Default Conceptual Graph Rules, Atomic Negation and Tic-Tac-Toe

Jean-François Baget[1,2] and Jérôme Fortin[3,2]

[1] INRIA Sophia Antipolis, 2004 Route des Lucioles 06902 Sophia Antipolis, France
baget@lirmm.fr
[2] LIRMM (CNRS & Université Montpellier II), F–34392 Montpellier Cedex 5, France
[3] IATE, UMR1208, F–34060 Montpellier Cedex 1, France
jerome.fortin@supagro.inra.fr

Abstract. In this paper, we explore the expressivity of default CG rules (a CG-oriented subset of Reiter's default logics) through two applications. In the first one, we show that default CG rules provide a unifying framework for CG rules as well as polarized CGs (CGs with atomic negation). This framework allows us to study decidable subclasses of a new language mixing CG rules with atomic negation. In the second application, we use default CG rules as a formalism to model a game, an application seldom explored by the CG community. This model puts into light the conciseness provided by defaults, as well as the possibilities they offer to achieve efficient reasonings.

1 Introduction

Default CG Rules have been introduced in [1] as a requirement for an agronomy application. These rules encode a subset of Reiter's default logics [2], with knowledge of form "if *hypothesis*, then *conclusion* is generally true, unless drawing that conclusion leads to contradict one of the *justifications* of the default", where the hypothesis, conclusion, and justifications are simple conceptual graphs. These default rules have two main interest. First they admit a natural graphical representation that extends the representation of CG rules, and thus inherits from the user-friendly charateristics of the CG formalism. Moreover, they form the corresponding fragment in Reiter's default logic of the rule fragment in FOL. We believe that the decidability arguments of the rule fragment [3,4] will find their counterpart in default rules. In this paper, we focus on two applications to study the expressiveness of that language.

The first uses default CG rules to encode *atomic negation* into conceptual graphs. Indeed, important extensions of Sowa's simple conceptual graphs [5] have concerned CG rules [6] and polarized graphs [7] (*i.e.* simple graphs enriched with atomic negation). There has been for now no unifying framework for these two extensions that relies upon graph-based reasonings to compute deduction (the work of [7], for example, that extends conceptual graphs to handle the whole first-order logics, mixes graph-based reasonings with a tableaux mechanism). We show here that default CG rules provide such a unifying framework, and put that framework to use to begin to explore decidable subclasses of a CG language that enriches CG rules with atomic negation.

M. Croitoru, S. Ferré, and D. Lukose (Eds.): ICCS 2010, LNAI 6208, pp. 42–55, 2010.

In the second application, we have chosen to present an example using default CG rules in a field seldom explored by the CG community: *games*. Our motivation was twofold: 1) we wanted a new type of application that forced us to think "out of the box", as was done with the ICCS Sisyphus-I initiative [8], and 2) we wanted a motivating example for Master's students in knowledge representation. This model has put into light two main interests of default CG rules: 1) they mix the intuitive graphical representation of CGs with the conciseness brought by Reiter's defaults, and 2) though default CG rules form a more complex language than rules, they offer mechanisms that allow for more efficient reasonings.

2 From Simple CGs to Default CGs

In this first section, we recall main notations and results required for the default CG rules used in this paper. In SECT. 2.1, we present the simple CGs of [5], in SECT. 2.2, the CG rules of [6], and finally in SECT. 2.3, the default CG rules of [1].

2.1 Simple Conceptual Graphs

Syntax With the simple CGs of [5], 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. Our simple CGs use *named generic markers*, and are extended to handle *conjunctive types*, as done in [9].

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 concept types of arity* k*. 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* $t_j \leq t'_i$*.*

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:*

- $\gamma : R \to C^+$ *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* degree$(r) = k$*.*
- λ *maps each concept node and each relation to its* label*. If* $c \in C$ *is a concept node, then* $\lambda(c) =$ (type(c), marker(c)) *where* 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 equivalent* 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 $\mathcal{V}$ is the conjunction of all $\phi((t, t'))$, for every pair (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 $\mathit{type}(c) = t_1 \sqcap \ldots \sqcap t_k$, then $\phi(c) = t_1(\mathit{marker}(c)) \wedge \ldots \wedge t_k(\mathit{marker}(c))$; and if $r \in R$, with $\gamma(r) = (c_1, \ldots, c_q)$ and $\lambda(r) = t$, then $\phi(r) = t(\mathit{marker}(c_1), \ldots, \mathit{marker}(c_q))$. 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 $\phi(G)$.

