Default Reasoning Implementation in CoGui

Patrice Buche, Jérôme Fortin, Alain Gutierrez

Default Reasoning Implementation in CoGui

Patrice Buche[1], Jérôme Fortin[2], and Alain Gutierrez[3]

[1] INRA IATE, LIRMM GraphIK, France
Patrice.Buche@supagro.inra.fr
[2] Université Montpellier II
IATE/LIRMM GraphIK, France
fortin@polytech.univ-montp2.fr
[3] CNRS LIRMM, France
alain.gutierrez@lirmm.fr

Abstract. This is an application paper in which we propose to present the actual implementation of default reasoning under conceptual graph formalism using CoGui. CoGui is a free graph-based visual tool, developed in Java, for building Conceptual Graph knowledge bases. We present the extension of this application to define and represent default CG rules (a CG-oriented subset of Reiter's default logics) and how to use these rules in skeptical or credulous reasoning.

1 Introduction

Default conceptual graph rules have been introduced in [1] in order to model expert knowledge, especially in agronomy applications. Default CG rules encode a subset of Reiter's default logic [2] and deal with knowledge of the form "if an hypothesis is proved true, a conclusion is generally true unless something that we know prevent us to infer this conclusion. Dealing with default is a kind of non monotonic reasoning because adding some new information to a knowledge base may prevent to apply some default.

The contribution of this paper is the presentation of the actual implementation of default reasoning under conceptual graph formalism using CoGui. CoGui is a free graph-based visual tool, developed in Java, for building Conceptual Graph knowledge bases.

The paper is organized as follows : Section 2 introduces classical notions of CG and CG rule. It gives some clues about their implementation in CoGui. Section 3 recalls basic definition of Reiter's defaults and introduces conceptual default rules. Section 4 is devoted to the presentation of the deduction algorithm using default CG rules.

2 Conceptuals Graphs in Cogui

In this section, we recall main notations and results required for the default CG rules used in this paper. In Section 2.2, we present the simple CGs of [3]. In Section 2.3, the CG rules of [4].

For each of these subsections we present how CoGui is designed to model these different notions. Figure 1 shows the structure of Cogui packages. CoGui is composed of four different layers associated with four different eclipse projects :

- the project *fr.lirmm.graphik.cogui.core* contains the model of knowledge representation, and all algorithms that permit to explore this model. It also contains tools to serialize data objects in xml, and to transpose a model in Datalog+ [5,6];
- the project *fr.lirmm.graphik.cogui.rdf* permits to make import and export from the CoGui internal model to RDF(S) and an OWL fragment [7];
- the project fr.lirmm.graphik.cogui.edit contains the user interface;
- the project *fr.lirmm.graphik.cogui.appli* is the upper layer of CoGui containing its entry point in the *CoGuiApplication* Class.

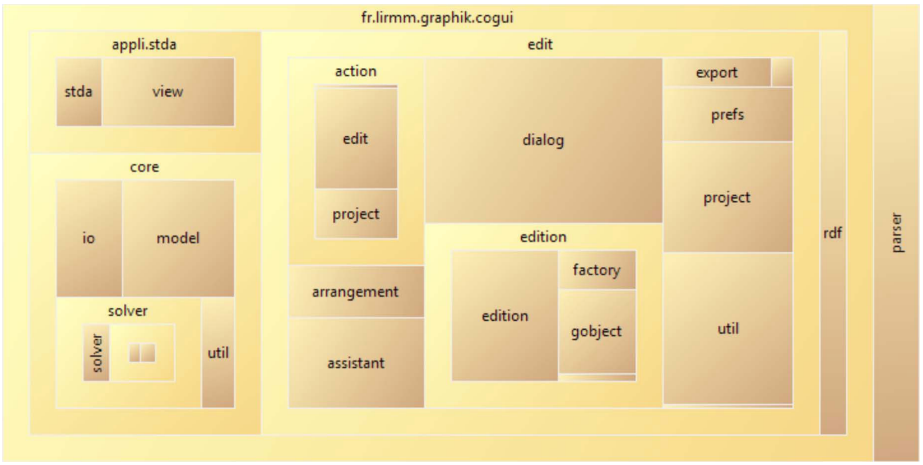

Fig. 1. Packages Structure of CoGui

2.1 Support

Syntax. With the simple CGs of [3], a knowledge base is structured into two objects: the vocabulary (also called support) encodes hierarchies of types, and the conceptual graphs (CGs) themselves represent entities and relations between them. Simple CGs are extended to handle *conjunctive types*, as done in [8].

