RWTH
LTCS–Report

Aachen University of Technology
Research group for
Theoretical Computer Science

The Guarded Fragment of Conceptual Graphs

Franz Baader Ralf Molitor Stephan Tobies

LTCS-Report 98-10

RWTH Aachen
LuFg Theoretische Informatik
http://www-lti.informatik.rwth-aachen.de

Ahornstr. 55
52074 Aachen
Germany

The Guarded Fragment of Conceptual Graphs*

Franz Baader† Ralf Molitor† Stephan Tobies†

December 30, 1998

Abstract

Conceptual graphs (CGs) are an expressive and intuitive formalism, which plays an important role in the area of knowledge representation. Due to their expressiveness, most interesting problems for CGs are inherently undecidable. We identify the syntactically defined guarded fragment of CGs, for which both subsumption and validity is decidable in deterministic exponential time.

1 Outline

We start by giving a formal definition of conceptual graphs. Based on this definition, we define the operator Φ, which maps CGs into the first order predicate calculus with equality. Our definition of Φ uses the same ideas already presented in [Sow84], but we correct some minor mistakes which were already pointed out in [Wer95a].

We show that the full set of CGs corresponds to the set of all first order formulae, by showing that for each formula φ there is an (effectively computable) CG G_φ such that $\Phi(G_\varphi) \equiv \varphi$.

Starting from the definition of the guarded fragment of first order logic, we identify a corresponding fragment of conceptual graphs, which can be defined solely by syntactic means. We cite some results on the decidability of the guarded fragment which induce the decidability of both validity and subsumption for the guarded fragment of CGs. The (relatively efficient) decidability of theses problems makes it possible to use guarded graphs in fully automatic graph provers.

2 Basic definitions

Firstly we will give the formal definition of conceptual graphs, which we will use in the remainder of this report. Starting point for the definition is the

*Part of this work was supported by the DFG

†Research Group for Theoretical Computer Science, University of Technology Aachen.

support, which is used to code basic ontological knowledge. We use the standard definition, which, for example, is also presented in [CMS98].

Definition 1 (Support)
A support *is a tuple* $\mathcal{S} = \langle \mathcal{T}_C, \mathcal{T}_R, N_I \rangle$ *where*

- $\mathcal{T}_C = (N_C, \leq_C)$ *is the* concept type hierarchy *of* $\mathcal{S}$. N_C *is a set of names of concept types which contains a distinguished type* $\top_C$, *and* $\leq_C$ *is a partial order on* N_C *with greatest element* $\top_C$.
- $\mathcal{T}_R = \langle (N_R^1, N_R^2, \dots), \leq_R \rangle$ *is the* relation type hierarchy *of* $\mathcal{S}$. N_R^i *contains the relation symbols of arity* i. *We require* $N_R^i \cap N_R^j = \emptyset$ *if* $i \neq j$. *We define* $N_R = \bigcup_{i \in \mathbb{N}} N_R^i$. $\leq_R$ *is a partial order on* N_R *for which relation types of different arity must be incomparable.*
- *The set of* individual markers *or* constants *is denoted by* N_I; *the* generic marker *is denoted by* $*$. *For convenience reasons we also define a partial order* $\leq_I$ *on* $N_I \cup \{*\}$ *by requiring* $*$ *to be the greatest element of* $\leq_I$ *and all other elements to be pairwise incomparable.*

Throughout this report, we consider examples over the support shown in Fig. 1, where all relation types are assumed to have arity 2.

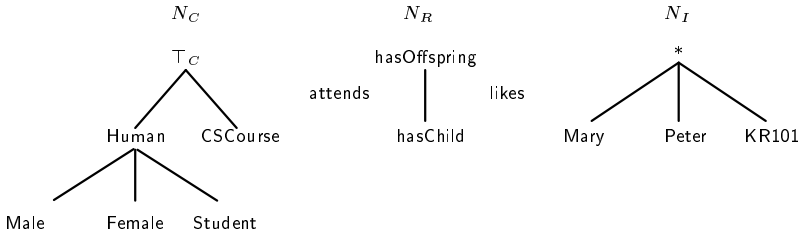


Figure 1: An example support

Based on this definition we now define simple graphs, which form the basic building blocks of CGs.

