Extending the CG Model by Simulations

Jean-François Baget

LIRMM (CNRS and Université Montpellier II)
161, rue Ada, 34392 Montpellier - France
baget@lirmm.fr

Abstract. Conceptual graphs (CGs) share with FOL a fundamental expressiveness limitation: only higher-order logics allow assertions of properties on predicates. This paper intends to push back this limit by reifying underlying relations of CGs (*is-a*, *a-kind-of*, *referent*) into first-class objects (i.e. nodes) of an equivalent, labelled graphs (LG) model. Benefits of this reification, applied on a subset of CGs, namely simple graphs and rules of form "if G then H", are discussed in terms of expressiveness, succintness and robustness. We show that using the LG model as an *interpreter* allows us to improve and extend the results in [2].

1 Introduction

Labelled Graphs have long been used as a natural and readable way to represent symbolic knowledge. The *Conceptual Graphs* model [11] can be seen as a higher level abstraction of Representation Networks [9,6], benefiting from further developments in Knowledge Representation: a clear distinction between *entities* and *relations*, and between *factual knowledge* and *background knowledge* (Fig. 1).

Though the CG model adds structuration and clarity to the represented knowledge, this improvement has a subtle trade-off: the relations s (for *subset of*, *a kind of*) and e (for *element of*, *is a*), explicit in representation networks, become implicit in CGs. Indeed, *a kind of* is encoded in the type hierarchies defining background knowledge, and *is a* is encoded in concept node labels.

The drawback is that CGs cannot handle these relations as "first-class objects", they are used in the deduction mechanism, but cannot be the *object* of reasonings. From a FOL point of view, it is possible to assert properties on objects, but not on relations (predicates) between objects. Some consequences are pointed out in [2], in a CG model restricted to simple graphs (SGs) and "if ... then" rules (SG rules). Namely:

1. It is not possible to express that *"If two concept nodes have the same individual marker, they represent the same entity"*. Instead, [2] uses one rule for each individual marker in the support (i.e. for each individual marker m, "if two concept nodes are marked by m, then they represent the same entity").
2. It is not possible to define the relation type EQUIVALENCE as a subtype of REFLEXIVE, SYMMETRICAL and TRANSITIVE, such that these properties (themselves encoded in rules) are inherited by EQUIVALENCE and all its subtypes. Instead, [2] defines a family of rules expressing the equivalence property, that must be implemented for *every* equivalence relation.

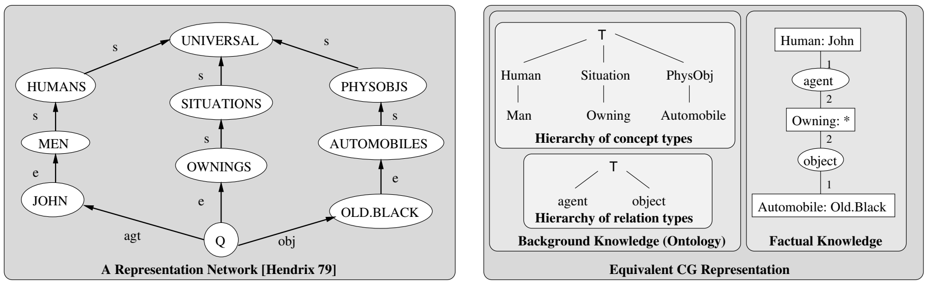

Fig. 1. Representation Networks and Conceptual Graphs Formalisms

These solutions are not satisfying in terms of succintness (families of rules defined for each marker and/or type) neither in terms of robustness (when a new type or marker is added, it is necessary to add new rules). Instead, we propose to reify implicit relations of the SG model (IS-A, A-KIND-OF, REFERENT) into first-class objects of a model that *simulates* SGs.

In Sect. 2 we present a general framework for simulating a reasoning model into another one. This framework will then be applied for simulating the {SG, rules} model (recalled in Sect.3) into a {labelled graphs, rules} model. Sect. 4 and 5 study this latter model, its semantics, and its expressiveness with respect to {SG, rules}. In Sect. 6, we build step by step a simulation of the {SG, rules} model into the {labelled graphs, rules} model, where implicit notions of the simulated model are reified. Finally, in Sect. 7, we discuss our gains in terms of expressiveness and present an extension of the {SG, rules} model, obtained from this simulation.

2 A Simulation Framework for Reasoning Models

In this paper, we study how a reasoning model can gain expressiveness by a simulation into another one. As [7], we define a reasoning model by: a *language* or *syntax* that specifies the objects we manipulate; a *deduction system* that computes the relation $\models$ between objects; and *valuation rules* or *semantics* that are given here through a translation into FOL formulas. To compare different knowledge representation formalisms, [5] gives several evaluation criteria:

1. Does the formalism support efficient reasonings?
2. How expressive is it?
3. How succinctly can the formalism express the sets of models that it can?
4. How does the knowledge representation form fare in the face of change?

Comparison of reasoning models in terms of complexity or decidability of the deduction problem (1.) is only slightly discussed in this paper. To compare the expressiveness of two reasoning models (2.), we define the notion of *simulation* of a reasoning model into another one, which is basically a reduction of a decision problem to another. We propose here some criteria to evaluate the very subjective notion of a *good* simulation, and discuss in the final part of this paper how a simulation that reifies implicit notions of a model can add expressiveness, succintness (3.) and robustness (4.) to this model.

Definition (Simulation). *A* simulation *of a reasoning model $\mathcal{M}$ into a model $\mathcal{N}$ is a mapping Γ from the objects of $\mathcal{M}$ into the objects of $\mathcal{N}$ such that A can be deduced from B in $\mathcal{M}$ if and only if $\Gamma(A)$ can be deduced from $\Gamma(B)$ in $\mathcal{N}$.*

2.1 Complexity of Γ: Importance of a Linear Simulation

We want a simulation to prove that *all reasonings in the model $\mathcal{M}$ can be performed in $\mathcal{N}$*, and wish to forbid part of the deduction mechanism in $\mathcal{M}$ to be encoded in the mapping Γ. The usual way to assert that $\mathcal{M}$ is "simpler" than $\mathcal{N}$ is to find a *polynomial time* simulation of $\mathcal{M}$ into $\mathcal{N}$. But a polynomial time mapping could be complex enough to encode in itself mechanisms of $\mathcal{M}$ that cannot be represented in $\mathcal{N}$ (Sect. 6.1). We want a simulation to compute a *syntactic translation* only. That is why we are interested in simulations that fulfill a much stronger constraint: *linear time mappings*. If there exists a polynomial time (resp. linear time) simulation of $\mathcal{M}$ into $\mathcal{N}$, we say that $\mathcal{N}$ is a *generalization* (resp. *strong generalization*) of $\mathcal{M}$. Reasoning models that generalize each other are said *equivalent* (resp. *strongly equivalent*).