Definition 1 (Vocabulary). *We call* vocabulary *a tuple* $\mathcal{V} = (\mathcal{C}, \mathcal{R} = (\mathcal{R}_1, \ldots, \mathcal{R}_k), \mathcal{M}_I, \mathcal{M}_G)$ *where* $\mathcal{C}$ *is a partially ordered set of* concept types *that contains a greatest element* $\top$*, each* $\mathcal{R}_i$ *is a partially ordered set of* relation types of arity i*,* $\mathcal{M}_I$ *is a set of* individual markers*, and* $\mathcal{M}_G$ *is a set of* generic markers*. Note that all these sets are pairwise disjoint, and that we denote all the partial orders by* $\leq$*.*

Definition 2 (Conjunctive types). *A* conjunctive concept type *over a vocabulary* $\mathcal{V}$ *is a set* $T = \{t_1, \ldots, t_p\}$ *(that we can note* $T = t_1 \sqcap \ldots \sqcap t_p$ *of arity* p*. If* $T = \{t_1, \ldots, t_p\}$ *and* $T' = \{t'_1, \ldots, t'_q\}$ *are two conjunctive concept types, then we also note* $T \leq T' \Leftrightarrow \forall t'_i \in T', \exists t_j \in T$ *such that* $t_j \leq t'_i$*.*

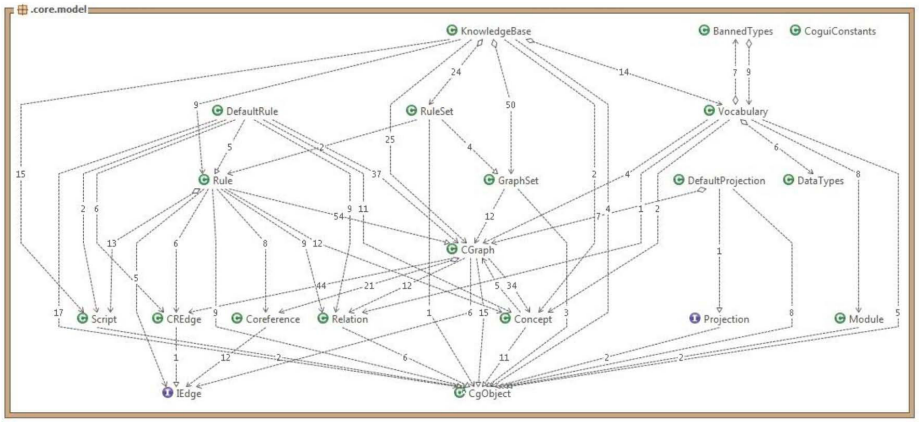

Fig. 2. Core Model of CoGui

Implementation. Figure 2 presents the core model of CoGui. All graphs are created in CoGui in a structure that use the JGraphT 0.8 Java Library [9]. Concept node hierarchy, relation node hierarchy are stored as oriented graphs. A list of individual markers associated with their privilege types is maintained. Figure 3 shows a concept type hierarchy in CoGui.

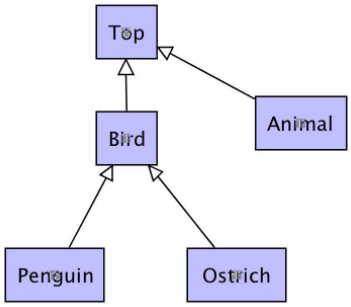

Fig. 3. A concept node hierarchy represented in CoGui

2.2 Simple Conceptual Graphs

A simple conceptual graph, also often called fact is defined as follows :

Definition 3 (Simple CGs). *A* simple CG *is a tuple* $G = (C, R, \gamma, \lambda)$ *where* C *and* R *are two finite disjoint sets (*concept nodes *and* relations*) and* γ *and* λ *two mappings:*

- $\gamma : R \rightarrow C^+$ *associates to each relation a tuple of concept nodes* $\gamma(r) = (c_1, \ldots, c_k)$ *called the* arguments *of* r*,* $\gamma_i(r) = c_i$ *is its* i*th argument and* $\mathsf{degree}(r) = k$*.*
- λ *maps each concept node and each relation to its* label*. If* $c \in C$ *is a concept node, then* $\lambda(c) = (\mathsf{type}(c), \mathsf{marker}(c))$ *where* $\mathsf{type}(c)$ *is a conjunctive concept type and*

marker(c) is either an individual marker of $\mathcal{M}_I$ or a generic marker of $\mathcal{M}_G$. If $r \in R$ is a relation and degree$(r) = k$, then $\lambda(r)$ is a relation type of arity k.