Definition 2 (Simple Graph)
Let $\mathcal{S}$ *be a support. A* simple graph *(SG) is a tuple* $g = \langle C, R, E, \ell \rangle$, *where*

- $\langle C, R, E \rangle$ *is an undirected bipartite graph with node sets* C *and* R *and edge relation* $E \subseteq C \times R$. *The sets* C *and* R *are called the sets of* concept nodes *and* relation nodes *of* g *respectively.*
- *The function* ℓ *labels* (C, R, E) *in the following way:*

$$\begin{aligned} \ell : R &\to N_R \\ C &\to N_C \times I \\ E &\to 2^{\mathbb{N}} \setminus \{\emptyset\} \end{aligned}$$

ℓ must satisfy: $\forall r \in R : \ell(r) \in N_R^n \Rightarrow \{\ell(c,r) \mid (c,r) \in E\}$ *is a disjoint partition of* $\{1,\ldots,n\}$.

For a concept node c with $\ell(c) = (type(c), ref(c))$, $type(c)$ and $ref(c)$ are called the type *and* referent *of c respectively.*

With $SG(\mathcal{S})$ we denote the set of all simple graphs over $\mathcal{S}$.

The condition on ℓ states that if a relation node r is labeled with a relation type $\ell(r)$ of arity n, then each element of $\{1,\ldots,n\}$ appears exactly once in the labels of the edges attached to r. This enables us to define the jth neighbour of a relation node r to be the concept node c such that $(c,r) \in E$ and $j \in \ell(c,r)$. For short we denote this concept node by $r(j)$.

Simple graphs have a straightforward graphical representation, by just drawing the graph and attaching the labels to the different parts of the graph. We choose the convention that concept nodes are drawn using a rectangular box and relation nodes are depicted by ovals both containing the labeling. Edges are drawn undirected and are labeled with their set of numbers.

Examples of simple graphs are the graphs inside the boxes of Fig.

SGs can be combined into more complex structures called graph propositions.

Definition 3 (Graph Proposition)
Let $\mathcal{S}$ be a support. A graph proposition *over $\mathcal{S}$ is either a negated graph proposition or a box which contains a SG (which may be empty) and finitely many graph propositions. These boxes are also called the* contexts *of the propositions. We require that all simple graphs appearing in a graph proposition have disjoint node sets. In a linear notation one can think of graph propositions as of the expressions generated by the EBNF grammar*

$$p ::= \left[g\ p^*\right] \mid \neg p\ .$$

where g stands for a SG over $\mathcal{S}$.

A subproposition *of p is a box p' which is directly contained in p.*

We say that a context p (strictly) dominates *a context q iff q is (strictly) contained in p.*

The set of concept nodes $C(p)$ (relation nodes $R(p)$, edges $E(p)$) of a graph proposition p is the union of all concept nodes (relation nodes, edges) appearing in the simple graphs of the graph proposition p.

For each SG g in p there is exactly one context p' which contains g at top level. This context is called the context of g, and we say that p' *contains* all nodes of g.

For a graph proposition of the form $p = \left[g\ p_1 \ldots p_n\right]$ where g is the empty simple graph, we also write $\left[\emptyset\ p_1 \ldots p_n\right]$.

Definition 4 (Conceptual Graph with coreference links)
Let $\mathcal{S}$ be a support. A conceptual graph over $\mathcal{S}$ is a pair $G = \langle p, coref \rangle$, where

1. *p is a graph proposition over* $\mathcal{S}$, *and*

2. *coref is a binary symmetric relation over* $C(p)$ *such that for each* $c_1, c_2 \in C(p)$: $(c_1, c_2) \in$ *coref implies that for the contexts* p_1, p_2 *which contain* c_1 *and* c_2 *respectively, either* p_1 *dominates* p_2 *or* p_2 *dominates* p_1. *coref is called the set of* coreference links.