Computing Deduction. Computing a graph homomorphism (known as projection in CGs) is a sound and complete algorithm for deduction of the associated FOL formulae. HOMOMORPHISM is an NP-complete problem, that becomes polynomial when the question graph is a tree (see [9,10] for more polynomial subclasses).

Definition 4 (Homomorphism). *Let $F = (C_F, R_F, \gamma_F, \lambda_F)$ and $Q = (C_Q, R_Q, \gamma_Q, \lambda_Q)$ be two simple CGs defined over a vocabulary $\mathcal{V}$. A* homomorphism *from Q to F is a mapping $\pi : C_Q \rightarrow C_F$ such that:*

- *if $c \in C_Q$ is individual, then $\mathit{marker}(c) = \mathit{marker}(\pi(c))$;*
- *if c and c' are two generic concept nodes with same markers, then $\pi(c) = \pi(c')$;*
- *$\forall c \in C_Q$, $\mathit{type}(\pi(c)) \leq \mathit{type}(c)$;*
- *$\forall r \in R_Q$, $\exists r' \in R_F$ such that $\lambda(r') \leq \lambda(r)$ and $\gamma(r') = \pi(\gamma(r))$.*

Theorem 1. *Let F and Q be two simple CGs defined over a vocabulary $\mathcal{V}$. Then $\Phi(\mathcal{V}), \Phi(F) \vdash \Phi(Q)$ iff there exists a homomorphism from Q to $\mathit{nf}(F)$.*

2.2 Conceptual Graph Rules

CG rules form an extension of CGs with knowledge of form "if *hypothesis* then *conclusion*". Introduced in [11], they have been further formalized and studied in [6,3].

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

Definition 5 (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 introduce an equivalent semantics Φ^f using function symbols. Basically, Φ^f translates in a straightforward way the skolemisation of existentially quantified variables. 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 $\Phi^f(\mathcal{R}) = \bigwedge_{R \in \mathcal{R}} \Phi^f(R)$.

The translations of the rule $R = (H, C)$ where $H = [Human : *x]$ and $C = [Human : *x] < -(isParent) < -[Human : *y]$ (in linear form, meaning that every human has a human parent) are:
$\Phi(R) = \forall x, (Human(x) \rightarrow \exists y(Human(y) \wedge isParent(y, x)))$
$\Phi(R)^f = \forall x, (Human(x) \rightarrow Human(f^y_C(x)) \wedge isParent(f^y_C(x), x))$

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 [3] for an up-to-date cartography of decidable subclasses of the problem.

Definition 6 (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 $\alpha(G, R, \pi) =$ 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 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 2. *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$(G), G_1, \dots, G_n$ 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.

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. Then, when applying a rule R on G (or a graph derived from G) according to a projection π, 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, consider the functional term $f_C^y(x_1, \ldots, x_k)$ associated to the variable y. Then the marker of c becomes $f_C^y(\pi(x_1), \ldots, \pi(x_k))$. 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].

2.3 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 [12,2].

Definition 7 (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 $a \in E$ 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 3 (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* $i \geq 0$, $E_{i+1} = Th(E_i) \cup \{\gamma \mid \frac{\alpha:\beta_1\cdots,\beta_n}{\gamma} \in \Delta, \alpha \in E_i \text{ and } \neg\beta_1, \cdots, \neg\beta_n \notin E\}$, *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}.

Some problems that are to be addressed in Reiter's default logics are the following:
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 $(\Delta, W) \vdash_C Q$?

Definition 8 (Default CGs (Syntax)). *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.

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

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.

Computing Deduction. Our alternate derivation mechanism α^f makes for an easier description of the sound and complete reasoning mechanism of [1]. 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 $\alpha^f(F', \textit{std}(D), \pi)$, and whose constraints are the $\pi(J_i)$ iff that successor is valid.

