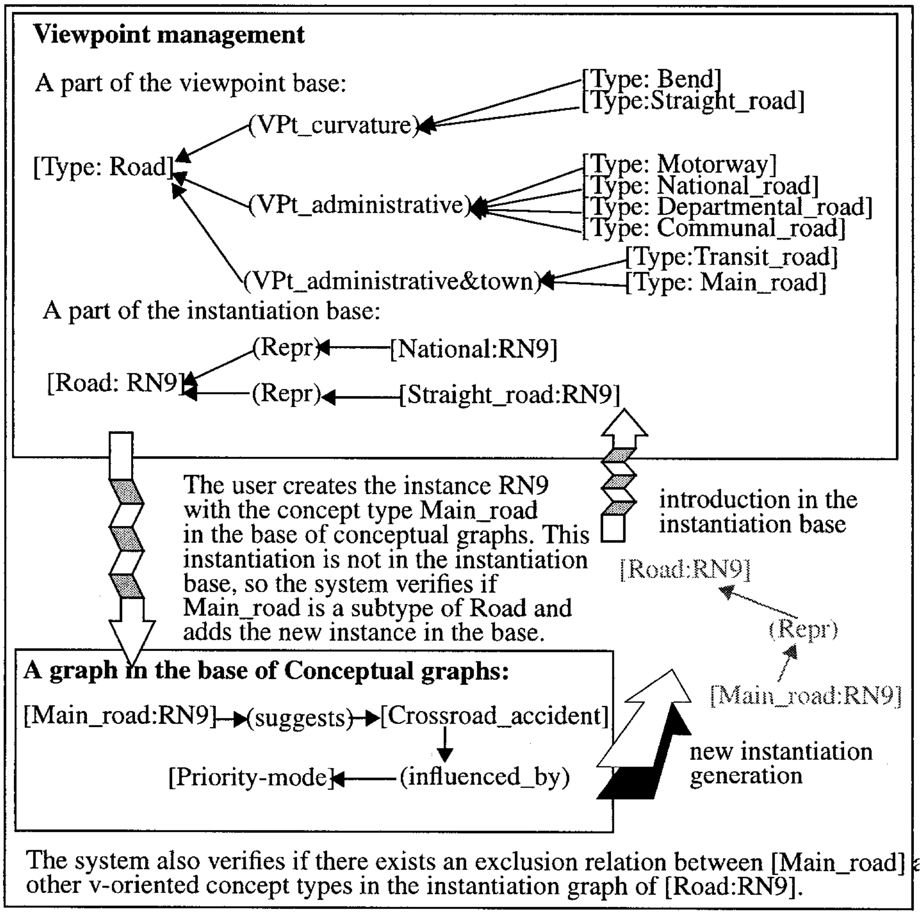

Viewpoint management
A part of the viewpoint base:
[Type: Bend]
[Type:Straight_road]
(VPt_curvature)
[Type: Road]
[Type: Motorway]
[Type: National_road]
(VPt_administrative)
[Type: Departmental_road]
[Type: Communal_road]
[Type:Transit_road]
(VPt_administrative&town)
[Type: Main_road]
A part of the instantiation base:
(Repr)
[National:RN9]
[Road: RN9]
(Repr)
[Straight_road:RN9]
The user creates the instance RN9 with the concept type Main_road in the base of conceptual graphs. This instantiation is not in the instantiation base, so the system verifies if Main_road is a subtype of Road and adds the new instance in the base.
introduction in the instantiation base
[Road:RN9]
(Repr)
[Main_road:RN9]
A graph in the base of Conceptual graphs:
[Main_road:RN9]→(suggests)→[Crossroad_accident]
[Priority-mode]←(influenced_by)
new instantiation generation
The system also verifies if there exists an exclusion relation between [Main_road] a
other v-oriented concept types in the instantiation graph of [Road:RN9].