A subgraph G' *of* G *is a conceptual graph* $G' = \langle p', coref' \rangle$, *where* p' *is a subproposition of* p *and* $coref'$ *is the restriction of coref to* $C(p')$.

By $CG(\mathcal{S})$ *we denote the set of all conceptual graphs over* $\mathcal{S}$.

A path $c_1, c_2, \ldots, c_n$ *such that* $(c_j, c_{j+1}) \in$ *coref for all* $1 \leq i < n$ *is called a* line of identity.

Note that the condition we impose on *coref* is also satisfied if c_1 and c_2 are contained in the same context, since each context dominates itself.

If $(c_1, c_2) \in$ *coref* and the context of c_1 (strictly) dominates the context of c_2 we say that also c_1 (strictly) dominates c_2.

An example of a CG built over the support from Fig. 1 is shown in Fig. 2 using the usual graphical elements. It asserts that each parent and child there is another parent of that child who likes the first parent. The extra markers p_i and c_i will be used in a latter section of this paper to refer to the different parts of the CG.

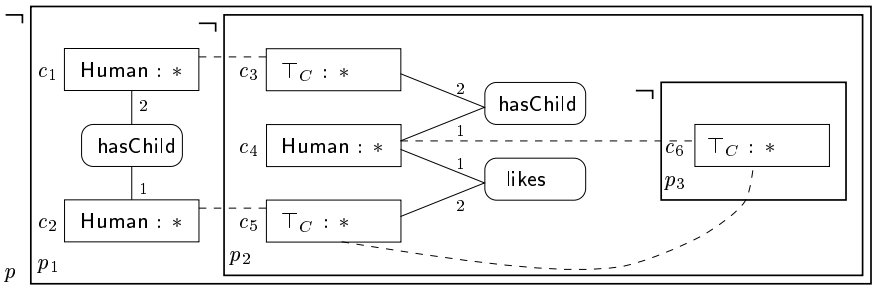


Figure 2: An example CG

The coreference links in CGs destroy their inductive structure and make them extremely difficult to handle in inductive proofs. Thus we introduce an syntactic variant of CGs with coreference links, in which variables are used to express the identity of concept nodes. Similar ideas can be found in [Sow84, KS97].

Definition 5 (Conceptual Graph with Variables)
Let $\mathcal{S}$ *be a support and* $\mathcal{V}$ *be a set of variables. We define* $\mathcal{V}_I = \mathcal{V} \cup N_I$. *We use the letters* s, t *possibly with subscript to range over the elements of* $\mathcal{V}_I$.

A conceptual graph with variables *over* $\mathcal{S}$ *and* $\mathcal{V}$ *is a tuple* $G = \langle p, id, links \rangle$ *such that*

1. *p is a graph proposition over* $\mathcal{S}$,

2. $id : C(p) \to \mathcal{V}_I$ *such that*

$$id(c) = \begin{cases} ref(c), & \text{iff } ref(c) \neq * \\ x \in \mathcal{V}, & \text{iff } ref(c) = * \end{cases}$$

and there are no two concepts c_1, c_2 *such that* $id(c_1) = id(c_2) = x \in \mathcal{V}$.

3. $links : C(p) \to \mathcal{P}(\mathcal{V}_I)$, *such that if* $x \in links(c) \cap \mathcal{V}$ *for a concept* c *there is a concept* c' *which dominates* c *and* $id(c') = x$.

With $varlinks(c)$ *we denote the set* $links(c) \cap \mathcal{V}$.

With $CG(\mathcal{S}, \mathcal{V})$ *we denote the set of all conceptual graphs with variables over* $\mathcal{S}$ *and* $\mathcal{V}$.

We use the following graphical representation for a concept node in a conceptual graph with variables: It is depicted by a labeled box. Inside the box we write the type and the referent of the node. On the left side under the box we write the value of *id* while on the right side we write the value of *links*. An example is given in Fig. 3.