A simple CG is said to be normal *if all its concept nodes have different markers. Any simple CG G can be put into its* normal form *nf(G) in linear time.*

Semantics. We associate a first order logics (FOL) formula $\Phi(\mathcal{V})$ to a vocabulary $\mathcal{V}$, and a FOL formula $\Phi(G)$ to a simple CG G. These formulae are obtained as follows:

Interpretation of a Vocabulary. Let $\mathcal{V} = (\mathcal{C}, \mathcal{R} = (\mathcal{R}_1, \ldots, \mathcal{R}_k), \mathcal{M}_I, \mathcal{M}_G)$ be a vocabulary. We can consider each concept type of $\mathcal{C}$ as a unary predicate name, each relation type of $\mathcal{R}_i$ as a predicate name of arity i, each individual marker of $\mathcal{M}_I$ as a constant, and each generic marker of $\mathcal{M}_G$ as a variable. For each pair (t, t') of concept types of $\mathcal{C}$ such that $t \leq t'$, we have a formula $\phi((t, t')) = \forall x(t(x) \rightarrow t'(x))$. For each pair (t, t') of relation types of arity i such that $t \leq t'$, we have a formula $\phi((t, t')) = \forall x_1 \ldots \forall x_i(t(x_1, \ldots, x_i) \rightarrow t'(x_1, \ldots, x_i))$. Then the FOL interpretation $\Phi(\mathcal{V})$ of the vocabulary $\mathcal{V}$ is the conjunction of the formulae $\phi((t, t'))$, for every pair of types (t, t') such that $t' < t'$.

Interpretation of a Simple CG. Let $G = (C, R, \gamma, \lambda)$ be a simple CG. We can associate a formula to each concept node and relation of G: if $c \in C$ and type$(c) = t_1 \sqcap \ldots \sqcap t_k$, then $\phi(c) = t_1($marker$(c)) \wedge \ldots \wedge t_k($marker$(c))$; and if $r \in R$, with $\gamma(r) = (c_1, \ldots, c_q)$ and $\lambda(r) = t$, then $\phi(r) = t($marker$(c_1), \ldots,$ marker$(c_q))$. We note $\phi(G) = \bigwedge_{c \in C} \phi(c) \wedge \bigwedge_{r \in R} \phi(r)$. The FOL formula $\Phi(G)$ associated with a simple CG is the existential closure of the formula $\phi(G)$.

Implementation. Simple CG are of course stored as graphs in CoGui. Each node of a Fact graph has a given label. Depending on the type of the considered node, two different types of label are used. For relation nodes, it is enough to label the node by its relation name. For concept nodes, type and marker are needed. A marker can be an individual marker or a generic marker. In a given fact Graph, each generic marker must have a unique name. In order to perform reasoning, one extra attribute is given for each node (concept and relation nodes). This attribute permit to identify uniquely any node of a CoGui project, stored in any kind of graphs (Support, Simple CG, Rules...).

2.3 Conceptual Graph Rules

CG rules form an extension of CGs with knowledge of form "if *hypothesis* then *conclusion*". Introduced in [10], they have been further formalized and studied in [4].

Syntax. A usual way to define CG rules is to establish *co-reference relations* between the hypothesis and the conclusion. We have used here instead *named generic markers*: generic nodes with same marker representing the same entity.

Definition 4 (CG rule). *A* conceptual graph rule*, defined on a vocabulary $\mathcal{V}$, is a pair $R = (H, C)$ where H and C are two simple CGs, respectively called the hypothesis and the conclusion of the rule.*

Semantics. We present here the usual Φ semantics of a CG rule, and recall the equivalent semantics Φ^f using function symbols introduced in [11]. This equivalent semantics makes for an easier definition of default rules semantics: since default rules are composed of different formulas, we cannot rely upon the quantifier's scope to link variables, and thus have to link them through functional terms.

Let $R = (H, C)$ be a CG rule. Then the FOL interpretation of R is the formula $\Phi(R) = \forall x_1 \dots \forall x_k(\phi(H) \rightarrow (\exists y_1 \dots \exists y_q \phi(C)))$, where $x_1, \dots, x_k$ are all the variables appearing in $\phi(H)$ and $y_1, \dots, y_q$ are all the variables appearing in $\phi(C)$ but not in $\phi(H)$. If $\mathcal{R}$ is a set of CG rules, then $\Phi(R) = \bigwedge_{R \in \mathcal{R}} \Phi(R)$.

