Implementation of a knowledge representation and reasoning tool using default rules for a decision support system in agronomy applications

Patrice Buche, Virginie Cucheval, Awa Diattara, Jérôme Fortin, Alain Gutierrez

Implementation of a knowledge representation and reasoning tool using default rules for a decision support system in agronomy applications

Patrice Buche[⋆] INRA IATE, LIRMM GraphIK, France

Virginie Cucheval[⋆⋆] CTFC Poligny France

Awa Diattara[⋆⋆⋆] INRA IATE France

Jérôme Fortin[†] Université Montpellier II IATE/LIRMM GraphIK, France

Alain Gutierrez[‡] CNRS LIRMM France

Abstract

This is an application paper in which we propose to use an extended version of the conceptual graph framework to represent and reason on expert knowledge for cheese making. In this extension, we propose to use default rules to represent specific pieces of knowledge. We use the CoGui software to manage conceptual graphs in the application from the CTFC data expressed in Freeplan. A specific end user interface has been designed on top of CoGui to ensure that end-user experts in cheese making can use it in a convenient way.

1 Introduction

The CTFC (Centre Technique des Fromages Comtois, which means Technical Centre for Comtois Cheese) is a technical centre in the east part of France that helps traditional cheese companies to manage cheese making process. For example when a cheese maker finds a problem in some cheeses (as bitterness or a consistency default), some diagnostic can be made by the CTFC in order to propose a solution to the cheese maker. For this purpose, the CTFC needs to capitalize knowledge and experience about traditional cheese making process. This knowledge is owned, mainly in an informal way, by a group of experts employed by CTFC. Capitalizing this knowledge is a crucial topic to maintain the quality of specific cheese making process as Comté, Morbier, Bleu de Gex or Mont d'or. It is also important to capitalize expert knowledge because some processes are not fashionable at a given moment but may become fashionable in the future. For example the traditional Comté had some holes like Swiss Gruyère around 15 years ago. There is no more hole nowadays in Comté, so if the knowledge on the way to obtain Comté with holes is not capitalized, nobody will be able to do it again in the future. As this knowledge is not always formally represented, the risk to loose this information is important.

This is why the CTFC has decided to develop a knowledge management system that allows to capitalize technological and scientific knowledge about cheese making. The implementation of this system requires a methodology to collect the operational knowledge of technical experts and identify the underlying scientific knowledge. A tool to structure this knowledge and a tool able to reason on this knowledge to provide answers to cheese experts queries are also required.

A methodological approach has been defined to collect the expert knowledges through two kinds of interviews: on the one hand, individual interviews and on the other hand, collective and contradictory ones. This approach is now operational and the expert knowledge resulting from those interviews is represented in a tree structure using a free mind mapping software called Freeplan. Once the knowledge collected and entered in Freeplan, a "scientific validation" session is done in collaboration with INRA researchers (National Institute of Agronomy) of the URTAL unit in Poligny. A lack of Freeplan is that pieces of information must be stored in separate files as the information is voluminous and no reasoning tool is available to use this information. For example, informations may be easily displayed in Freeplan mind map. But, if the user wants to find the list of possible corrective actions in order to control a given property of the cheese (quality or default), it becomes difficult to display all the mind maps in a short time. Therefore, automatic operations are required to analyze the knowledge. As the CTFC wants to manage the knowledge associated with around 50 properties of cheese, a complementary tool that permits to query automatically the information is required.

To reach this aim, our approach consists in translating the knowledge described in Freeplan mind maps into the conceptual graph formalism which permits to perform automatic reasoning. To implement the tool, we use CoGui free software, which is a conceptual graph editor that permits to manage the terminology of an application domain, the facts, the rules and more recently default rules. We developed a user interface on top of CoGui to ensure that end-users of the application can easily use it without knowing anything about the conceptual graph formalism.

The paper is organised as follows : the next section presents how the CTFC actually structures its knowledge using Freeplan mind maps. In Section 3, we recall the conceptual graph formalism and explain how the expert knowledge

⋆ Patrice.Buche@supagro.inra.fr
⋆⋆ v-cucheval@ctfc.fr
⋆⋆⋆ awa.diattara@supagro.inra.fr
† fortin@polytech.univ-montp2.fr
‡ alain.gutierrez@lirmm.fr

of the CTFC can be entered in CoGui using rules and default rules. Section 4 presents the end-user application, built on top of CoGui, which permits experts to access to the knowledge without particular background formalism. We conclude the paper in the last section.