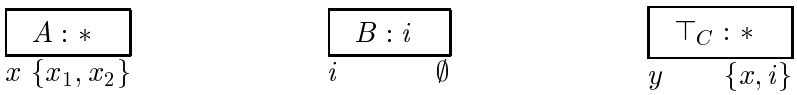


Figure 3: Examples of annotated concept nodes

Conceptual graphs with coreference links can be translated to conceptual graphs with variables using the algorithm in Assumption 4.2.6 in [Sow84].

Let $G = \langle p, coref \rangle$ be the conceptual graph with coreference links. Proceed as follows:

- Fix an arbitrary mapping *id* which satisfies Property 2 of Def. 5.
- Define $links(c) = \{id(c') \mid c' \text{ dominates } c\}$.

Coreference links are expressed using the mapping *links*. Note that Property 2 of Def. 4 guarantees that Property 3 of the previous definition is satisfied by this construction of *links*.

The translation of the graph G from Figure 2 into a conceptual graph with variables can be found in Figure 4.

We can reverse this translation as follows: Let $G = \langle p, id, links \rangle$ be a conceptual graph with variables. We define *coref* by

$$coref = \{(c, c') \mid id(c') \in links(c)\} \cup \{(c', c) \mid id(c') \in links(c)\}$$

For the reverse direction, Property 3 of the above definition ensures that the resulting relation *coref* satisfies Property 2 of Def. 4.

We call these translations the *canonical* translation from conceptual graphs with coreference links to conceptual graphs with variables and vice versa.

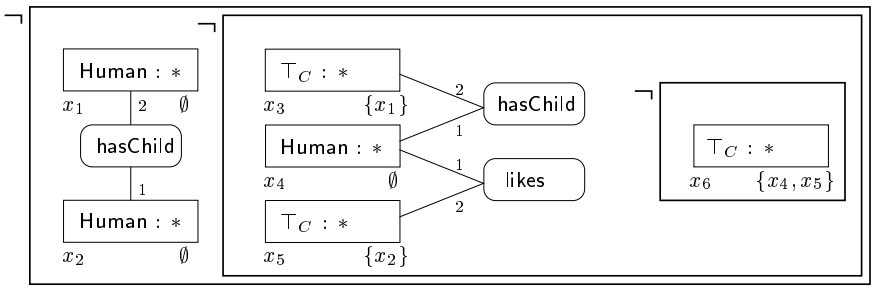


Figure 4: A conceptual graph with variables

3 A FO semantics for conceptual graphs

In his book [Sow84], Sowa gives a definition of the FO semantics of conceptual graphs by defining the Φ operator. Subsequently, Wermelinger [Wer95a] corrected some minor mistakes in Sowa's definition, but the definition he also takes into account higher order features of CGs which we are not interested in. After stripping off the higher order features and adapting his definition to our slightly more formal definition of CGs, one gets the following definition for the operator Φ:

Definition 6 (The operator Φ)
Let $\mathcal{S}$ be a support, $\mathcal{V}$ the set of variables, and $\Sigma_{\mathcal{S}}$ the first order signature which corresponds to $\mathcal{S}$ ($\Sigma_{\mathcal{S}} = N_C \cup N_R \cup N_I$). The operator Φ, which maps $CG(\mathcal{S}, \mathcal{V})$ to $FO(\Sigma_{\mathcal{S}})$ is defined by the following rules.

Let $G = \langle p, id, links \rangle$ be the graph to be translated.

1. *We define the translation of the conceptual graph inductively over the structure of the underlying graph proposition. Basic building block is the function ϕ which maps a simple graph to an FO formula.*

   Let $g = \langle C, R, E, \ell \rangle$ be a simple graph which occurs as a label in G. We define

   $$\phi(g) = \bigwedge_{c \in C} \phi(c) \wedge \bigwedge_{r \in R} \phi(r)$$

   For a concept node c we define

   $$\phi(c) = \bigwedge_{s \in links(c)} id(c) \doteq s \wedge \begin{cases} id(c) \doteq id(c) & \text{if } type(c) = \top_C \\ P(id(c)) & \text{if } type(c) = P \end{cases}$$

   For a relation node r with $\ell(r) = R \in N_R^j$ we define

   $$\phi(r) = R(id(r(1)), \ldots, id(r(j)))$$

The quantifier prefix $\phi_p(g)$ is defined by

$$\phi_b(g) = \exists x_1 \ldots \exists x_n$$

where $\{x_1, \ldots, x_n\} = \{id(c) \mid c \in C\} \cap \mathcal{V}$.