As an alternate semantics, let G be a simple CG and X be a set of nodes. We denote by $F = \{f_1, \dots, f_p\}$ the set of variables associated with generic markers that appear both in G and in X. The formula $\phi^f_X(G)$ is obtained from the formula $\phi(G)$ by replacing each variable y appearing in $\phi(G)$ but not in F by a functional term $f^y_G(f_1, \dots, f_p)$. Then the FOL interpretation (with function symbols) of a rule $R = (H, C)$ is the formula $\Phi^f(R) = \forall x_1 \dots \forall x_k(\phi(H) \rightarrow \phi^f_X(C))$ where X is the set of nodes appearing in H. If $\mathcal{R}$ is a set of CG rules, then $\Phi^f(\mathcal{R}) = \bigwedge_{R \in \mathcal{R}} \Phi^f(R)$. Note that this semantics translates in a straightforward way the skolemisation of existentially quantified variables.

Implementation. A rule is stored in CoGui as a single bi-colored graph with two connected components :

- one component stands for the hypothesis of the rule and is tagged by one color,
- the second component models the conclusion of the rule, and is tagged with the other color.

As some nodes of the hypothesis need to correspond to some nodes of the conclusion, a list of coreference links is needed for each rule. The justification and the conclusion must be two connected component. This choice have been made in CoGui in order to optimize some graph operations.

Remark that it wouldn't be sufficient to model a rule by a single bi-colored connected graph, because this representation would not permit to specialize (according to the concept type hierarchy) in the conclusion any node of the Hypothesis. For example, one can express that each Animal that flies is a Bird. This rule is shown in Figure 4.

Computing Deduction. We present here the forward chaining mechanism used to compute deduction with CG rules. In general, this is an undecidable problem. The reader can refer to [12] for an up-to-date cartography of decidable subclasses of the problem.

Definition 5 (Application of a rule). *Let G be a simple CG, $R = (H, C)$ be a rule, and π be a homomorphism from H to G. The* application of R on G according to π *produces a normal simple CG $\alpha(G, R, \pi) = \mathbf{nf}(G \oplus C_\pi)$ where:*

- *C_π is a simple CG obtained as follows from a copy of C: (i) associate to each generic marker x that appears in C but not in H a new distinct generic marker $\sigma(x)$; (ii) for every generic concept node c of C whose marker x does not appear*

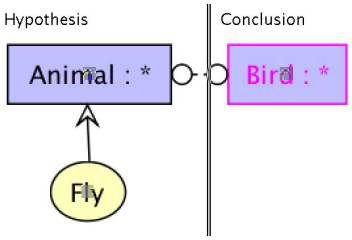

Fig. 4. A rule that specialize a node of the hypothesis

in H, replace marker(c) *with $\sigma(x)$; and (iii) for every generic concept node c of C, if* marker(c) *also appears in H, then replace* marker(c) *with* marker$(\pi(c))$.
- *the operator $\oplus$ generates the* disjoint union *of two simple CGs G and G': it is the simple CG whose drawing is the juxtaposition of the drawings of G and G'.*

Theorem 1. *Let G and Q be two simple CGs, and $\mathcal{R}$ be a set of CG rules, all defined on a vocabulary $\mathcal{V}$. Then the following assertions are equivalent:*

- $\Phi(\mathcal{V}), \Phi(G), \Phi(\mathcal{R}) \vdash \Phi(Q)$
- $\Phi(\mathcal{V}), \Phi(G), \Phi^f(\mathcal{R}) \vdash \Phi(Q)$
- *there exists a sequence $G_0 =$ nf$(G), G_1, \ldots, G_n$ of simple CGs such that:*
  (i) $\forall 1 \leq i \leq n$, there is a rule $R = (H, C) \in \mathcal{R}$ and a homomorphism π of H to G_{i-1} such that $G_i = \alpha(G_{i-1}, R, \pi)$;
  and (ii) there is a homomorphism from Q to G_n.

Note that the forward chaining algorithm that relies upon the above characterization is ensured to stop when the set of rules involved is *range restricted*, *i.e.* their logical semantics Φ does not contain any existentially quantified variable in the conclusion (see [13] for more details about tractable cases).