2 Structuring CFTC knowledge using Freeplane

Currently, expert and scientific CTFC knowledge are structured in a tree using the open software Freeplane. The structure of the tree is performed in such a way that from a given descriptor (defined as a desired or not desired property of cheese), we list all explanatory mechanisms, from the most general to the most specific and leading finally to some corrective actions (see Figure 1). We read this figure as follows : The descriptor (which may be, for example, a default in the cheese making process) can be explain by Explanatory Mechanism 1 or by Explanatory Mechanisms 2... Each mechanism can be explained by several sub-mechanisms. This way to consider the different mechanisms to explain a default naturally leads to a tree structure. Finally a mechanism is associated with a particular corrective action, which can permit to control the descriptor (for example, to avoid a default if the descriptor represents a default) which is the root of the tree.

Moreover, relationships between explanatory mechanisms and more complex actions must be taken into account. We can cite for example the joint effects or the compensatory actions, which are presented in the following.

We're talking about joint effect when two or more explanatory mechanisms should be grouped together to affect the descriptor or the $n-1$ mechanism level. The effect is expressed in the Freeplan tree by the creation of the relationship "*AND*". An example of joint effect in given in Figure 2, which represents that the descriptor "salt intake after 15 days is low" is explain by the mechanism "low salt absorption", which is explained by "low granularity of cheese crust", which is jointly explained by "using smooth mold" and "high duration contact between cheese and mold ($> 8h$)".

A corrective action is defined as a way to control a descriptor (for example to correct a default) for the next manufacturing. On another hand, a compensatory action is an action that is taken to control a descriptor (for example to correct a default) during the current manufacturing. Compensatory actions are expressed on the tree using the relationship "UNLESS" ("SAUF SI" in french). Figure 3 presents an example of compensatory action. It means that we can explain that a "badly realized brining" is due to a "use of a low concentration of brine ($< 250g/L$))" unless the "brining time extended".

The mind map model of Freeplane can represent all the knowledge expressed by the CTFC. However, the same explanatory mechanisms may appear in different trees (concerning different descriptors). In this case, using Freeplan model leads to duplicate many informations, which is not satisfactory for several reasons : it is a waste of time to manage duplications, especially when the knowledge must be updated, and this may lead to some inconsistencies of the knowledge base if updates are not propagated in all the duplicates.

Moreover, no reasoning can be performed using the Freeplan software. To overcome this drawback, we propose to use the CoGui software which allows both the representation of knowledge and reasoning with the conceptual graph model. It includes all the required elements to represent the explanation mechanisms and the joint effect relationships.

3 Structuring knowledge of CTFC with CoGui

In general, a knowledge representation formalism should satisfy three essential criteria. First, it must allow to represent the knowledge of an application domain in a declarative way, meaning that the semantics associated with the knowledge must be defined regardless of the programs that use the knowledge base. The second criterion is that for reasoning, it must allow to make inferences that are based on logic. The third criterion is to structure the knowledge in an unambiguous way: it means that the informations linked in a semantic way should be grouped and that the knowledge of different natures must be clearly differentiated. Conceptual graphs (noted CG in the following) and CoGui (one CGs editor) satisfy all these three essential criteria.

3.1 The Conceptual Graph Formalism

The conceptual graph formalism [Sowa, 1984; Chein and Mugnier, 2009] is a knowledge representation and reasoning formalism based on labelled graphs. In its simplest form, a CG knowledge base is composed of two parts: the *support*, which encodes terminological knowledge –and constitutes a part of the represented domain ontology– and *basic conceptual graphs* built on this support, which express assertional knowledge, or *facts*. The knowledge base can be further enriched by other kinds of knowledge built on the support: in this paper, we will consider two kinds of *rules*: "usual rules" and CG defaults, which lead to non-monotonic reasoning.

The support. It provides the ground vocabulary used to build the knowledge base. It is composed of a set of *concept types*, denoted by $\mathcal{T}_C$, and a set of *relation types* (or simply *relations*), denoted by $\mathcal{T}_R$. Relation types represent the possible relationships between concept instances, or properties of concept instances for unary relation types. $\mathcal{T}_C$ is partially ordered by a *kind of* relation, with $\top$ being its greatest element. $\mathcal{T}_R$ is also partially ordered by a *kind of* relation, with any two comparable relation types having necessarily the same arity (i.e., number of arguments). Each relation type has a signature that specifies its arity and the maximal concept type of each of its arguments.