2.2 Backward Translation of Objects and Proofs

It is important, for every object A in $\mathcal{N}$, to compute efficiently if $A \in \Gamma(\mathcal{M})$ (i.e. if A is the representation of an object of $\mathcal{M}$) and, in that case, to translate back A into the *unique* object A' of $\mathcal{M}$ such that $\Gamma(A') = A$; thus Γ should be injective. This feature allows to develop a reasoning model on top of an existing one by implementing only the simulation Γ and its backward translation in such a way that this translation is invisible to the end user: he only needs to manipulate objects of the model $\mathcal{M}$. The interest of backward translation of objects extends to backward translation of *proofs*. Intuitively, if A and B are objects of $\mathcal{M}$, a proof of $A \vDash B$ in the model $\mathcal{M}$ can be seen as a sequence of operations allowed by the deduction system of $\mathcal{M}$. Consider a proof $\Gamma(A) \vDash \Gamma(B)$ of the model $\mathcal{N}$. A proof $A \vDash B$ of $\mathcal{M}$ can be built if there exists a *surjective mapping* of the proofs in $\Gamma(\mathcal{M})$ into the proofs in $\mathcal{M}$. This property would allow an end user to visualize all proofs in the model $\mathcal{M}$, even if reasonings are *interpreted* into the model $\mathcal{N}$. Even more, should we want an application that compute deduction in $\mathcal{M}$ with some user interaction features (step by step visualization of proof, user assisted deduction), this property should extend to subsequences of a proof.

2.3 Preservation of Complexity Classes

Let $C_{\mathcal{M}}$ and $C_{\mathcal{N}}$ be the complexity classes of the deduction problems in $\mathcal{M}$, and $\mathcal{N}$; let $\mathcal{A}_{\mathcal{M}}$ and $\mathcal{A}_{\mathcal{N}}$ be algorithms solving these problems ("respecting" the complexity classes $C_{\mathcal{M}}$ and $C_{\mathcal{N}}$); and let Γ be a simulation of $\mathcal{M}$ into $\mathcal{N}$. We want to keep reasonings in the model $\mathcal{M}$ *as efficient as possible*, even when they are interpreted through the translation in the model $\mathcal{N}$. That is why $\mathcal{A}_{\mathcal{N}}$, applied to objects of $\Gamma(\mathcal{M})$, should satisfy $C_{\mathcal{M}}$. Note that, if it is not the case, $\mathcal{A}_{\mathcal{M}}$ can still be used, assuming that we have an efficient reconnaissance algorithm for $\Gamma(\mathcal{M})$, or can even give new insights for a general improvement of $\mathcal{A}_{\mathcal{N}}$. This discussion could also be extended to the preservation of complexity classes for particular subsets of $\mathcal{M}$.

In this paper, we propose a linear simulation Θ from the {SG, rules} model into a "low-level" model. Since there also exists a linear simulation from the latter model into {SG, rules}, both models are strongly equivalent. This simulation ensures backward translations of objects as well as proofs, and, since the two models are strongly equivalent, complexity classes of deduction problems are preserved. The simulation Θ reifies implicit notions of the CG model (types, markers) into first-class objects of the "low-level" model. Benefits of this reification will be further discussed in the last section (Sect. 7) of this paper.

3 The Simple Graphs and {SG, Rules} Models

Here we recall fundamental definitions and results on a subset of CGs: simple graphs (SGs) [4], and simple graphs rules [8,2].

3.1 Simple Graphs

Syntax. Basic ontological knowledge is encoded in a structure called a *support.* A support $\mathcal{S} = (T_C, T_R, \mathcal{I})$ is given by two finite partially ordered sets T_C and T_R, respectively the set of *concept types* and the set of binary[1] *relation types*, and a set of *individual markers* $\mathcal{I}$. T_C (resp. T_R) admits a greatest element, denoted by $\top_C$ (resp. $\top_R$). A partial order is defined on $\mathcal{I} \cup \{*\}$: the *generic marker* $*$ is the greatest element and elements of $\mathcal{I}$ are pairwise non comparable.

We denote by $G = (V_C, V_R, E, \text{label}, \text{co-ref})$ a *simple graph* defined on a support, where V_C and V_R are respectively the sets of *concept nodes* and *relation nodes*; E is the set of *edges* (the two edges incident to a relation node are labelled by 1 and 2); *co-ref*, the co-reference relation, is an equivalence relation on the set of generic concept nodes. Two concept nodes are said *co-identical* if they have the same individual marker or if they belong to the same *co-reference* class. Intuitively, it means that these two concept nodes represent the same entity. In what follows, we impose that *co-identical* concept nodes have the same type.

Deduction System. Let $\mathcal{S}$ be a support, and G and H be two SGs defined on $\mathcal{S}$. A projection Π from H into G is a mapping from $V_C(H)$ into $V_C(G)$ and from $V_R(H)$ into $V_R(G)$ that preserves edges and their labelling, may restrict labels of concept and relation nodes, and preserves co-identity:

1. $\forall e = (x_1, x_2) \in E(H), e' = (\Pi(x_1), \Pi(x_2)) \in E(G)$ and $\text{label}(e) = \text{label}(e')$
2. $\forall x \in V_C(H) \cup V_R(H), \Pi(x) = x' \Rightarrow \text{label}(x') \leq \text{label}(x)$
3. $\forall x, y \in V_C(H), \text{co-ref}(x, y) \Rightarrow \text{co-ident}(\Pi(x), \Pi(y))$

[1] For the sake of simplicity, we restrict these definitions to binary relation types but all results could easily be extended to n-ary relation types.