The "functional semantics" can provide us with an alternate rule application mechanism α^f. Let us begin by "*freezing*" the graph G, *e.g.* by replacing each occurrence of a generic marker by a distinct individual marker that not appears yet in G. Then, when applying a rule R on G (or a graph derived from G) according to a projection π, we consider the formula $\Phi^f(R)$ associated with R. Should the application of $R = (H, C)$ produce a new generic node c from the copy of a generic node having marker y, we consider the functional term $f^y_C(x_1, \ldots, x_k)$ associated to the variable y. Then the marker of c becomes $f^y_C(\pi(x_1), \ldots, \pi(x_k))$. Thanks to the previous theorem, α^f makes for an equivalent forward chaining mechanism, that has an added interest. It allows to have a functional constant identifying every concept node generated in the derivation. This feature will be used to explain default rules reasonings in an easier way than in [1].

3 Default Rules

3.1 Default CG Rules

A Brief Introduction. Let us recall some basic definitions of Reiter's default logics. For a more precise description and examples, the reader should refer to [14,2].

Definition 6 (Reiter's default logic). *A Reiter's* default theory *is a pair* (Δ, W) *where* W *is a set of FOL formulae and* Δ *is a set of* defaults *of form* $\delta = \frac{\alpha(\overrightarrow{x}):\beta_1(\overrightarrow{x}),\cdots,\beta_n(\overrightarrow{x})}{\gamma(\overrightarrow{x})}$, $n \geq 0$, *where* $\overrightarrow{x} = (x_1, \cdots, x_k)$ *is a tuple of variables,* $\alpha(\overrightarrow{x})$, $\beta_i(\overrightarrow{x})$ *and* $\gamma(\overrightarrow{x})$ *are FOL formulae for which each free variable is in* $\overrightarrow{x}$.

The intuitive meaning of a default δ is "For all individuals $(x_1, \cdots, x_k)$, if $\alpha(\overrightarrow{x})$ is believed and each of $\beta_1(\overrightarrow{x}), \cdots, \beta_n(\overrightarrow{x})$ can be consistently believed, then one is allowed to believe $\gamma(\overrightarrow{x})$". $\alpha(\overrightarrow{x})$ is called the *prerequisite*, $\beta_i(\overrightarrow{x})$ are called the *justifications* and $\gamma(\overrightarrow{x})$ is called the *consequent*. A default is said *closed* if $\alpha(\overrightarrow{x})$, $\beta_i(\overrightarrow{x})$ and $\gamma(\overrightarrow{x})$ are all closed FOL formulae.

Intuitively, an *extension* of a default theory (Δ, W) is a set of formulae that can be obtained from (Δ, W) while being consistently believed. More formally, an extension E of (Δ, W) is a minimal deductively closed set of formulae containing W such that for any $\frac{\alpha:\beta}{\gamma} \in \Delta$, if $\alpha \in E$ and $\neg\beta \notin E$, then $\gamma \in E$. The following theorem provides an equivalent characterization of extensions that we use here as a formal definition.

Theorem 2 (Extension). *Let* (Δ, W) *be a closed default theory and* E *be a set of closed FOL formulae. We inductively define* $E_0 = W$ *and for all* $i \geq 0$, $E_{i+1} = Th(E_i) \cup \{\gamma \mid \frac{\alpha:\beta_1\cdots,\beta_n}{\gamma} \in \Delta, \alpha \in E_i$ *and* $\neg\beta_1, \cdots, \neg\beta_n \notin E\}$, *where* $Th(E_i)$ *is the deductive closure of* E_i. *Then* E *is an extension of* (Δ, W) *iff* $E = \cup_{i=0}^{\infty} E_i$.

Note that this characterization is not effective for computational purposes since both E_i and $E = \cup_{i=0}^{\infty} E_i$ are required for computing E_{i+1}.

The following reasoning tasks come with Reiter's Default Logic:

- EXTENSION: Given a default theory (Δ, W), does it have an extension?
- SKEPTICAL DEDUCTION: Given a default theory (Δ, W) and a formula Q, does Q belong to all extensions of (Δ, W)? In this case we note $(\Delta, W) \vdash_S Q$.
- CREDULOUS DEDUCTION: Given a default theory (Δ, W) and a formula Q, does Q belong to an extension of (Δ, W)? In this case we note $(\Delta, W) \vdash_C Q$?