CoGui (Conceptual Graphs Graphical User Interface)[1] is a software which permits to build knowledge bases as CGs. It provides a Java graphical interface for editing support, CGs , rules and constraints. The knowledge base can be serialized in a XML predefined format called CogXML. It includes a querying system based on forward chaining mechanism for querying a knowledge base.

[1] http://www2.lirmm.fr/cogui/

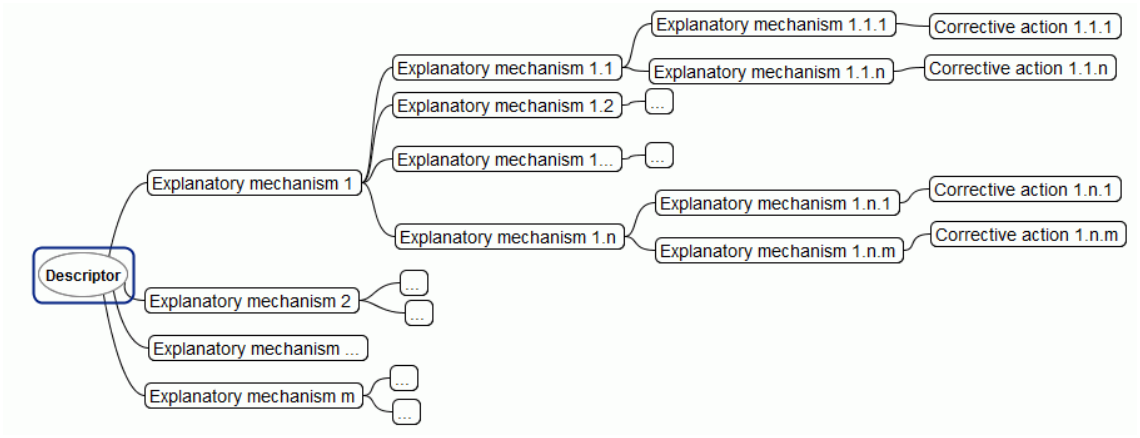

Fig. 1. Tree structure of CTFC knowledge expressed using the FreePlan software

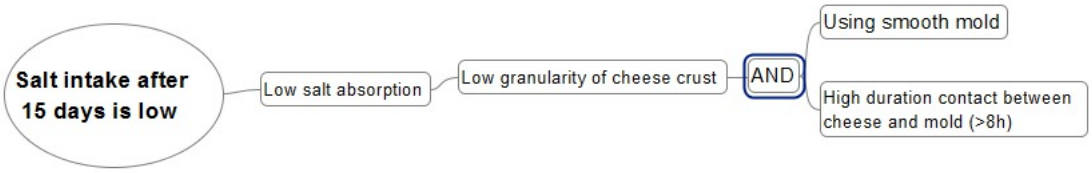

Fig. 2. Representation of a joint effect

Fig. 3. Example of a compensatory action (ACOM)

Definition of the support for the CTFC application. We recall that all the knowledge of CTFC is represented as trees in Freeplane. To model this knowledge in CoGui, we first define the basic vocabulary as concept types and relation types. The concept types used to represent the specific vocabulary of the CTFC application are of three categories:

- **cheese descriptors** with two subtypes of concepts: the sensory descriptors and analytical descriptors,
- **explanatory mechanisms** including three subtypes of concepts: the states of milk, the states of the cheese and the process parameters,
- **actions** with two subtypes of concepts:compensatory actions which appear in UNLESS relationships and corrective actions associated with the last level of explanatory mechanisms in Freeplane trees.

We have identified two types of relations:

- **unary relations** with several subtypes to qualify the set of different sensory and analytical descriptors and corrective action mechanisms
- **binary relations** with several subtypes including: the relationship *"is explained by"* that connects a descriptor or an explanatory mechanism to an explanatory mechanism, the relationship *has for corrective action* that connects an explanatory mechanism of last level in the Freeplan tree to a **corrective action**.

Figure 4 shows on its left-side the hierarchy of concept types and on its right-side the hierarchy of relations. The current version of the CTFC knowledge base is composed of 114 concept types and 39 relation types.