2. *Now we are able to define the translation function for a conceptual graph. Hence we define* Φ *inductively over syntactic structure of* p*:*

   - $\Phi\big[g\ p_1 \ldots p_m\big] := \phi_p(g).\Big(\phi(g) \wedge \bigwedge_{j=1,\ldots,m} \Phi(p_j)\Big)$
   - $\Phi\big[\neg p\big] := \neg\Phi\big[p\big]$

We use the convention that an empty conjunction of formulae is defined as *true*.

For the example graph $\widehat{G}$ from Figure 4, the operator Φ yields

$$\begin{aligned}\Phi(\widehat{G}) = \neg(\exists x_1 x_2.(&\mathsf{Human}(x_1) \wedge \mathsf{Human}(x_2) \wedge \mathsf{hasChild}(x_2, x_1) \\ \neg(\exists x_3 x_4 x_5.(&x_3 \doteq x_1 \wedge x_5 \doteq x_2 \wedge \mathsf{Human}(x_4) \wedge \\ &\mathsf{hasChild}(x_4, x_3) \wedge \mathsf{likes}(x_4, x_5) \wedge \\ \neg(\exists x_6.(&x_6 \doteq x_4 \wedge x_6 \doteq x_5))))\end{aligned}$$

where unnecessary subformulae of the form $x_i \doteq x_i$ have been eliminated. Note that the subformula $\neg(\exists x_6.(x_6 \doteq x_4 \wedge x_6 \doteq x_5))$ only expresses the inequality of x_4 and x_5.

For a conceptual graph G which conforms to Def. 5 the resulting formula will have no free variables. This is due to Property 3 of this definition which ensures that for each variable x which occurs in the set $links(c)$ for a concept c there is a concept c' for which $id(c') = x$ and the position of this concept in the graph ensures that x will only occur in the range of the quantifier which belongs to the concept c'.

Of course we can extend the domain of the operator Φ to the set of CGs with coreference links using the canonical translation: Let $G \in \mathrm{CG}(\mathcal{S})$ be a CG with coreference links and let $\widehat{G}$ be its corresponding CG with variable such that $\hat{G}$ is the canonical translation of G. We define $\Phi(G) := \Phi(\widehat{G})$. Note that the choice of the labeling id in the canonical translation has no impact on $\Phi(G)$ since it is a closed formula.

In the case of simple graphs, this translation is compatible to the definitions of a FO semantics given both in [Sow84] and [CMS98].

4 Graph Representation of FO Formulae

Once we have established a FO semantics for CGs, it is interesting to find out, which fragment of FO can be expressed with CGs. It turns out that the CGs exhaust the entire set of FO formulae and that it is possible to define an

efficiently computable function Ψ which has the property that $\Phi(\Psi(\alpha)) \equiv \alpha$ for each FO formula α.

To facilitate inductive proofs over the structure of FO formulae, we have to deal with free variables. The class of conceptual graphs, which is appropriate for these formulae, is called *conceptual graphs with free variables*, is obtained by weakening Property 3 of Def. 5 as follows:

3'. *links* : $C(p) \to \mathcal{P}(\mathcal{V})$, such that if $x \in$ *varlinks*(c) for a concept c in the context p and there is a concept c' in context p' with $id(c') = x$ then p' dominates p.

If the function Ψ is applied to a formula α with free variables the result will be a CG with free variables, while the result of the function Φ applied to a conceptual graph with free variables is a formula α with free variables.

Conceptual graphs with free variables now have the desired inductive structure we will need in the subsequent proofs. While a subgraph of a conceptual graph with variables might not be a such a graph itself (because it may violated Property 3), this is not the case for conceptual graphs *with free* variables. This is stated by the following lemma, the proof of which immediately follows from the definition of CGs with free variables.