Syntax of Default CGs

Definition 7 (Default CGs). *A* default CG*, defined on a vocabulary* $\mathcal{V}$*, is a tuple* $D = (H, C, J_1, \ldots, J_k)$ *where* H*,* C*,* $J_1, \ldots$*, and* J_k *are simple CGs respectively called the hypothesis, conclusion, and justifications of the default.*

Semantics. The semantics of a default CG $D = (H, C, J_1, \ldots, J_k)$ is expressed by a closed default $\Delta(D)$ in Reiter's default logics.

$$\Delta(D) = \frac{\phi(H) \ : \ \neg\phi^f_X(C), \phi^f_{X\cup Y}(J_1), \ldots, \phi^f_{X\cup Y}(J_k)}{\phi^f_X(C)}$$

where X is the set of nodes of the hypothesis H and Y is the set of nodes of the conclusion C. If $D = (H, C, J_1, \ldots, J_k)$ is a default, we note *std*$(D) = (H, C)$ its *standard part*, which is a CG rule.

Implementation. A default rule is stored as a single multi–colored graph. As for classical rules, one color is needed for the hypothesis, one for the conclusion, and n other colors are used to tag each of the n justifications. Figure 4 shows a default rule that means "unless a birds is known as a penguin or an ostrich, we can suppose that it flies".

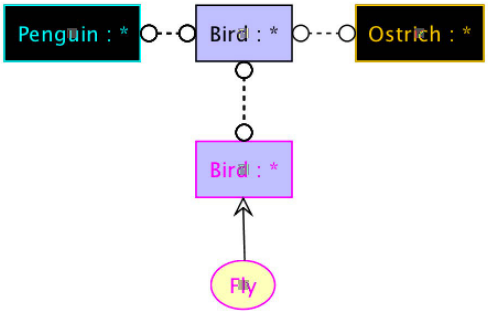

Fig. 5. Representation of a default rule

4 Default Reasoning with Default CG Rules in CoGui

In this section, we explain the algorithm that permits to compute deduction in the alternate derivation mechanism α^f introduced in [11]. This mechanism is introduced for an easier description of the sound and complete reasoning mechanism of [1]. Moreover, this alternate derivation mechanism permits to get a strait–forward way to implement default reasoning in CoGui. Let G and Q be two simple CGs, $\mathcal{R}$ be a set of CG rules, and $\mathcal{D}$ be a set of default CG rules, all defined over a vocabulary V. A node of the default derivation tree DDT($\mathcal{K}$) of the knowledge base $\mathcal{K} = ((V, G, \mathcal{R}), \mathcal{D})$ is always labelled by a simple CG called *fact* and a set of simple CGs called *constraints*. A node of DDT($\mathcal{K}$) is said *valid* if there is no homomorphism of one of its constraints or the constraints labelling one of its ancestors into its fact. Let us now inductively define the tree DDT($\mathcal{K}$):

- its root is a node whose fact is G and whose constraint set is empty;
- if x is a valid node of DDT($\mathcal{K}$) labelled by a fact F and constraints $\mathcal{C}$, then
  for every rule D in $\mathcal{D}$, for every homomorphism π of the hypothesis of D into a simple CG F' $\mathcal{R}$-derived from F,
  x admits a successor whose fact is the fact $\alpha^f(F', \mathit{std}(D), \pi)$, and whose constraints are the $\pi(J_i)$ iff that successor is valid.

Theorem 3. *Let G and Q be two simple CGs, $\mathcal{R}$ be a set of CG rules, and $\mathcal{D}$ be a set of default CG rules, all defined over a vocabulary V. Then $\Phi(Q)$ belongs to an extension of the Reiter's default theory $(\{\Phi(V), \Phi(G), \Phi(\mathcal{R})\}, \Delta(\mathcal{D}))$ iff there exists a node x of DDT$((V, G, \mathcal{R}), \mathcal{D})$ labelled by a fact F such that $\Phi(V), \Phi(F), \Phi(\mathcal{R}) \vdash \Phi(Q)$.*

Intuitively, this result [1] states that the leaves of $DDT(\mathcal{K})$ encode extensions of the default. What is interesting in this characterization is that: 1) though our default CGs are not normal defaults in Reiter's sense, they share the same important property: every default theory admits an extension; and 2) if an answer to a query is found in any node of the default derivation tree, the same answer will still be found in any of its successors.