Basic conceptual graphs. A basic CG is a bipartite graph composed of:

(i) a set of *concept nodes* (pictured as rectangles), which represent entities, attributes, states or events;

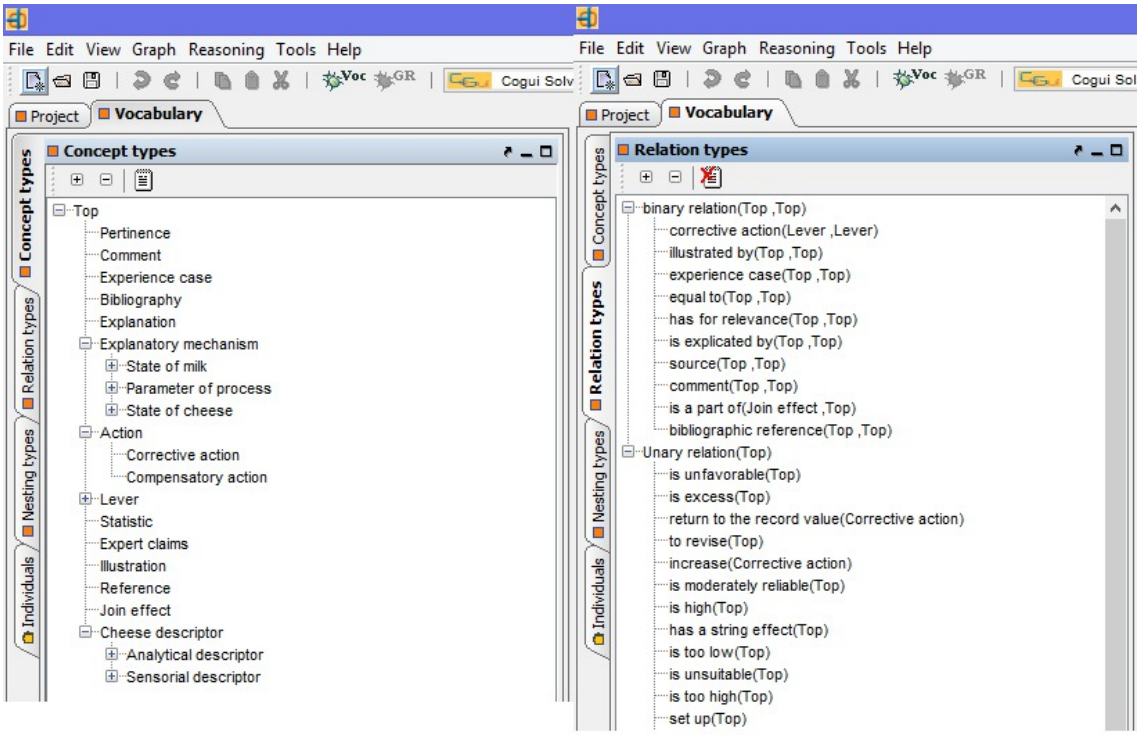

Fig. 4. Hierarchy of concept types and relation types for the CTFC application

(ii) a set of *relation nodes* (pictured as ovals), which express the nature of relationships between concept nodes;

(iii) a set of *edges* linking relation nodes to concept nodes;

(iv) a *labelling* function, which labels each node or edge: the label of a concept node is a pair $t : m$, where t is a concept type and m is a marker; the label of a relation node is a relation type; the label of an edge is its rank in the total order on the arguments of the incident relation node.

Furthermore, a basic CG has to satisfy relation signatures: the number of edges incident to a relation node is equal to the arity of its type r, and the concept type assigned to its neighbour by an edge labelled i is less or equal to the i^{th} element of the signature of r. The marker of a concept node can be either an identifier referring to a specific individual or the generic marker (denoted $*$) referring to an unspecified instance of the associated concept type. The generic marker followed by a variable name (for instance $*x$) is used in a basic CG or a rule to indicate that the instance (noted x) represented by several concept nodes is the same. A basic CG without occurrence of the generic marker is said to be totally instantiated.

Logical translation. Conceptual graphs have a logical translation in first-order logic, which is given by a mapping classically denoted by ϕ. ϕ assigns a formula $\phi(S)$ to a support S, and a formula $\phi(G)$ to any basic CG G on this support. First, each concept or relation type is translated into a predicate (a unary predicate for a concept type, and a predicate with the same arity for a relation type) and each individual marker occurring on the graphs is translated into a constant. Then, the *kind of* relation between types of the support is translated by logical implications.

Given a basic conceptual graph G on S, $\phi(G)$ is built as follows. A distinct variable is assigned to each concept node with a generic marker. An atom of the form $t(e)$ is assigned to each concept node with label $t : m$, where e is the variable assigned to this node if $m = *$, otherwise $e = m$. An atom of the form $r(e_1, \ldots, e_k)$ is assigned to each relation node with label r, where e_i is the variable or the constant corresponding to the i^{th} neighbour of the relation. $\phi(G)$ is then the existential closure of the conjunction of all atoms assigned to its nodes.