A SG is said in *normal form* if all co-identity classes are restricted to the trivial ones (co-ident$(x, y) \Rightarrow x = y$). The normal form $NF(G)$ of a graph G is obtained by fusionning all nodes that belong to the same co-identity class.

We consider two deduction systems in the SG model (effectively defining two reasoning models): we note $\mathcal{S}, G \vDash H$ if there exists a projection from H into G, and $\mathcal{S}, G \vDash_{NF} H$ if there exists a projection of H into $NF(G)$ (remark that if $\mathcal{S}, G \vDash H$, then $\mathcal{S}, G \vDash_{NF} H$, but the reverse does not hold).

Semantics. The deduction system $\vDash_{NF}$ is sound and complete w. r. t. the FOL semantics Φ [4], i. e. $\mathcal{S}, G \vDash_{NF} H$ iff $\Phi(\mathcal{S}), \Phi(G) \vDash \Phi(H)$, while the deduction system $\vDash$ is sound and complete w. r. t. the FOL semantics Ψ (an alternative to Φ proposed in [10]). A formal presentation of these semantics can be found in [10], while [2] presents their consequences on the different reasonings allowed.

3.2 Simple Graph Rules

Simple graph rules (SG rules) can be seen as graphs that embody knowledge of the form "if *hypothesis* can be deduced from a graph, so can *conclusion*".

Syntax. Let $\mathcal{S}$ be a support. A *simple graph rule* is given by a simple graph R defined on $\mathcal{S}$ and a mapping color: $V(R) \rightarrow \{0, 1\}$. The subgraph of R generated by 0-colored (resp. 1-colored) nodes is called the *hypothesis* (resp. *conclusion*) of the rule. Furthermore, the following constraints ensure that the application of a rule on a SG always generates a SG: 1) the subgraph of R generated by 0-colored nodes is a SG (but without restrictions on *co-ref*); 2) co-identity classes whose members are of different colors are forbidden; 3) co-identity classes whose members are 0-colored nodes are without restriction; 4) co-identity classes whose members are 1-colored nodes suffer the same restrictions as for SGs.

Deduction System. Let $\mathcal{S}$ be a support, G be a SG defined on $\mathcal{S}$, and R be a SG rule defined on $\mathcal{S}$. We say that R is *applicable* to G iff there exists a projection from the hypothesis of R into G. Let Π be such a projection. An *immediate derivation* of G by the application of R following Π is a graph G' obtained by 1) making the disjoint union of G and a copy of the conclusion of R; 2) for each edge (r, c) labelled n in R, where c is a concept node of color 0 and r is a relation node of color 1, linking the copy of r to $\Pi(c)$ by an edge labelled n.

Let $\mathcal{R}$ be a set of SG rules defined on $\mathcal{S}$. We say that $G, \mathcal{R}$ *derives* a SG G' if there exists a sequence of immediate derivations leading to G' by application of rules in $\mathcal{R}$. We say that $G, \mathcal{R}$ *normally derives* G' if each immediate derivation is followed by a normalization of the obtained graph.

Then again, we obtain two deduction systems in the {SG, rules} model. We note $\mathcal{S}, \mathcal{R}, G \vDash H$ if there exists a graph G' such that $G, \mathcal{R}$ derives G' and H can be projected into G'. We note $\mathcal{S}, \mathcal{R}, G \vDash_{NF} H$ if there exists a graph G' such that $NF(G), \mathcal{R}$ normally derives G' and H can be projected into G'.

Semantics. The FOL semantics Φ and Ψ are extended to SG rules: $\vDash_{NF}$ is sound and complete w. r. t. Φ [8], and $\vDash$ is sound and complete w. r. t. Ψ [2].

4 The Labelled Graphs Model

The labelled graphs (LG) model is very similar to the SG model, but does not take into account the notions of support or type: we consider it as a "syntactically neutral" version of SGs. We first define formally the objects we manipulate, then propose two deduction mechanisms based on projection whose semantics mimic the ones defined in the SG model. It is not surprising, this model being simpler, that there exists a linear simulation of labelled graphs into SGs.

4.1 Syntax and Deduction Mechanisms

Let $\mathcal{A}$ be a set of symbols, and $* \notin \mathcal{A}$ a special symbol named the *generic label.* A labelled graph (LG) $G = (V, E, \text{label})$ is given by a set of nodes V, a symmetrical relation E on $V \times V$, and a mapping label from V into $\mathcal{A} \cup \{*\}$.

Let H and G be two LGs, a *LG-projection* (projection in the LG model) is a mapping Π from $V(H)$ into $V(G)$ that preserves edges and non-generic labels:

1. $\forall (x_1, x_2) \in E(H), (\Pi(x_1), \Pi(x_2)) \in E(G)$
2. $\forall x \in V(H), \text{label}(x) \neq * \Rightarrow \text{label}(\Pi(x)) = \text{label}(x)$

A LG is said in *normal form* if all non-generic labels are different. The normal form $NF(G)$ of a LG G is obtained by fusionning nodes having the same, non-generic label. We note $H \sqsubseteq G$ if there exists a LG-projection from H into G.

We note $G \vDash H$ if there exists a LG-projection from H into G, and note $G \vDash_{NF} H$ if there exists a LG-projection from H into $NF(G)$. Unless otherwise noted, we will consider $\vDash$ when referring to the LG model.

4.2 Linear Simulations of Labelled Graphs into SGs

Lemma 1. *The* $(SGs, \vDash)$ *model is a strong generalization of the* $(LGs, \vDash)$ *model and the* $(SGs, \vDash_{NF})$ *model is a strong generalization of the* $(LGs, \vDash_{NF})$ *model.*

Proof. This result is not surprising, but the simulations Γ of $(\text{LGs}, \vDash)$ into $(\text{SGs}, \vDash)$ and Γ_{NF} of $(\text{LGs}, \vDash_{NF})$ into $(\text{SGs}, \vDash_{NF})$ will be used in Theor.1.