4.1 Implementation

In order to compute deduction in CoGui, a new class `CoguiDefaultRuleApplyer` has been created. This class contains a list of Classical rules, a list of default rules and a stack of j–constraints. This stack of j–constraints is used to store each justification that has been "supposed" before applying a default. In practice, a j–constraint is stored as any conceptual graph. Contrary to the classical constraints in conceptual graphs [13], j–constraints need to be attached to the fact on a precise place. The semantic of a classical constraint is the following : if a constraint C can be projected in a graph G, then the knowledge base $\mathcal{K} = (G, C)$ is inconsistent. So if a constraint contains some generic marker, then the constraint can be projected in any place of the graph G.

When dealing with default rules, j–constraints have a different semantics. This is due to the fact that a given justification is precisely linked to an hypothesis of a default.

Let us consider the default $D = (Bird(x), fly(x), Penguin(x))$. This default means that if we find a bird in a database, then we can deduce that this bird flies unless we know that it is a penguin. So if in a fact graph G, we find a concept node $Bird(y)$, associated with the generic marker y, we can try to apply the default D. After adding the conclusion of the default, we have to keep in mind for future deduction (with classical rules or default rules) that the bird y (and only this one) can not be considered in any way as a penguin. It is not possible to use the classical constraint $Penguin(x)$ because it would mean that any individual can be a penguin, which is obviously not what we want to model. So the difference between a classical constraint and a j–constraint is just in its semantics : When adding the j–constraint $Penguin(y)$, it just says that any node that is labeled by the generic marker y (and only this particular generic marker) can represent a penguin.

The main algorithm implemented in CoGui to find extensions of a given knowledge base is based on a backtrack algorithm. We begin with an initial fact graph G, we choose one default and one projection, we apply this default on the projection and run a saturation round by classical rules. If no more default can be applied, no constraint is violated and no default rule is active, the current state is an extension. Otherwise, we continue our exploration of the default derivation tree. Once an extension is found, the current state is saved as a new extension, and a backtrack is operated. This backtrack is possible with the help of the reverse actions implemented in CoGui.

$createJConstaint(J_i, \pi)$ is a procedure that creates a j–constraint by renaming each generic marker of J_i by the corresponding label of the projection π if the considered node is linked with the hypothesis in D, and with a new generic marker otherwise. The procedure $createAddConclusionAction$ permits to create a single defeasible action that permits to add a graph (conclusion of the default) to another graph according to a certain projection.

Algorithm 1. Main deduction algorithm

Data: $\mathcal{K} = ((V, G, \mathcal{R}), \mathcal{D})$
Result: The set of extensions of $\mathcal{K}$
begin
  /* Initialisation of global variables */
  $currentGraph \longleftarrow G$;
  $constraintStack \longleftarrow \emptyset$;
  $currentStackReverseActions \longleftarrow \emptyset$;
  $stackReverseActions \longleftarrow \emptyset$;
  $extensionsSet \longleftarrow \emptyset$;
  $applyMaster()$;

Algorithm 2. applyMaster

Data: $\mathcal{K}$ (in), currentGraph (in/out); constraintStack (in/out); currentStackReverseActions (in/out); stackReverseActions (in/out)
begin
  $ruleApplied \longleftarrow false$;
  for $D = (H, C, J_1, \cdots, J_n) \in \mathcal{D}$ **do**
    for π *projection of H in currentGraph* **do**
      $applied \longleftarrow applyDefault(D, \pi)$;
      if *applied* **then**
        $ruleApplied \longleftarrow true$;
        $A \longleftarrow createSaturateAction(currentGraph, \mathcal{R})$;
        applyAction(A,currentGraph);
        stackReverseAction.put(reverse(A));
        $applyMaster()$;
        $A \longleftarrow stackReverseAction.pop(A)$;
        applyAction(A);
  if $not(ruleApplied)$ **then**
    extensionSet.add(New copy of current graph);

Algorithm 3. applyDefault

```
Data: Default D, projection π (and currentGraph; constraintStack;
      currentStackReverseActions ; stackReverseActions as (global variables)
Result: Boolean : true iff D has been applied
begin
    /* Checking justifications                                    */
    applicable ⟵ true;
    for J_i justification of D do
        if π can be extended to project J_i then
            applicable ⟵ false;
    if applicable then
        for J_i justification of D do
            C ⟵ createJConstraint(D, J_i, π);
            constraintStack.put(C);
        A ⟵ createAddConclusionAction(currentGraph, C, π);
        applyAction(A,currentGraph);
        stackReverseActions.put(reverse(A));
        return true;
    else
        return false;
```

5 Conclusion

It is now possible to model default CG rule with CoGui and to make default reasoning. Default CG rules model a subset of classical Reiter's default rule, in which the conclusion is always seen as an implicit justification. The reasoning implementation permit to compute all extensions of a given knowledge base. The construction of the extension is done by exploring the default deviation tree by a deep first search, so there is no need to store all the derivation trees to compute deduction.