Specialization relation, homomorphism. Any set of conceptual graphs is partially preordered by a *specialization relation*, which can be computed by a *graph homomorphism* (allowing the restriction of the node labels), also called *projection* in the conceptual graph community. The specialization relation, and thus homomorphism, between two graphs, corresponds to the logical entailment between the corresponding formulas, i.e., there is a homomorphism from G to H both built on a support S if and only if $\phi(G)$ is entailed by $\phi(H)$ and $\phi(S)$ (see e.g., [Chein and Mugnier, 2009] for details)[2].

The specialization relation is particularly interesting in our application to query the knowledge base. For example, an expert will be able to look for any kind of explanatory mech-

[2] Note that, for the homomorphism completeness part, H has to be in normal form: each *individual* marker appears at most once in it, i.e., there are no two concept nodes in H representing the same identified individual.

anisms or can specify that he/she wants only to retrieve the explanatory mechanisms associated with specific parameters of the process.

Basic CG rules. Basic CG rules [Salvat and Mugnier, 1996] are an extension of basic CGs. A CG rule (notation: $R = (H, C)$) is of the form "if H then C", where H and C are two basic CG (respectively called the *hypothesis* and the *conclusion* of the rule), which may share some concept nodes. Generic markers referenced by a variable name as $*x$ refer to the same individual in the hypothesis and in the conclusion. Graphically, it can be represented by a single bicolored basic CG.

A rule R is applicable to a basic CG G if there is a homomorphism from its hypothesis to G. Let π be such a homomorphism. Then, the *application of R on G according to π* produces a basic CG obtained from G by adding the conclusion of R according to π, i.e., merging each frontier node c of the added conclusion with the node of G that is image of c by the homomorphism π.

The mapping ϕ to first-order logic is extended to CG rules. Let $R = (H, C)$ be a CG rule, and let $\phi'(R)$ denote the conjunction of atoms associated with the basic CG underlying R (all variables are kept free). Then, $\phi(R) = \forall x_1 \dots \forall x_k(\phi'(H) \rightarrow (\exists y_1 \dots \exists y_q \phi'(C)))$, where $\phi'(H)$ and $\phi'(C)$ are the restrictions of $\phi'(R)$ to the nodes of H and C respectively, $x_1, \dots, x_k$ are the variables appearing in $\phi(H)$ and $y_1, \dots, y_q$ are the variables appearing in $\phi(C)$ but not in $\phi(H)$.

Given a set of rules $\mathcal{R}$, basic CGs G and H (representing for instance a query and a set of facts), all defined on a support S, $\phi(G)$ is entailed by $\phi(H)$, $\phi(S)$ and the logical formulas assigned to $\mathcal{R}$ if and only if there is a sequence of rule applications with rules of $\mathcal{R}$ leading from H to a basic CG H' such that there is a homomorphism from G to H' (in other words, by applying rules to H, it is possible to obtain H' which entails G).

When a rule is applied, it may create new individuals (one for each generic concept node in its conclusion, i.e., one for each existential variable y_i in the logical translation of the rule). In the following, we will assume that all facts (represented as basic CGs) are completely instantiated. Then, when a rule is applied, we will instantiate each new generic concept node created, by replacing its generic marker with a new individual marker (which can be seen as a Skolem function, moreover without variable in this case). This way of doing will allow us to represent CG defaults in a simpler way (see the next section).

Translation of Freeplane trees into CG rules. In the CTFC application, each Freeplane tree is translated into a set of CG rules. To do that, each elementary explanatory mechanism (defined as a couple of explanatory mechanisms linked by the *"is explained by"* relationship) will be translated into a new CG rule. By this way, a given information about an explanatory mechanism has to be entered only once, even if is it used to explain several descriptors. Moreover as we will see in the next sections, it will be possible to reconstruct easily all the CTFC knowledge trees, just by defining a descriptor (root of the tree) as a graph fact and by applying all the rules to it.

Figure 5 is an example of a rule meaning that "a low salt rate in ripened cheese" is explained by "a low salt intake during ripening". This CG rule is associated with one of the explanatory mechanisms represented in the Freeplan knowledge tree partially given in Figure 3. Note that the hypothesis and the conclusion of the rule are presented as two different CGs, linked together by a co-reference link (represented by a dashed line in the figure 5).