Theorem 4. *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.

3 Using Default CG Rules for Atomic Negation

Neither simple CGs nor CG rules can handle negation, even in its basic atomic form. Indeed, their reasonings do not support branching, necessary in tableaux-like mechanisms as soon as negation or disjunction is involved. We show that default CG rules can handle as well the semantics of polarized graphs (simple CGs enriched with atomic negation [13]) as the semantics of their extension to polarized graphs rules.

3.1 Polarized Graphs

Simply put, polarized graphs [13] form an extension of simple CGs in which all types are polarized: a type with a positive polarization is translated into a positive atom in FOL, while a type with a negative polarization is translated by a negated atom.

Definition 9 (Signed types). *Let $\mathcal{V}$ be a vocabulary. A* signed concept (resp. relation) type *on $\mathcal{V}$ is a pair of form $(+,t)$ or $(-,t)$ where t is a concept type of $\mathcal{V}$ (resp. a relation type of $\mathcal{V}$). $(-,\top)$ is not an allowed signed type. A* signed conjunctive concept type *is a set $\{s_1,\ldots,s_p\}$ of signed concept types.*

Definition 10 (Polarized CGs). *A polarized CG is defined as a simple CG with signed types used in the labels of concept nodes and relations.*

Though that transformation does not preserve reasonings, it is possible to encode a polarized graph into a simple CG. Let us consider the following transformation *sg*:

For each type $t \neq \top$ appearing in a type hierarchy T of the vocabulary V, replace t by the two types $+t$ and $-t$ in the same type hierarchy of the vocabulary $sg(V)$. Then for each $t \leq t' \neq \top$ in T, add $+t \leq +t'$ and $-t' \leq -t$ in the type hierarchy T of $sg(V)$. $\top$ is also the maximal element in the obtained type hierarchy.

For each signed type $(+,t)$ or $(-,t)$ appearing in a polarized graph G, replace that type by the corresponding type $+t$ or $-t$ in the simple CG $sg(G)$

Semantics. Let $G = (C, R, \gamma, \lambda)$ be a polarized CG. We translate G by a FOL formula $\Phi^*(G)$ defined as follows: for every concept type c with type $(+,t_1) \sqcap \ldots \sqcap (+,t_k) \sqcap (-,t'_1) \sqcap \ldots \sqcap (-,t'_q)$, we have the formula $\phi^*(c) = t_1(\mathit{marker}(c)) \wedge \ldots \wedge t_k(\mathit{marker}(c)) \wedge \neg t'_1(\mathit{marker}(c)) \wedge \ldots \wedge \neg t'_q(\mathit{marker}(c))$. For every relation with type $(+,t)$, we have $\phi^*(r) = \phi(r)$, and for every relation with type $(-,t)$ we have $\phi^*(r) = \neg(\phi(r))$. Then the formula $\phi^*(G)$ is built from the formulae $\phi^*(x)$ in the same way as $\Phi(G)$ is built from the formulae $\phi(x)$.

Computing Deduction. Though satisfiability of a polarized CG is easy to check in linear time, the homomorphism mechanism is insufficient to compute deduction in this formalism (a Π^2P-complete problem). One has to rely upon *completions*.

Property 1 (Satisfiability). A polarized CG G, with $nf(G) = (C, R, \gamma, \lambda)$, is unsatisfiable if and only if either there exists a concept node $c \in C$ and concept types t, t' such that $\{(+,t),(-,t')\} \subseteq \mathit{type}(c)$, and $t \leq t'$; or there exists two relations r and r' such that $\gamma(r) = \gamma(r')$, $\lambda(r) = (+,t)$, $\lambda(r') = (-,t')$ and $t \leq t'$.

Property 2. Let F and Q be two polarized CGs over a vocabulary V. Then $\Phi(sg(V))$, $\Phi(sg(F)) \vdash \Phi(sg(Q)) \;\Rightarrow\; \Phi(V), \Phi^*(F) \vdash \Phi^*(Q)$, but the converse is false in general.