The simulation Γ is illustrated in Fig. 2 by the transformation of the LG G_1 into the SG G_2. We define a support $\Gamma(\mathcal{A})$ as follows: $*$ is associated to the greatest element of T_C, each label of $\mathcal{A}$ is associated to a distinct type of T_C (denoted by the same symbol). All types corresponding to $\mathcal{A}$ are pairwise non-comparable. T_R is restricted to one relation type, named *link.* $\mathcal{I}$ is empty. Each node of an LG is transformed into a concept node whose type is the one associated with its label and whose marker is generic. Each edge ab is transformed into two symmetrical relation nodes typed *link*, linking $\Gamma(a)$ and $\Gamma(b)$.

Γ_{NF} is illustrated in Fig. 2 by the transformation G_1 into G_3. The only concept type in $\Gamma_{NF}(\mathcal{A})$ is *node*, T_R is restricted to the relation type *link*, and $\mathcal{I}$ is equal to $\mathcal{A}$. The difference with Γ consists in concept node labeling: a node labelled m is transformed into a concept node whose type is *node*, and whose marker is generic if m is generic, or the element of $\mathcal{I}$ associated to m otherwise.

We now check that Γ and Γ_{NF} are linear transformations, that $G \vDash H$ iff $\Gamma(\mathcal{A}), \Gamma(G) \vDash \Gamma(H)$, and that $G \vDash_{NF} H$ iff $\Gamma_{NF}(\mathcal{A}), \Gamma_{NF}(G) \vDash_{NF} \Gamma_{NF}(H)$. □

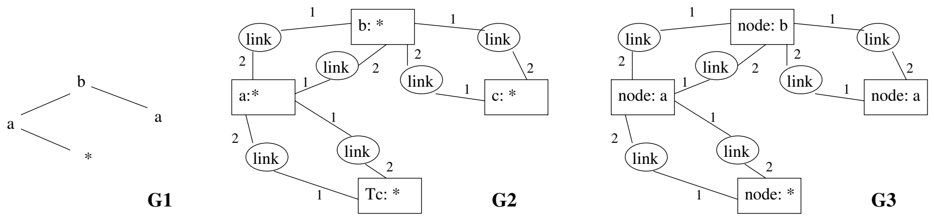

Fig. 2. Simulations of LG models in SGs

4.3 Semantics

Semantics ϕ and ψ for the two labelled graph models are defined as:

- $\psi(\mathcal{A}) = \Phi(\Gamma(\mathcal{A}))$ and $\psi(G) = \Phi(\Gamma(G))$
- $\phi(\mathcal{A}) = \Phi(\Gamma_{NF}(\mathcal{A}))$ and $\phi(G) = \Phi(\Gamma_{NF}(G))$

Theorem 1 (Soundness and completeness). *The deduction system* $\vDash$ *(resp.* $\vDash_{NF}$*) in the LG model is sound and complete w. r. t. the semantics* ψ *(resp.* ϕ*).*

Proof. This is a corollary of Lem. 1, since Γ only creates SGs in normal form. □

5 The {Labelled Graphs, Rules} Model

Labelled graph rules (LG rules) are designed as a "light syntax" version of SG rules. Extending the results in Sect. 4, we simulate the {LG, rules} model into the {SG, rules} model, and give sound and complete semantics.

5.1 Syntax and Deduction Mechanisms

A *LG rule* $R = (V, E, \text{label}, \text{color})$, is given by a LG (V, E, label) and a mapping color from V into $\{0, 1\}$. The subgraph $\text{hyp}(R)$ (resp. $\text{conc}(R)$) generated by 0-colored (resp. 1-colored) nodes, circled in white (resp. gray) in the representation of the graph, is called the *hypothesis* (resp. *conclusion*) of the rule (Fig. 3).

A LG rule R is *applicable* to a LG G if there exists a LG-projection (say Π) from the hypothesis of R into G. In that case, a LG G' is an *immediate derivation* of G following R and Π if G' is obtained by making the disjoint union of G and a copy of $\text{conc}(R)$, then, for each edge $hc \in E(R)$ such that $h \in \text{hyp}(R)$ and $c \in \text{conc}(R)$, adding an edge between $\Pi(h)$ and $\text{copy}(c)$.

The mechanism of rule application is illustrated in Fig. 3, where $G_1, \ldots G_5$ are the immediate derivations of G following R. The notions of *derivation* and *normal derivation* are defined as in Sect. 3.2, as a sequence of immediate derivations (each one followed by a normalization in the latter case). We note $\mathcal{R}, G \vDash H$ if there exists a graph G' such that $G, \mathcal{R}$ derives G' and there exists a LG projection of H into G'. We note $\mathcal{R}, G \vDash_{NF} H$ if there exists a graph G' such that $NF(G), \mathcal{R}$ normally derives G' and there exists a LG projection of H into G'. Unless otherwise noted, we will use $\vDash$ when considering the {LG, rules} model.

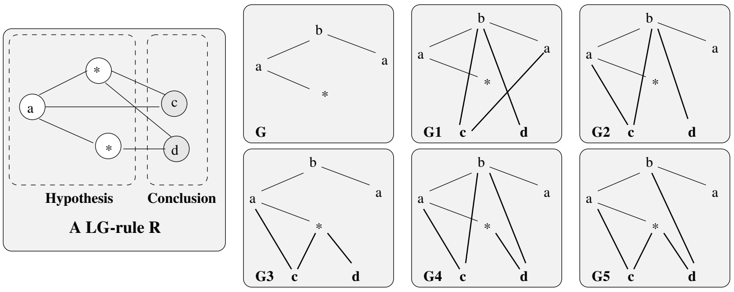

Fig. 3. Five immediate derivations of a graph G

5.2 Linear Simulations of Labelled Graphs Rules into SG Rules

Lemma 2. *The* $(\{SGs,\ rules\}, \models)$ *(resp.* $(\{SGs,\ rules\}, \models_{NF})$*) model is a strong generalization of the* $(\{LGs,\ rules\}, \models)$ *(resp.* $(\{LGs,\ rules\}, \models_{NF})$*) model.*