Fig. 8. Example of interaction

4.2 Interaction for Conceptual Graph Construction

The support allows to build a conceptual graph without viewpoint. Using both the instantiation base and the support permits conceptual graph construction with viewpoints. All the instantiation through viewpoints made in a graph, must be stored in the instantiation base before their use (we choose the first methodology described in the section 3.2.2).

4.3 Rules on Conceptual Graph and Viewpoints

The integration of viewpoints has an influence on the definition rules on CG.
There are four canonical formation rules: Copy, Restrict, Join and Simplify [13], that can normally be used for the formation of classic CG. But we can define rules derived from them and specific for the viewpoint management: in particular the Restrict and Join rules can be redefined according to a viewpoint. So we created a new rule, based on Sowa's Restrict rule:

Definition: Restrict w.r.t. a given viewpoint
Let v a viewpoint. The restriction w.r.t. the viewpoint v is obtained by replacement of the concept label with a more specific label: replacement of a generic referent with an individual referent and/or replacement of the concept type with a subtype according to the viewpoint v (the conformity relation must be verified).

Sowa's Join rule is based on the identity of concept. As we have defined an equivalence relation on concept types, this relation helps us to define a new join relation with respect to the equivalence relation in the viewpoint base:

Definition: Join w.r.t the equivalence relation inter-viewpoints
Let c_1=[C_1:ref] a v-oriented concept of a graph G_1 and c_2=[C_2:ref] a v-oriented concept of a graph G_2. If there exists an «equiv» relation between C_1 and C_2, then we can join G_1 and G_2 by identification of c_1 and c_2. A relevant concept among c_1 and c_2 must be chosen for the join.

4.4 Projection Operation and Viewpoints

The projection of a graph G into a graph G' is not really modified, but we can also focus the projection according to a viewpoint v:

Definition: projection operation w.r.t. a viewpoint
Let G and G' two conceptual graphs. The viewpoint projection according to a viewpoint v, of G into G' is an application π:G→G', where πG is a subgraph of G', such that:
- For each concept c in G, πc is a concept in G' and type(πc) is a subtype of type(c) according to the viewpoint v.
- For each conceptual relation r in G, type(πr)=type(r). If the ith arc of r is linked to a concept c in G, the ith arc of πr must be linked to πc in G'.

After a classic projection operation, we can also verify whether the result of this clas-

sic projection corresponds in fact to a projection w.r.t. a viewpoint: if there exists a viewpoint v such that for each concept c of G, type(πc) is a subtype of type(c) according to this viewpoint v, then πG is a subgraph of G' according to the viewpoint v.
This precision allows to know if two different experts say the same thing in different viewpoints. They can describe the same thing with their own concepts that depend on their knowledge, competence domain, and perception of the situation or the object to describe.
This projection operation with the viewpoint notion permits comparison of expertise through the conceptual graph formalism. So it is very interesting to know how to represent those different representations and perceptions.

5. Conclusion

In this paper, we described the interest of viewpoints in knowledge representation and we proposed an introduction of viewpoints in conceptual graph formalism.

5.1 Comparison with TROPES and ROME

The creation of the viewpoint base relies on several elements in TROPES model like the class arborescence and bidirectional, unidirectional bridges. Indeed our viewpoint base contains conceptual graphs that describe partially the class arborescence of TROPES thanks to the introduction of the viewpoint relation. Our equivalence and inclusion relations play the same role as bidirectional and unidirectional bridges. Therefore our instantiation base uses the simple instantiation of ROME and the description of multi-representations via a representation relation like in ROME.

5.2 Related Work and Perspectives

The view notion was already used in [4], but instead of introducing viewpoints in conceptual graphs, the aim was rather to use conceptual graph formalism as a meta-language to analyze and compare different specifications of requirements with respect to each other. The multiple views were in fact multiple schemas of representation.
In [8], Martin associates concepts or conceptual graphs to elements of structured documents. He uses viewpoints to permit different representations of the same document element in a knowledge base in order to compare them. Our aim is to define viewpoints to help knowledge representation for multi-expert knowledge acquisition and also to have an accessible and evolutive knowledge base through viewpoints. Our approach is also aimed at making comparison of expertises.