We have here encoded part of the negation semantics with simple CGs, but we're still missing the axioms translating the excluded middle principle. Intuitively, let us consider fact G asserting that a blue cube A is on top of a cube B that is itself on top of a cube C that is not blue. Now our question Q is: is there a blue cube x on top of a cube y that is not blue ? Though there can be no homomorphism from Q to G, FOL asserts that the cube B is either blue or not blue. And in both cases we find an answer to the question Q, thus proving deduction. This is the kind of reasonings that led [13] to use the notion of completion in order to obtain a sound and complete deduction mechanism.

Definition 11 (Completion). *A completion of a polarized CG G is a satisfiable polarized CG G' obtained from G by a sequence of applications of the following rules. G' is said maximal if no application of one of these rules produces new information.*
AddC: *If c is a concept node of G and t is a relation type appearing in G, then replace marker(c) with marker$(c) \cup \{(+,t)\}$ or marker$(c) \cup \{(-,t)\}$;*
AddR: *If $c_1, \ldots, c_p$ are concept nodes of G and t is a relation type of arity p appearing in G, then add a relation r with $\gamma(r) = (c_1, \ldots, c_p)$ and $\lambda(r) = (+,t)$ or $(-,t)$.*

Theorem 5. *Let F and Q be two satisfiable polarized CGs defined over $\mathcal{V}$. Then $\Phi(\mathcal{V})$, $\Phi^*(F) \vdash \Phi^*(Q)$ iff, for every completion F' of F, $\Phi(sg(\mathcal{V})), \Phi(sg(F')) \vdash \Phi(sg(Q))$.*

3.2 Computing PG Deduction with Default CG Rules

Let us now show that default CG rules can handle negation of polarized graphs. We consider a set $\mathcal{D}^*$ of default rules that handles the completion mechanism. To each concept type t we can associate two default CGs whose logical translation are:

$$D^+(t) = \frac{+\top(x) : -t(x)}{+t(x)} \text{ and } D^-(t) = \frac{+\top(x) : +t(x)}{-t(x)}$$

Intuitively, the first one asserts that for any concept node c and any concept type t, we can assert that c has type t unless something else makes us deduce that c has type $\neg t$. And to each relation type t of arity k we can also associate two defaults CGs:

$$D^+(t) = \frac{+\top(x),\ldots,+\top(x_k) : -t(x_1,\ldots,x_k)}{+t(x,\ldots,x_k)} \text{ and } D^-(t) = \frac{+\top(x_1),\ldots,+\top(x_k) : +t(x_1,\ldots,x_k)}{-t(x_1,\ldots,x_k)}$$

Theorem 6. *Let F and Q be two satisfiable polarized CGs defined over $\mathcal{V}$. Then $\Phi(\mathcal{V})$, $\Phi^*(F) \vdash \Phi^*(Q) \Leftrightarrow \mathcal{K} = ((\Phi(sg(\mathcal{V})) \wedge \Phi(sg(F))); \Delta(\mathcal{D}^*)) \vdash_S \Phi(sg(Q))$.*

Proof: The proof is immediate, and based on Th. 5, since we can easily check that the fact labeling each node of DDT$(\mathcal{K})$ is a completion of F, and then that the extensions of $\mathcal{K}$ encodes all the possible maximal completions of F. □

A first remark is that encoding the completion mechanism in default CG rules seems like overkill. Indeed, a simple negation as failure mechanism asserting, for example, that for any concept node c, we can add him the type $+t$ unless $-t$ is already present would provide us with the same result, without resorting to such a complex reasoning mechanism. But that simple solution would not cope with polarized rules.

3.3 An Extension to Polarized Graphs Rules

We show here that the set of default rules $\mathcal{D}^*$ used to represent the semantics of polarized graphs is sufficient to handle those of a new CG language that combines rules and atomic negation: polarized CG rules.

Definition 12 (Polarized CG rules). *A* polarized CG rule *is a pair $R = (H, C)$ of polarized graphs where H is called the hypothesis of the rule and C its conclusion.*

Semantics. Let $R = (H, C)$ be a polarized CG rule. Then the FOL interpretation of the rule 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 $\Phi^*(\mathcal{R})$ is the conjunction of the formulae $\Phi^*(R)$, for every rule $R \in \mathcal{R}$.