Default Rules in the Conceptual Graph Formalism

We now present an extension of CG rules, which has been introduced in [Baget *et al.*, 2009; Baget and Fortin, 2010] and allows for default reasoning. It can be seen as a graphical implementation of a subset of Reiter's default logic [Reiter, 1980]: indeed, we restrict the kind of formulae that can be used in the three components of a default. These three components are called the hypothesis H, the conclusion C and some justifications $J_1, \cdots, J_k$. We can deal directly with non–closed defaults, i.e., without instantiating free variables before processing the defaults. In Reiter's logic, the application of a default is subject to a consistency check with respect to current knowledge: each justification J has to be consistent with the current knowledge, i.e., $\neg J$ should not be entailed by it. In CG defaults, justifications are replaced by graphs called *constraints*; a constraint C can be seen as the negation of a justification: C should not be entailed by current knowledge.

Definition 1 (CG defaults). *A* CG default *is a tuple $D = (H, C, C_1, \dots, C_k)$ where H is called the hypothesis, C the conclusion and $C_1, \dots,$ and C_k are called the constraints of the default; all components of D are themselves basic CGs and may share some concept nodes.*

Briefly said, H, C and each C_i respectively correspond to the prerequisite, the consequent and the negation of a justification in a Reiter's default. H, C and the C_i's can share some concept nodes that have the same marker. These markers can be individual or generic, in which case the identification of the concept nodes is made by comparing the name of the variable associated with this generic marker.

The intuitive meaning of a CG default is rather simple: "for all individuals $x_1 \dots x_k$, if $H[x_1 \dots x_k]$ holds true, then $C[x_1 \dots x_k]$ can be inferred provided that no $C_i[x_1 \dots x_k]$ holds true". If we can map by homomorphism the hypothesis H of a default to a fact graph G, then we can add the conclusion of the default according to this homomorphism (as in a standard rule application), unless this homomorphism can be extended to map one of the constraints from the default. As already pointed out, while the negation of a justification in a Reiter's default should not be entailed, in a CG default the constraint itself should not be entailed.

The entailment mechanism is based on the construction of a default derivation tree, we let the reader refer to [Baget *et al.*, 2009; Baget and Fortin, 2010] for more precise details about default entailment mechanism.

Representing compensatory actions in the CTFC application using CG default rules. In the CTFC application, the default

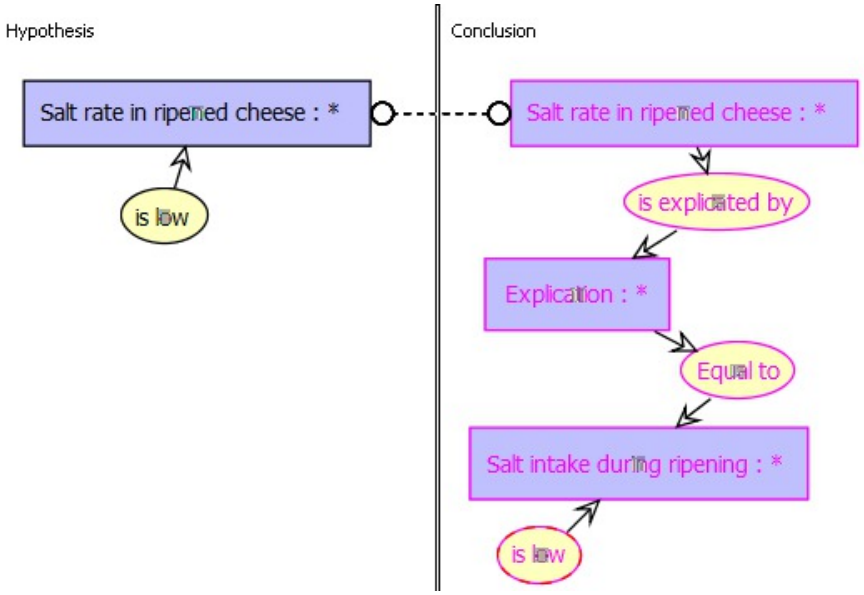

Fig. 5. Example of a standard CG rule associated with an elementary explanatory mechanism

rules permit to model the compensatory actions in the CG knowledge base. Figure 6 shows how to represent a default rule in CoGui. This CG default rule is the translation in the CG model of the Freeplan knowledge tree presented in Figure 3. The hypothesis and the conclusion of the default are shown as for standard rule, while the justification is put in grey (left part of the rule).

The current version of the CTFC knowledge base is composed of 67 CG rules including 3 CG default rules associated with 3 descriptors. As the objectives of the CTFC is to manage 50 descriptors in the knowledge base, the estimated size of the targeted knowledge base is around 1000 CG rules.