We can notice the interest of exploiting second order relations, with an associated behaviour, as suggested in a previous work [11]. Other relations can be exploited. Our perspective is to extend the viewpoint management to the «composed-of» relation, and to test all the viewpoint management on an application.

References

1. D.G. Bobrow, T. Winograd, *An overview of KRL, a Knowledge Representation Language*, Cognitive Science, vol.1 n°.1, p. 3-45, 1977.

2. B. Carré, L. Dekker and J-M. Geib. *Multiple and Evolutive Representation in the ROME Language. Towards an integrated Corporate Information System.* In Proc. TOOLS' 90, Paris, 26-29 June 1990.

3. L. Dekker and B. Carré. *Multiple and Dynamic Representation of frames with Points of View in FROME.* In Proc. RPO'92. La Grande Motte, France, 22-23 June, 1992.

4. H. Delugach, *Specifying Multiple-Viewed Software Requirements With Conceptual Graphs*, Jour. Systems and Software, vol. 19, p. 207-224, 1992.

5. G.Ellis, *Object-Oriented Conceptuals Graphs*, In Ellis et al eds, Conceptual Structures: Applications, Implementation and Theory. Proc. of ICCS'95. Springer Verlag. Santa Cruz, CA, USA, Aug. 1995, p. 144-172.

6. J. Esch, *Contexts, Canons and Coreferent Types.* In Terpfenhart &al eds, Conceptual Structures: Current Practices: Proc. of the ICCS'94. College Park, Maryland, USA, August 1994, Springer Verlag, LNAI n. 835, p. 185-195.

7. O. Marino, F. Rechenmann, P. Uvietta, *Multiple perspectives and classification mechanism in Object-oriented Representation*, Proc. 9th ECAI, Stockholm, Sweden, p. 425-430, Pitman Publishing, London, August 1990.

8. P. Martin, *Exploitation de graphes conceptuels et de documents structurés et hypertextes pour l'acquisition de connaissances et la recherche d'information*, PhD Thesis, University of Nice-Sophia-Antipolis, October 1996.

9. M. Minsky, *A Framework for Representing Knowledge*, in The psychology of Computer Vision, McGrawHill, New York, P.H. Winston (ed), Chap. 6, p. 156-189, 1975.

10. G.T. Nguyen, D. Rieu, J. Escamilla, *An Object Model for Engineering Design*, in Proc. of ECOOP'92, Utrecht, The Netherlands, June/July 1992, Springer-Verlag, p. 232-251.

11. M. Ribière, R. Dieng, M. Blay-Fornarino, A-M. Pinna-Dery, *Link-based Reasoning on Conceptual Graphs*, in Suppl. Proceedings of ICCS'96, Sydney, Australia, August 1996, p 146-160.

12. M. Stefik, D.G. Bobrow, *Object-Oriented Programming: Theme and Variations.* The A.I. Magazine, vol. 6, n° 4, p 40-62, 1985.

13. J. F. Sowa. Conceptual Structures, *Information Processing in Mind and Machine.* Reading, Addison-Wesley, 1984.

14. J. F. Sowa. *Relating Diagrams to Logic.* In Mineau & al eds, Conceptual Graphs for Knowledge Representation: Proc. of ICCS'93. Springer Verlag, LNAI n. 699. Quebec City, Canada, August 1993, p. 1-35.

15. M. Wermelinger. *Conceptual Structures and First-Order Logic.* In Ellis et al eds, Conceptual Structures: Applications, Implementation and Theory. Proc. of ICCS'95. Springer Verlag. Santa Cruz, CA, USA, Aug. 1995, p. 323-337.