Computing Deduction

Theorem 7. *Let F and Q be two polarized CGs, and $\mathcal{R}$ be a set of polarized CG rules, all defined over a vocabulary $\mathcal{V}$. Then $\Phi(\mathcal{V}), \Phi^*(F) \vdash \Phi^*(Q)$ iff either 1) there exists an unsatisfiable polarized CG U such that $\Phi(sg(\mathcal{V})), \Phi(sg(F)), \Phi(sg(\mathcal{R})) \vdash \Phi(sg(U))$; or 2) $((\Phi(sg(\mathcal{V})) \wedge \Phi(sg(F)) \wedge \Phi(sg(\mathcal{R})))$; $\Delta(\mathcal{D}^*)) \vdash_S \Phi(sg(Q))$.*

Proof: We provide here the reader with a sketch of proof since a complete one would require a precise introduction of the combined Tableaux/CG reasoning mechanisms of [7]. First see that if condition 1) is satisfied, then the knowledge base $(\mathcal{V}, F, \mathcal{R})$ is unsatisfiable, and everything can be deduced from it. Then see that condition 2) means that Q can be deduced from any satisfiable polarized CG that can be obtained by a sequence of completion and rule application. The possible completions correspond to the different branchings in the tableaux algorithm of [7]. □

A range restricted rule is a rule where all generic markers of the conclusion are already in the hypothesis (e.g. such that their logical interpretation Φ^* admits no existentially quantified variable in their conclusion). Finally, we can present a first decidability result for polarized CG rules.

Theorem 8. *Let $\mathcal{R}$ be a set of* range restricted *polarized CG rules. Then the deduction problem is decidable.*

Proof: Thanks to Theorem 7, we can encode that deduction problem into a skeptical deduction using a default theory whose "normal"rules are those of $\mathcal{R}$, and whose defaults are those of $\mathcal{D}^*$. Since the standard part of these defaults is range restricted, and the union of two range restricted sets of rules is range restricted, we can conclude, thanks to Theorem 9 of [1], that the deduction problem is decidable. □

4 A Two-Players Game Artificial Intelligence Using Default CG Rules

In this section, we present an original application of default reasoning based on a two players board game. To ensure a maximum readability, we have chosen a very simple board game: Tic-Tac-Toe. This model can be easily adapted for other kind of games like Four in a Row. We begin to describe how to represent the initial state of the game that is a simple CG drawing of a 3×3 grid and two players. Then using default CG rules, we give the evolution rules that permits to obtain the tree of all possible states of the game. Finally, we give some default rules that allow to find the best way to play.

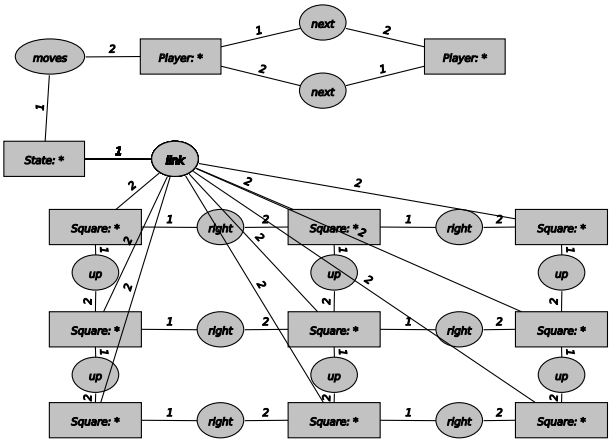

Fig. 1. Initial state of the board game

4.1 Initial Game Board

The initial game board is presented in Figure 1. Concept nodes typed `Player` represent the two who are engaged in the game. We present here the model of a two player game, but it is possible to add as many players as we want in the same way. A relation `next(Player,Player)` indicates who plays next someone. The state of the board during the game is represented by a concept node `State`. A state is linked with the player who has to play this turn by the relation $moves(State, Player)$. The `State` is also linked to several concept nodes that represent the current topology of the game board. For our Tic-Tac-Toe game, the board is constructed with 9 concept nodes `Square`, linked together by some relation `up` and `right` that indicates the relative position of each square. All the squares of a given state are related to the concept node `State` that it describes (see Figure 1).