Proof. As in the proof of lemma 1, we exhibit two simulations Δ and Δ_{NF}. Let $R = (G, \text{color})$ be a LG rule. $\Delta(R)$ (resp. $\Delta_{NF}(R)$) is obtained by adding to $\Gamma(G)$ (resp. $\Gamma_{NF}(G)$) the following coloration: the color 0 is assigned to nodes issued from the hypothesis of $\mathcal{R}$, the color 1 to all others. We can verify that these simulations are linear, that, thanks to lemma 1, a LG rule R is applicable to a labelled graph G iff $\Delta(R)$ (resp. $\Delta_{NF}(R)$) is applicable to $\Gamma(G)$ (resp. $\Gamma_{NF}(G)$), and that, to every immediate derivation of a LG G into G' following R corresponds an immediate derivation of $\Gamma(G)$ into $\Gamma(G')$ following $\Delta(R)$. □

5.3 Semantics

We extend the semantics ϕ and ψ as in Sect. 4. Let R be a LG rule:

- $\psi(R) = \Phi(\Delta(R))$
- $\phi(R) = \Phi(\Delta_{NF}(R))$

Theorem 2 (Soundness and completeness). *The deduction system* $\models$ *(resp.* $\models_{NF}$*) in the* $\{LG,\ rules\}$ *model is sound and complete w. r. t.* ψ *(resp.* ϕ*).*

Proof. Lemma 2, soundness and completeness for the {SG, rules} models. □

6 Simulations of {SG, Rules} into {LG, Rules}

Having defined the LG and {LG, rules} models as particular cases of SG and {SG, rules} (Lem. 1 and 2), we now simulate various SG models (beginning with a basic one) into {LG, rules} models by reifying implicit relations of SGs.

6.1 Basic Simple Graphs

We call basic simple graph (BSG) a SG defined on a support $\mathcal{S}$ without co-reference links (all co-reference classes are trivial), nor individual markers ($\mathcal{I} = \emptyset$). BSGs being in normal form, the Φ and Ψ semantics are equivalent.

Proposition 1. *The BSG model is equivalent to the LG model.*

Proof. We have proven (lemma 1) that the BSG model is a strong generalization of the LG model, since Γ produces only BSGs. We now exhibit a *polynomial* simulation of the BSG model into the LG model. Fig.4 represents the simulation, by the mapping Θ_1, of the graph in Fig. 1 (without its individual markers).

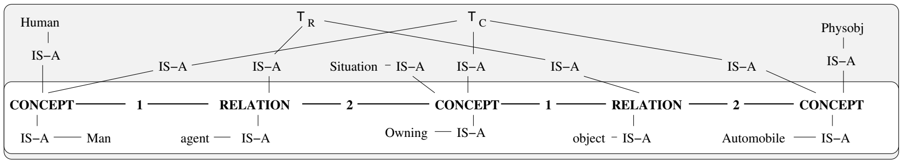

Fig. 4. Simulation of the BSG model into the LG model via Θ_1

We first define skel(G), a LG called the skeleton of G (boldface in Fig. 4), obtained by creating a node skel(c) labelled by CONCEPT (resp. RELATION) for each concept node (resp. relation node) x in G, and, for each edge $rc \in G$, a node labelled by the label of this edge, whose neighbors are skel(r) and skel(c).

The LG $\Theta_1(G, \mathcal{S})$ is then obtained from skel(G), by adding a node labelled by t for each type $t \in T_C \cup T_R$ (there must be no identical symbol in these two sets), then for each node x typed t in G, for each type t' in $T_C \cup T_R$ such that $t \leq t'$, adding a node labelled IS-A linking skel(x) to the node labelled by t'. We denote by $\Theta_1^-(G)$ a subgraph of Θ_1 where nodes are only linked by IS-A to their most specific type (the subgraph in the white rectangle of Fig. 4).

We check that Θ_1 is a *polynomial* simulation. For computational efficiency, we point out that, G and H being two BSGs defined on $\mathcal{S}$: $\mathcal{S}, G \models H$ iff $\Theta_1(G, \mathcal{S}) \models \Theta_1^-(H)$ (leading to a smaller graph to project). □

We doubt it is possible to find a *linear* simulation of the BSG model into the LG model. Intuitively, reasonings by LG-projection and reasonings on the hierarchy of types have a fundamentally different nature: one is concerned with existence of objects, the other embodies knowledge on all types that verify some property. Reasonings on a hierarchy are basically rules. The following simulations will be given by a linear mapping translating only syntactic information of a SG, and by a constant set of rules (considered as a *library* for the LG interpreter) that encode reasonings that cannot be achieved by mere LG-projection.

Proposition 2. *The {LG, rules} model is a strong generalization of the BSG model.*

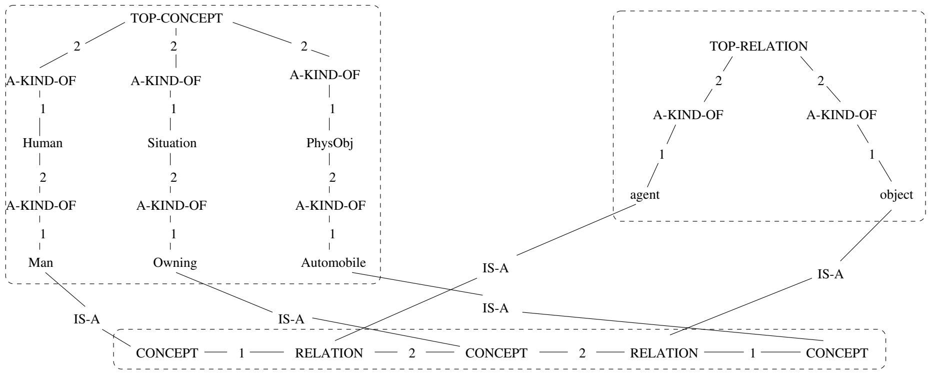

Fig. 5. Transformation via Θ_2 of the BSG and support in Fig. 1 into a LG

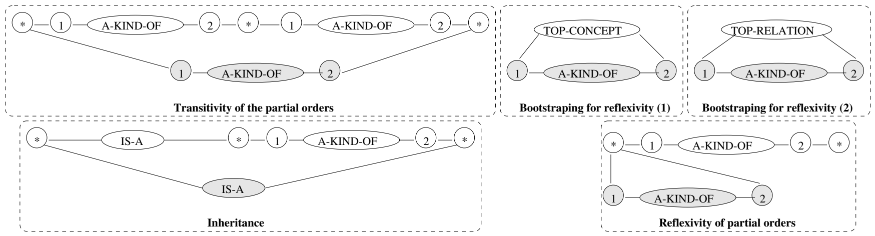
A-KIND-OF
TOP-CONCEPT
TOP-RELATION
IS-A
Transitivity of the partial orders
Bootstraping for reflexivity (1)
Bootstraping for reflexivity (2)
Inheritance
Reflexivity of partial orders