4 End-user application

The objective of this section is to describe the implementation an application for the CTFC technicians that have no background on knowledge representation models and especially on the CG model. Therefore, an end-user application must be proposed on top of the CoGui software used to manage the CG knowledge base.

Functional requirements address the specifications defined for the CTFC application. This application must be as transparent as possible regarding the CG knowledge formalism. The information should be presented to the end-user in a similar way as to the knowledge tree presentation in Freeplan.

For our system, we have identified the following needs:

- **Display of the list of possible corrective actions associated with a given descriptor:** an expert should be able to choose a cheese descriptor from a provided list, and then get all the corrective actions associated with it. For example, if in the presence of the descriptor corresponding to the cheese default "salt intake after 15 days is low", the expert wants to know the different corrective actions that can be used to solve the problem. Moreover, for a given corrective action, the expert wants to know which compensatory actions would be avoided if the corrective action is implemented.

- **Visualization of the path from a descriptor to the associated corrective actions:** for a given descriptor, the user must have access to the "*path*" that links the descriptor to the associated corrective actions in order to visualize the various intermediate explanatory mechanisms.

- **Impact of the implementation of a corrective action:** the expert wants to know the different descriptors that may be impacted by the choice of a particular corrective action.

We obviously see that these requirements are all quite easy to fulfil when we represent the knowledge as CG rules with CoGui but was impossible to take into account when the knowledge model by CTFC was represented using the mind map model of Freeplan.

An end-user application has been developed in order to fulfil to all these specifications. It permits to demonstrate that CoGui can be used as an internal knowledge base engine in a dedicated application software and there is no need for the final user to be familiar neither with CoGui nor with the conceptual graph formalism. Figure 7 and 8 shows 2 screen copy of the final end-user application. In this last figure, we see that we can reconstruct the tree structure mind map model of Freeplan from the CG rules defined in CoGui. Doing that is quite easy as it only requires to define the root of the tree (the descriptor) as a simple conceptual graph, and to saturate the knowledge base composed of CG rules to construct the tree.

Figure 9 shows a screen copy of the application which displays all the descriptors impacted by a given corrective action.

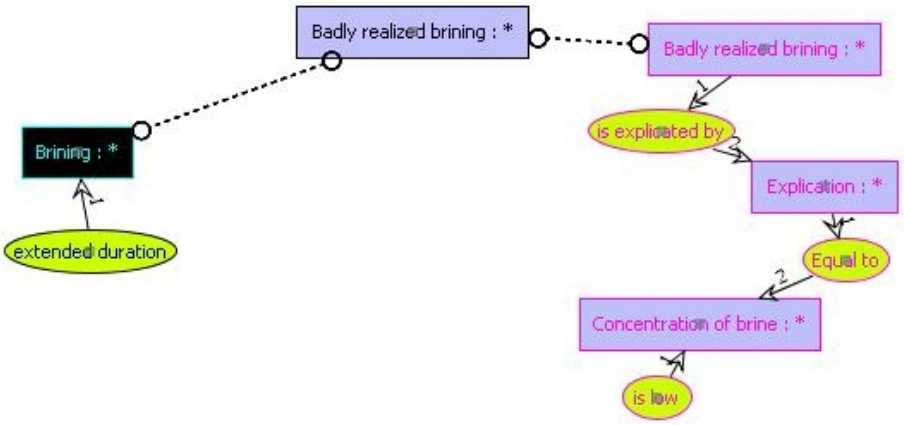

Fig. 6. Example of CG default rule translating in the CG model the Freeplan representation of a compensatory action shown in figure 3.

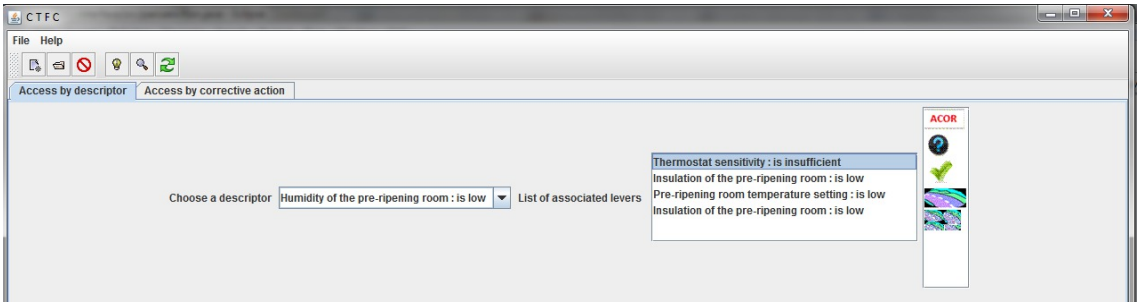

Fig. 7. Screen-shot of the final application showing how the corrective actions (4 actions in this example) of a given descriptor are presented.