4.2 Creating the Space of All Possible Games

Now let us see how to simulate a turn in a given state. To this purpose, we need some rules that create a new state, with a new board in which one the square has been modified. One rule is used to create the new state and duplicate the played square, and several ones are needed to rebuilt the whole board by cloning the squares of the precedent state. Figure 2.A presents the main playing rule. This default CG rule can be applied in a state in which there is an empty square. That means that the rule needs two justifications ensuring that any player already played in the square (the justifications of the defaults are represented here in dark nodes, please note that n different colors are needed to represent in a single graph n different justifications). We supposed that if a player has already won in a given state, then the state is tagged with the relation `over`. The fact that such a state can not be played is ensured by adding the needed justification. When applied,

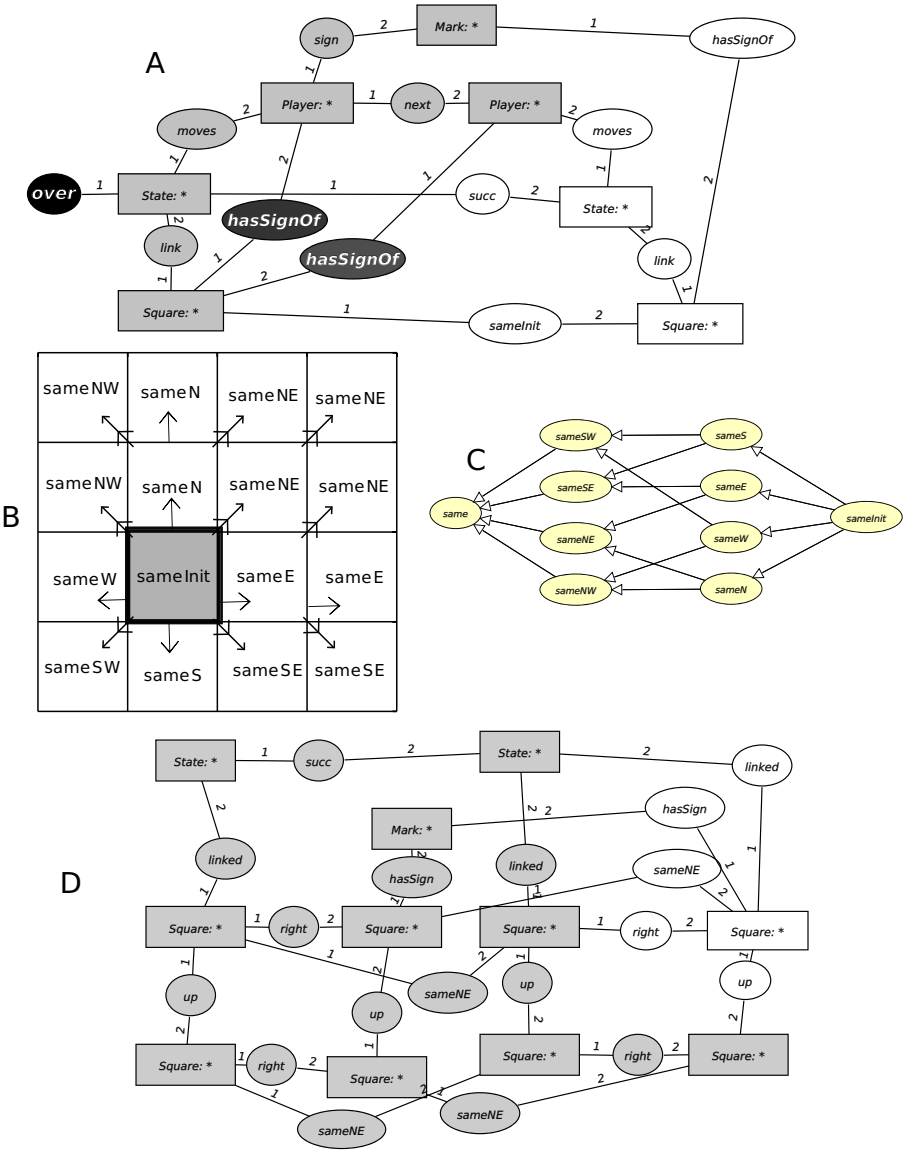

Fig. 2. Playing rules

the rule creates a new state that is indicated as a *successor* of the previous one, linked to the next player and a square linked to the player who just plays. This square is also linked to its previous instantiation by the relation `sameInit`.