Fig. 6. Library $\mathcal{R}_s$ of LG-rules used for the transformation Θ_2

Proof. Let G be a BSG defined on $\mathcal{S}$. We first encode the support in a LG enc($\mathcal{S}$). This graph contains two particular nodes, labelled TOP-CONCEPT and TOP-RELATION, that represent the types $\top_C$ and $\top_R$. Then, for each type t being a *direct* subtype of types $t_1, \ldots, t_k$, we add a new node labelled t that is linked by a chain – 1 – A-KIND-OF – 2 – to the representation of each of the t_i. The LG $\Theta_2(G, \mathcal{S})$ is obtained by the disjoint union of skel(G) (see proof of Prop. 1) and enc($\mathcal{S}$), then, for each node labelled CONCEPT or RELATION in this skeleton, linking it by a chain[2] – IS-A – to the node labelled by *its most specific type.*

The graph in Fig. 5 illustrates this transformation: note that, this time, transformation of the support has been purely syntactic. No information on the properties of IS-A or A-KIND-OF has been needed to generate this graph. These properties are given by the rules $\mathcal{R}_s$ in Fig. 6. These rules express the transitivity and reflexivity of the partial order on types, and that an entity or relation inherits all the super-types of its given type. Though only this last rule is necessary to prove the proposition, they will all be used when simulating co-reference.

We must now check that Θ_2 is linear and verify the following equivalences (the last one being presented in an optimization perspective): $\mathcal{S}, G \models H$ iff $\Theta_2(G, \mathcal{S}), \mathcal{R}_s \models \Theta_2(H, \mathcal{S})$ iff $\Theta_2(G, \mathcal{S}), \mathcal{R}_s \models \Theta_1^-(H)$ □

[2] As for *a-kind-of*, we could have used a chain – 1 – IS-A – 2 –. This is not necessary here, since the "orientation" is implicitly given, from the object to its type.

Remark 1. All rules presented here require there is a *unique* node representing each type (same for individual markers). All LG rules to be applied on SGs must be designed in such a way that the graphs they generate keep this invariant true.

6.2 Introducing Co-reference and Individual Markers

Proposition 3. *The {LG, rules} model is a strong generalization of the (SG, $\models$) and the (SG, $\models_{NF}$) models.*

Proof. Let G be a SG on a support $\mathcal{S}$. As in [2], reification of co-identity is done by simulation into the SG model itself. $\theta(\mathcal{S})$ is the support obtained by adding to $\mathcal{S}$ four new relation types, namely REFLEXIVE, SYMMETRICAL, TRANSITIVE and EQUIVALENCE, the first being subtypes of $\top_R$, the latter a subtype of the three others, then adding CO-IDENT as a subtype of EQUIVALENCE, and CO-REF as a subtype of CO-IDENT. $\theta(G)$ is obtained by adding a relation node typed CO-REF between nodes in the same co-reference class (this simulation is linear, we only need to generate $n-1$ nodes for a co-reference class of size n). It remains now to handle individual markers. The mapping Θ_3 is an extension of Θ_2. The graph obtained with Θ_3 contains one node labelled m for each individual marker m in $\mathcal{I}$ (required in Sect. 6.3), all linked to a unique node labelled MARKERS. Each node labelled CONCEPT obtained from an individual node marked m is linked by a node labelled REFERENT to the node representing m. In Θ_3^-, only the most specific types and the individual markers present in the graph are represented.

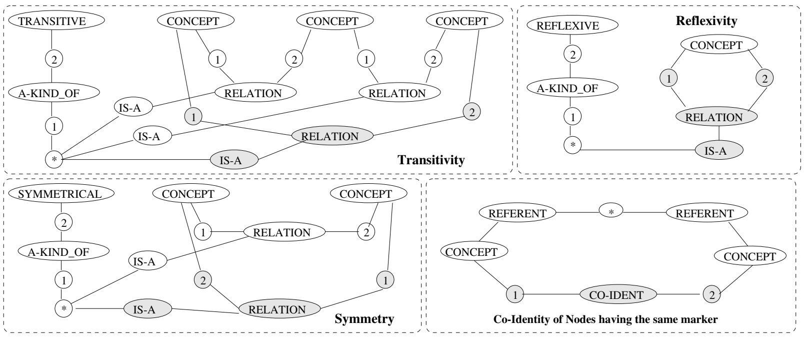

Fig. 7. Library $\mathcal{R}_c$ of LG-rules defining the co-identity relation

Fig. 7 defines the set of rules $\mathcal{R}_c$: the three first rules ensure that, for every relation type declared as a subtype of EQUIVALENCE (and in particular CO-REF and CO-IDENT), this relation behaves as an equivalence relation. The last one indicates that concept nodes having the same marker are co-ident. These rules are sufficient to simulate the (SG, $\models$) model, i.e.: $\mathcal{S}, G \models H$ iff $\Theta_3(\theta(G), \theta(\mathcal{S})), \mathcal{R}_s \cup \mathcal{R}_c \models \Theta_3^-(\theta(H))$ (proof of this assertion is given by Prop. 2, pointing out that these rules are a “higher-order version” of the ones presented in [2]).