Lemma 7
Let $G \in CG(\mathcal{S}, \mathcal{V})$ be a conceptual graph with free variables and let G' be a sub-graph of G. Then G' is a conceptual graph with free variables.

We will now define the function Ψ, which maps FO formulae to CGs with free variables:

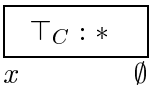
Definition 8
Let $\mathcal{S}$ be a support and $\Sigma_{\mathcal{S}}$ the corresponding first order signature. The graph representation Ψ,

$$\Psi : FO(\Sigma_{\mathcal{S}}) \to CG(\mathcal{S}, \mathcal{V})$$

is defined as follows:

We consider the formula α to be in a normal form which is built using only $\exists, \wedge$, and $\neg$, also we assume each variable to be bound exactly once. Any closed formula can easily be transformed into this normal form by renaming variables and applying simple syntactic transformations. We define $\Psi(\alpha)$ recursively over the structure of α:

- $\Psi(\exists x.\beta) = \left[g_\top\ \Psi(\beta)\right]$ *where $g_\top$ is the simple graph which looks as follows:*



- $\Psi(\beta_1 \wedge \beta_2) = \left[\emptyset\ \Psi(\beta_1)\ \Psi(\beta_2)\right]$.

- $\Psi(\neg\beta) = \neg\Psi(\beta)$.
- $\Psi(Px) = \big[g_P\big]$, *where* g_P *is a simple graph containing only a single concept node:*

$$g_A: \quad \underset{z}{\boxed{P:*}}_{\{x\}}$$

  z *is a fresh variable which does not occur anywhere in the formula nor somewhere else in the conceptual graph.*

- $\Psi(Pi) = \big[g_P\big]$, *where* g_P *is a simple graph containing only a single concept node:*

$$g_A: \quad \underset{i}{\boxed{P:i}}_{\emptyset}$$

- $\Psi(x \doteq y) = \big[g_\top\big]$, *with*

$$g_\top: \quad \underset{z}{\boxed{\top_C:*}}_{\{x,y\}}$$

  z *is a fresh variable which does not occur anywhere in the formula nor somewhere else in the conceptual graph.*

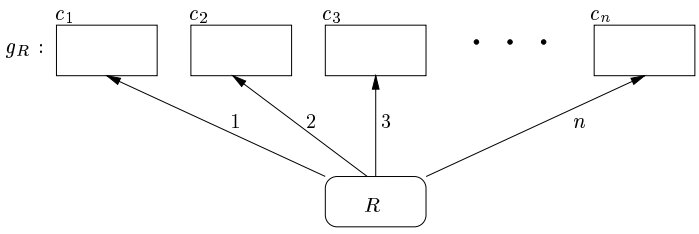
- $\Psi(x \doteq i) = \big[g\big]$, *with*

$$g: \quad \underset{i}{\boxed{\top_C:i}}_{\{x\}}$$

- $\Psi(i_1 \doteq i_2) = \big[g\big]$, *with*

$$g: \quad \underset{i_1}{\boxed{\top_C:i_1}}_{\{i_2\}} \qquad \underset{i_2}{\boxed{\top_C:i_2}}_{\{i_1\}}$$

- $\Psi(Rs_1 \ldots s_n) = \big[g_R\big]$, *where* g_R *has the following structure:*



  The structure of the concept nodes c_j *depends on whether* s_j *is a variable or an identifier. If* s_j *is the variable* x*, then it looks as follows:*

$$c_j: \quad \underset{z_j}{\boxed{\top_C:*}}_{x}$$

If s_j is the identifier i, then c_j has a different structure:

$$c_j : \begin{array}{|c|}\hline \top_C : i \\ \hline i \qquad\quad \emptyset \\ \hline \end{array}$$

The variables z_j are fresh variables which do not occur anywhere in the formula nor somewhere in else in the conceptual graph.