Once this default rule is applied, the whole grid has to be replicated from the played square. The difficulty here is that we want to ensure that each square and relative position of the initial state is cloned once and only once, and linked to the correct player

mark. The method that we present here can be used to duplicate any kind of square-checkered board. To do that we start from the first duplicated square that is the last played one in the Tic-Tac-Toe game. This square is linked with its "father" by the special relation `sameInit`. From this square, we can duplicate the squares that are in the four cardinal directions with four different rules that use four specialized `same` relations named `sameN`, `sameS`, `sameE` and `sameW` which duplicate the squares in the north, south, east and west direction. The propagation of the cloning process is shown in Figure 2.B for a 4 by 4 grid.

Then four rules are needed to duplicate squares in each quarter of the plan. The rule that duplicates the right upper quarter plan (in the direction North-East) is given in Figure 2.D. To ensure that the duplication process runs correctly in each direction, it is sufficient to use the special hierarchy relation shown in Figure 2.C. It means that the relation `sameN` can be used for the generation of both north-east quarter plan and north west quarter plan. So all the duplication rules are directed and non–compatible (due to the relation hierarchy), and each square is duplicated once and only once.

Some rules are needed to end the game. For example if we can match 3 aligned squares (linked together by the relation `right`) played by the same player then we conclude that this player wins, the other player looses and the corresponding state is over. This rule is not a default one. Two other default rules are also needed, one to end the party tagging `over` a state over if there is no more space on the grid, and the other one to deduce that an over state is draw unless there is a winner. Applying this set of rules permit to create the complete search tree of all possible games (see Figure 3) which is the only extension of our model, and knowing who is the winner of each terminal board.

4.3 Searching the Best Way to Play

Knowing who wins each terminal state in the graph of all possible state makes easy to recursively deduce the status of each state of the game. That will allow to determine for each state if a player is ensured to win, lose or can obtain a draw.