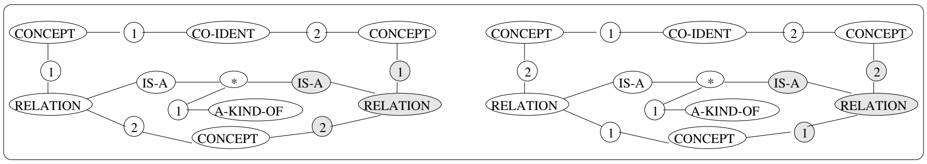

Fig. 8. Library $\mathcal{R}_n$ of LG-rules handling the normalization process

To simulate the normalization step for deduction in the (SG, $\vDash_{NF}$) model, we add another set of rules, $\mathcal{R}_n$ (Fig. 8). Again, thanks to Prop. 2, by verifying that these rules are a "higher-order version" of the ones presented in [2], we prove that $\mathcal{S}, G \vDash_{NF} H$ iff $\Theta_3(\theta(G), \theta(\mathcal{S})), \mathcal{R}_s \cup \mathcal{R}_c \cup \mathcal{R}_n \vDash \Theta_3^-(\theta(H))$. □

6.3 Introducing Rules

Theorem 3. *The* $\{$*LG, rules*$\}$*, (*$\{$*SG, rules*$\}$*,* $\vDash$*) and (*$\{$*SG, rules*$\}$*,* $\vDash_{NF}$*) models are strongly equivalent.*

Proof. Sect. 5 proves one part of the equivalence, we must now prove that the $\{$LG, rules$\}$ model is a strong generalization of both the ($\{$SG, rules$\}$, $\vDash$) and the ($\{$SG, rules$\}$, $\vDash_{NF}$) models. Let $\Upsilon(R)$ be a linear time mapping from SG rules into LG rules. Let $R = (G, color)$ be a SG rule defined on $\mathcal{S}$. $\Upsilon(R)$ is obtained from $\Theta_3^-(\theta(G))$ (the simulation of a SG into a LG) by assigning the color 0 to *all* nodes defining the support and to nodes representing the hypothesis of R, and the color 1 to all others (Fig. 9). This translation is designed in such a way that the "unique type and marker" invariant (see Rem. 1) is preserved by derivation. Let $\mathcal{S}$ be a support, G and H be two SGs defined on $\mathcal{S}$, we prove that:

1. $\mathcal{S}, \mathcal{R}, G \vDash H$ iff $\Theta_3(\theta(G), \theta(\mathcal{S})), \Upsilon(\mathcal{R}) \cup \mathcal{R}_s \cup \mathcal{R}_c \vDash \Theta_3^-(\theta(H))$
2. $\mathcal{S}, \mathcal{R}, G \vDash_{NF} H$ iff $\Theta_3(\theta(G), \theta(\mathcal{S})), \Upsilon(\mathcal{R}) \cup \mathcal{R}_s \cup \mathcal{R}_c \cup \mathcal{R}_n \vDash \Theta_3^-(\theta(H))$

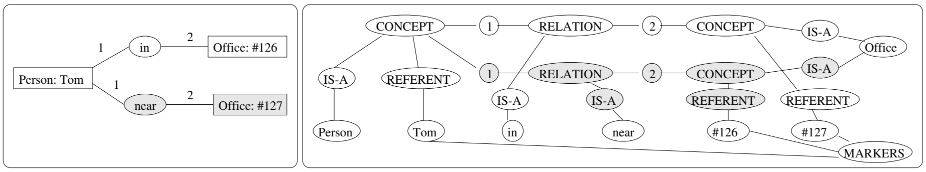

Fig. 9. A SG rule and the obtained transformation by Υ

Check that the application of a SG rule R on a SG G gives a SG G' iff the simulation of G derives the simulation of G' by some applications of the LG rules $\mathcal{R}_s \cup \mathcal{R}_c$, then an application of $\Upsilon(R)$. □

Corollary 1. *The* $\{$*Nested Graphs, rules*$\}$ *model is strongly equivalent to the* $\{$*LG, rules*$\}$ *model.*

Proof. [2] simulates of nested graphs into SGs, using a set $\mathcal{N}$ of SG rules. We compose this *linear* simulation with Θ_3, and translate $\mathcal{N}$ by Υ. □

7 Conclusion

In this paper, we have explored a basic projection-based reasoning model, the LG model, *a priori* less expressive than the SG model. However, we have proven that the {LG, rules} model is *strongly equivalent* to the {SG, rules} and even the {Nested graphs, rules} models. The simulation $\varTheta$ exhibited to translate an instance of {SG, rules} deduction problem into an instance of the {LG, rules} deduction problem possess all "good properties" discussed in Sect. 2:

1. $\varTheta$ is linear;
2. We prove, though only hints are included in this paper, there exists a *linear* (resp. *polynomial*) backward translation of objects (resp. proofs) of the {LG, rules} model into objects (resp. proofs) of the {SG, rules} model [3];
3. Since the two models are strongly equivalent, they are both in the same complexity class (i.e. deduction problems are *semi-decidable* in both models).

Let $\mathcal{M}$ be a reasoning model: the {LG, rules} model can be seen as an *interpreter* for $\mathcal{M}$ if it is provided with a simulation of $\mathcal{M}$, backward translations mechanisms, and a *library* of LG-rules that mimics specific reasonings of $\mathcal{M}$. But, having built an interpreter for the {SG, rules} and {NG, rules} models by designing a single, easier to implement model, what are the benefits of this simulation? We first discuss its added expressiveness, in terms of succintness and robustness, then show how the simulation can be used to extend the model.

7.1 The Succintness and Robustness Criteria

Let us first point out the "rules factorization" gained by the simulation. Though [2] gives SG rules for the same result, it requires: $3 \times k$ rules to indicate that $r_1, \ldots, r_k$ are equivalence relations; $|\mathcal{I}|$ rules to indicate that nodes sharing an individual marker are co-identical; $2 \times |\top_R|$ rules to simulate normalization.

Using our LG rules library, that use types and markers as "first-class objects" of the model, we need only a constant number of rules (11), before declaring $r_1, \ldots, r_k$ as subtypes of EQUIVALENCE.

But there is another benefit: using the rules defined in [2], one must add new rules each time a new type or a new marker is added to the support, otherwise reasonings are not complete. And keeping trace of all the rules that must be added when updating the support in a complex modelization soon becomes difficult. Now, reification allows to express properties on types and markers. These properties can be encoded in libraries of LG rules, which implement various SGs semantics, in such a way that it can be invisible to the end-user. The interest is that these properties can be *inherited*, and do not need to be encoded again by the end-user. Reification of types and markers indeed adds expressiveness, as well in terms of succintness (factorization of rules) as in terms of robustness (resistance to changes of data).