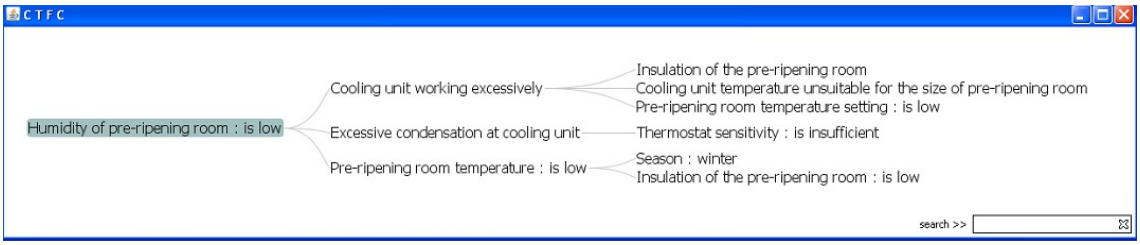

Fig. 8. Screen-shot of the final application showing a particular path in a knowledge tree

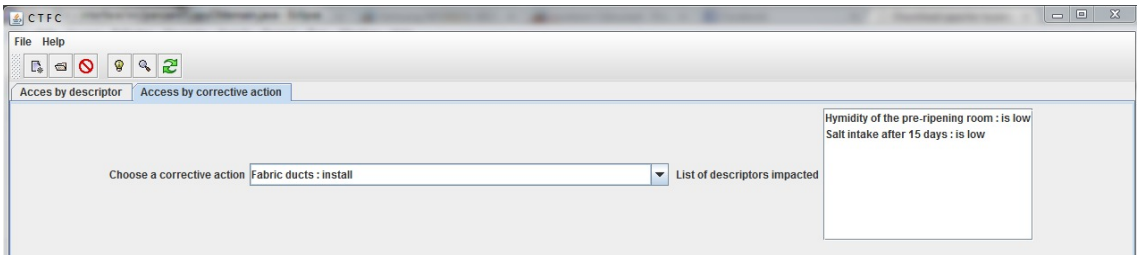

Fig. 9. Screen-shot of the final application showing all the descriptors impacted by a given corrective action

5 Conclusion

We have presented in this paper a real end-user application requiring knowledge representation and reasoning. The CTFC, which made the specifications of the application, needs a tool that will be used by technological experts of traditional

cheese production. We showed that on the one hand the application has a strong formal background based on conceptual graphs formalism and on the other hand that this formalism is relevant to model a complex knowledge information system. Especially, we have shown that the default conceptual rules are a good solution to manage complex knowledge as compensatory actions in the CTFC application. We have developed an end user application on top of the CoGui software which implements the CG model. This end-user application ensures that a non knowledge representation specialist can use it easily. Perspectives of this work will be to extend the CG model in order to be able to represent and take into account in the reasoning the reliability of the relationship *"is explained by"* between two explanatory mechanisms, which is another requirement of the CTFC application. Finally, another perspective of this work, suggested by the CTFC application, is to develop semi-automatic translation tools of semi-structure knowledge representation models, as Freeplan mind maps, into formal representation models as the CG model.

References

[Baget and Fortin, 2010] J.F. Baget and J. Fortin. Default conceptual graph rules, atomic negation and Tic–Tac–Toe. In *Proc. of ICCS'10*, volume 6208 of *LNAI*, pages 42–55, 2010.

[Baget *et al.*, 2009] J.F. Baget, M. Croitoru, J. Fortin, and R. Thomopoulos. Default conceptual graph rules : preliminary results for an agronomy application. In *Proc. of ICCS'09*, volume 5662 of *LNAI*, pages 86–99, 2009.

[Chein and Mugnier, 2009] Michel Chein and Marie-Laure Mugnier. *Graph-based Knowledge Representation and Reasoning. Computational Foundations of Conceptual Graphs*. Springer, Advanced Information and Knowledge Processing Series, London, 2009.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Salvat and Mugnier, 1996] E. Salvat and M-L. Mugnier. Sound and complete forward and backward chaining of graph rules. In *Proc of ICCS 1996: Conceptual Structures: Knowledge Representation as Interlingua*, volume 1115, 1996.

[Sowa, 1984] J. F. Sowa. *Conceptual Structures: Information Proc. in Mind and Machine*. Addison–Wesley, 1984.