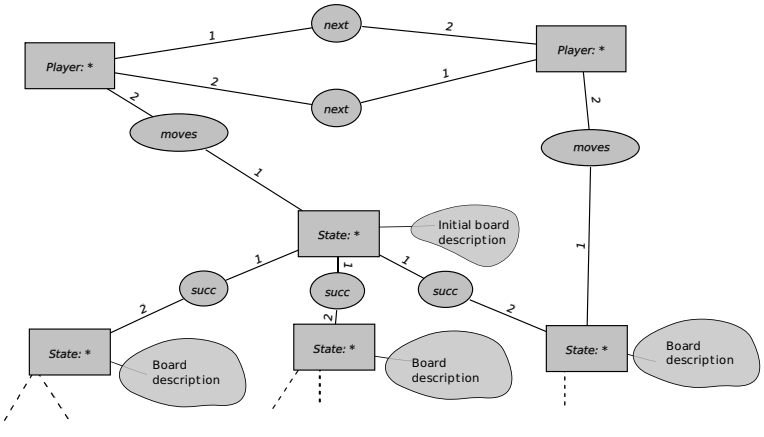

Fig. 3. General model of playing game

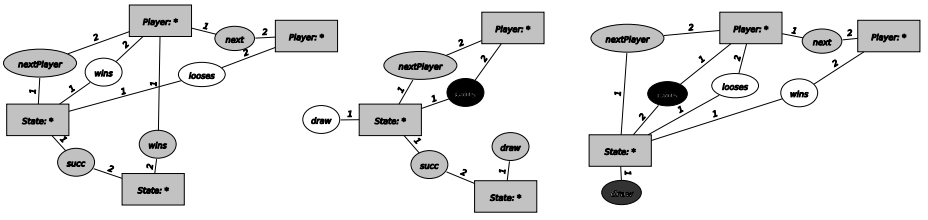

Fig. 4. Status deduction rules

To do that we need the 3 more rules shown in Figure 4 which shows the rules that permit to deduce the status of a state from the status of its successors. For a given state, the status of the game can be draw or won by a player. The status information is initialized in the terminal states, which are leaves of the tree of the possible states according to ending rules. The status of the game is then recursively computed from the leaves to the root of the possible states tree. For a state S, if there is an immediate state successor in which the current player of S is the winner, then S is a winning state for its current player because there is a way for the player to win whatever the other player does (first rule of Figure 4). Otherwise, if there is an immediate successor for which the game status is draw, then the status of the state S is draw, as it is possible for our player to ensure at least a draw (second rule of Figure 4). To apply this rule, we need to add that the current player is not already ensured to win, which is the justification of this rule. If it is not possible for the current player of S to win, nor to obtain a draw, then S is a loosing state for the current player. Note that in this presented rule, the two dark nodes represent two different justifications. That means that if one of this dark nodes can be projected, then the default can not be applied.

Finally, applying all this rules leads to one and only one extension in which we can easily find what is the best way for a player in any state of the game. As we have only one extension, the sceptical and credulous deductions are equivalent. This extension contains 26 830 different possible states of the game. This huge search tree does not take into account the fact that some states are equivalent up to reflexions and rotations (that can be computed through rules).

To know if there is a best way to play in one state, one can just try to find in the extension if the current player is linked to the initial state by the relation `wins`. If it is the case, then one of the best way to play can be found in searching one successor of the initial state in which our player is winning.

5 Conclusion

In this paper we have studied two applications of the default CG rules introduced in [1], in order to assess the expressivity of that new language for the CG family. We have first shown that default CGs allowed to express the semantics of atomic negation in FOL, with a concise and intuitive model (the set of defaults $\mathcal{D}^*$) that translated exactly the knowledge present in the polarized graph algorithm of [13]. In the same time, we have used default CGs to prove a new decidability result in an expressive CG language

that mixes CG rules with atomic negation. Then, we have exhibited a concise default CGs model of the Tic-Tac-Toe game that shows that, though default CGs are more complex than the usual CG rules, they offer possibilities to stop the generation of new consequences and thus, as can be done for the cut in Prolog, to encode reasonings in an efficient way. Our goal is now twofold: firstly, to encode a preference mechanism on the defaults, allowing for example in our game to consider first the extensions in which we can prove we will win, and only in case they are none the extensions in which we do not lose; and secondly, to study efficient reasoning mechanisms in default CG rules, building upon the results obtained for CG rules by [3].

References

1. Baget, J., 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.) Conceptual Structures: Leveraging Semantic Technologies. 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. 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)
4. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. In: Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), pp. 77–86 (2009)
5. Sowa, J.F.: Conceptual graphs for a database interface. IBM Journal of Research and Development 20, 336–357 (1976)
6. Salvat, E., Mugnier, M.L.: Sound and complete forward and backward chaining of graph rules. In: Eklund, P., Mann, G.A., Ellis, G. (eds.) ICCS 1996. LNCS, vol. 1115, pp. 248–262. Springer, Heidelberg (1996)
7. Kerdiles, G.: Saying It with Pictures: a logical landscape of conceptual graphs. PhD thesis, Univ. Amsterdam (2001)
8. Tepfenhart, W., Cyre, W.: Proceedings of the 7th International Conference on Conceptual Structures, ICCS 1999, Blacksburg, VA, USA, July 12-15. Springer, Heidelberg (1999)
9. 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)
10. Gottlob, G., Leone, N., Scarcello, F.: A comparison of structural CSP decomposition methods. Artificial Intelligence 124 (2000)
11. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison Wesley, Reading (1984)
12. Brewka, G., Eiter, T.: Prioritizing default logic: Abridged report. In: Festschrift on the occasion of Prof.Dr. W. Bibel's 60th birthday. Kluwer, Dordrecht (1999)
13. Mugnier, M., Leclere, M.: On querying simple conceptual graphs with negation. Data & Knowledge Engineering 60(3), 468–493 (2007)