From this list of extensions it is simple to check if a fact is skeptically or credulously entailed by the knowledge base : CoGui permits to manipulate any CoGui object with a devoted script language. Scripts are object stored in CoGui that permit to creates easily automatic tasks that manipulates any CoGui objects. Scripts are develloped using a Java-like langage. This functionality has not been described in this paper in order to focus on the default reasoning, but scripts can be easily used to perform skeptical and credulous deduction. One further work is to directly integrate this functionality to the CoGui native possibilities.

Default rules in CoGui are now used for some applications projects. We use it in particular to model expert knowledge in some agronomy applications [15]. In this domain, an application called Capex (for expert capitalization) has been developed over CoGui and its default reasoning possibilities.

References

1. Baget, J.-F., Croitoru, M., Fortin, J., Thomopoulos, R.: Default conceptual graph rules: Preliminary results for an agronomy application. In: Rudolph, S., Dau, F., Kuznetsov, S.O. (eds.) ICCS 2009. LNCS (LNAI), vol. 5662, pp. 86–99. Springer, Heidelberg (2009)
2. Reiter, R.: A logic for default reasoning. Artificial Intelligence 13, 81–132 (1980)
3. Sowa, J.F.: Conceptual graphs for a database interface. IBM Journal of Research and Development 20(4), 336–357 (1976)
4. Salvat, E., Mugnier, M.L.: Sound and complete forward and backward chaining of graph rules. In: Eklund, P., Ellis, G., Mann, G. (eds.) ICCS 1996. LNCS, vol. 1115, pp. 248–262. Springer, Heidelberg (1996)
5. Calì, A., Gottlob, G., Lukasiewicz, T., Pieris, A.: Datalog+/-: A family of languages for ontology querying. In: de Moor, O., Gottlob, G., Furche, T., Sellers, A. (eds.) Datalog 2010. LNCS, vol. 6702, pp. 351–368. Springer, Heidelberg (2011)
6. Mugnier, M.-L.: Ontological Query Answering with Existential Rules. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 2–23. Springer, Heidelberg (2011)
7. Baget, J.-F., Croitoru, M., Gutierrez, A., Leclère, M., Mugnier, M.-L.: Translations between RDF(S) and conceptual graphs. In: Croitoru, M., Ferré, S., Lukose, D. (eds.) ICCS 2010. LNCS, vol. 6208, pp. 28–41. Springer, Heidelberg (2010)
8. Baget, J.-F.: Simple Conceptual Graphs Revisited: Hypergraphs and Conjunctive Types for Efficient Projection Algorithms. In: Ganter, B., de Moor, A., Lex, W. (eds.) ICCS 2003. LNCS (LNAI), vol. 2746, pp. 229–242. Springer, Heidelberg (2003)
9. Naveh, B., Contributors: Jgrapht (March 2014), `http://jgrapht.org/`
10. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison–Wesley (1984)
11. Baget, J.-F., Fortin, J.: Default conceptual graph rules, atomic negation and tic-tac-toe. In: Croitoru, M., Ferré, S., Lukose, D. (eds.) ICCS 2010. LNCS, vol. 6208, pp. 42–55. Springer, Heidelberg (2010)
12. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: Extending decidable cases for rules with existential variables. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, pp. 677–682 (2009)
13. Chein, M., Mugnier, M.L.: Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs, 1st edn. Springer Publishing Company, Incorporated (2008)
14. Brewka, G., Eiter, T.: Prioritizing default logic: Abridged report. In: Festschrift on the Occasion of Prof. Dr. W. Bibel's 60th Birthday. Kluwer (1999)
15. Buche, P., Cucheval, V., Diattara, A., Fortin, J., Gutierrez, A.: Implementation of a knowledge representation and reasoning tool using default rules for a decision support system in agronomy applications. In: Croitoru, M., Rudolph, S., Woltran, S., Gonzales, C. (eds.) GKR 2013. LNCS (LNAI), vol. 8323, pp. 1–12. Springer, Heidelberg (2014)