Finally, we point out that our simulation is complete only for LGs verifying the "unique type and marker" assumption (see Rem. 1). The problem is we have represented knowledge "two entities have the same type" by "the types of these two entities are represented by the same node". Should we want to overcome this limitation, we could define a "meta-relation" expressing that knowledge.

7.2 An Extended Simple Graphs Model

The SG model imposes strong syntactic conditions on co-reference links: for example, fusionning two concept nodes having different types or different individual markers is impossible in the model, hence the draconian constraints on SG rules, designed in such a way that never will co-referent nodes have different types or markers. But the deduction mechanism of the {LG, rules} model, applied to the simulation of a SG, ignores such constraints. Then a natural question is: "what happens if we remove all constraints on co-reference, and make reasonings on the simulation of these objects? How does the result of these reasonings translate back to the SG model?"

The first problem is to handle the case of co-identical nodes having different types or markers. The set of rules $\mathcal{R}_d$ presented in Fig. 10 are an answer to that problem. They indicate that if two nodes having type t and t' represent the same entity, then the type of this entity is some subtype of t and t'; and the marker of a node can be considered as a marker for all its co-ident nodes. We consider $\mathcal{R}$ the library of LG rules consisting of $\mathcal{R}_s \cup \mathcal{R}_c \cup \mathcal{R}_n \cup \mathcal{R}_d$. We can extend the {SG, rules} model by dropping off all constraints on co-reference, and simulate it (using the transformation Θ) into the {LG, rules} model with the library $\mathcal{R}$.

Let us now outline an extended simple graph (ESG) model, that allows one to translate back those reasonings. An ESG $G = (V_C, V_R, E, \text{label}, \text{co-ref})$, defined on a support $\mathcal{S}$, can be seen as a simple graph where:

- the type of a concept node is a non-empty *subset* of T_C;
- the marker of a concept node is a *subset* of $\mathcal{I}$ or the generic marker $*$;
- there is no constraint on concept nodes that can be declared co-referent;
- two *individual* concept nodes x, y are co-ident if $\text{marker}(x) \cap \text{marker}(y) \neq \emptyset$.

Intuitively, a set of types $\{t_1, \ldots, t_k\}$ (as used, by example, in [1]) can be seen as the conjunction of types $t_1 \sqcap \cdots \sqcap t_k$, and a set of individual markers as aliases for the same entity. The partial ordering $\leq_e$ on labels is used for projection:

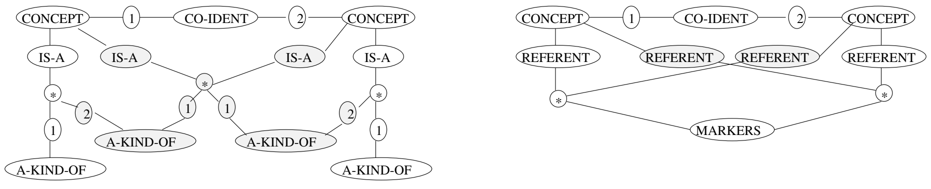

Fig. 10. Library $\mathcal{R}_d$ of LG-rules handling co-identity without restrictions

- Let t and t' be two types. Then $t \leq_e t'$ iff $\forall t'_i \in t'$, $\exists t_j \in t$ such that $t_j \leq t'_i$.
- Let m and m' be two individual markers. Then $m \leq_e m'$ (and $m' \leq_e m$) iff $m \cap m' \neq \emptyset$. Moreover, for every marker m, $m \leq_e *$.

An ESG is in *normal form* if all its co-identity classes are trivial. An ESG is put into normal form by fusionning co-identical nodes. The label resulting from the fusion of `[t: m]` and `[t': m']` is `[t ∪ t': M]`, where $M = m \cup m'$ if both markers are individual, $*$ if both are generic, otherwise M is the individual marker m or m'. The deduction system in the {ESG, rules} model is based upon *normal derivation*. Sound and complete FOL semantics can be given, using the semantics of the equivalent {LG, rules} model, or directly extending Φ (see [3]).

This example shows that, not only simulation adds robustness with respect to changes in data (e.g. changes in the support, as discussed above) but also with respect to changes in the model. We believe that a {LG, rules}-based interpreter, provided with macros that describe syntactic translations of graphs, can be the basis of a good prototyping and development tool for different graph-based reasoning models.

References

1. Franz Baader, Ralf Molitor, and Stephan Tobies. Tractable and decidable fragments of conceptual graphs. In *Proc. of ICCS'99*, pages 480–493. Springer, 1999.
2. Jean-François Baget. A Simulation Of Co-Identity with Rules in Simple and Nested Graphs. In *Proc. of ICCS'99*, pages 442–455. Springer, 1999.
3. Jean-François Baget. Extending the CG Model by Simulations. Research report, LIRMM, 2000.
4. Michel Chein and Marie-Laure Mugnier. Conceptual Graphs are also Graphs. Research Report, 1995.
5. G. Gogic, H. Kautz, C. Papadimitriou, and B. Selman. The Comparative Linguistics of Knowledge Representation. In *Proc. of IJCAI'95*, pages 862–869, 1995.
6. Gary G. Hendrix. Encoding Knowledge in Partitioned Networks. In *Associative Networks*, pages 51–92. Academic Press, 1979.
7. D. Kayser. *La représentation des connaissances*. HERMES, Paris, 1997.
8. E. Salvat and M-L. Mugnier. Sound and Complete Forward and Backward Chainings of Graph Rules. In *Proc. of ICCS'96*. Springer, 1996.
9. Stuart C. Shapiro. A net structure for semantic information storage, deduction and retrieval. In *Advance Papers of IJCAI'71*, pages 512–523, 1971.
10. G. Simonet. Two FOL Semantics for Simple and Nested Conceptual Graphs. In *Proc. of ICCS'98*. Springer, 1998.
11. John. F. Sowa. *Conceptual structures : Information processing in mind and machine*. Addison-Wesley, 1984.