Please note that for a closed formula α the resulting graph structure $\Psi(\alpha)$ is indeed a proper conceptual graph with variables. The assumptions made about α being in a certain normal form, guarantee that Properties 2 and 3 of Def. 5 are satisfied.

The following theorem justifies the name *graph representation* for the function Ψ:

Theorem 9
Let $\mathcal{S}$ be a support and $\Sigma_{\mathcal{S}}$ the corresponding first order signature. For each formula $\alpha \in FO(\Sigma_{\mathcal{S}})$ (possibly with free variables)

$$\Phi(\Psi(\alpha)) \equiv \alpha.$$

Proof: Due to our definition of conceptual graphs with free variables we can show this theorem inductively over the structure of FO formulae. These are the bases cases of the induction. They all follow easily from the definition of the function ϕ which maps simple graphs to FO formulae and the definition of Ψ for the atomic formulae.

- $\Phi(\Psi(Px)) = \exists z.z \doteq x \wedge Pz \equiv Px.$
- $\Phi(\Psi(Pi)) = Pi.$
- $\Phi(\Psi(x \doteq y)) = \exists z.z \doteq x \wedge z \doteq y \wedge z \doteq z \equiv x \doteq y.$
- $\Phi(\Psi(x \doteq i)) = i \doteq x \wedge i \doteq i \equiv x \doteq i.$
- $\Phi(\Psi(i_1 \doteq i_2)) = i_1 \doteq i_2 \wedge i_1 \doteq i_1 \wedge i_2 \doteq i_1 \wedge i_2 \doteq i_2 \equiv i_1 \doteq i_2.$
- $\Phi(\Psi(Rs_1 \ldots s_n) \equiv Rs_1 \ldots s_n$ similar to the case for Pi and Px.

In all three cases the induction step follows very easily from the definitions of Φ and Ψ and by using the induction hypothesis:

- $\Phi(\Psi(\exists x.\beta)) \equiv \exists x.x \doteq x \wedge \Phi(\Psi(\beta)) \equiv \exists x.\beta$
- $\Phi(\Psi(\neg\beta)) = \Phi\big[\neg\Psi(\beta)\big] = \neg\Phi(\Psi(\beta)) \equiv \beta$
- $\Phi(\Psi(\beta_1 \wedge \beta_2)) = \Phi\big[\emptyset\ \Psi(\beta_1)\ \Psi(\beta_2)\big] = \mathit{true} \wedge \Phi(\Psi(\beta_1)) \wedge \Phi(\Psi(\beta_2)) \equiv \beta_1 \wedge \beta_2$

■

5 The Guarded Fragment of Conceptual Graphs

Definition 10 (The Guarded Fragment)
Let Σ be a FO signature. The guarded fragment *$GF(\Sigma)$ of first-order logic is defined inductively as follows:*

1. *Every atomic formula over Σ belongs to $GF(\Sigma)$.*
2. *$GF(\Sigma)$ is closed under the connectives $\neg, \wedge, \vee, \rightarrow$, and $\leftrightarrow$.*
3. *If $\mathbf{x}, \mathbf{y}$ are tuples of variables, $\alpha(\mathbf{x}, \mathbf{y})$ is atomic and $\beta(\mathbf{x}, \mathbf{y})$ is a formula in $GF(\Sigma)$, such that $free(\beta) \subseteq free(\alpha) = \{\mathbf{x}, \mathbf{y}\}$, then the formulae*

$$\exists \mathbf{x}.(\alpha(\mathbf{x}, \mathbf{y}) \wedge \beta(\mathbf{x}, \mathbf{y}))$$
$$\forall \mathbf{x}.(\alpha(\mathbf{x}, \mathbf{y}) \rightarrow \beta(\mathbf{x}, \mathbf{y}))$$

   belong to $GF(\Sigma)$.

The loosely guarded fragment *$LGF(\Sigma)$ is defined similarly to $GF(\Sigma)$ but we weaken the quantifier rule as follows:*

3'. *If $\mathbf{x}, \mathbf{y}$ are tuples of variables, $\beta(\mathbf{x}, \mathbf{y})$ is in $LGF(\Sigma)$, and $\alpha_1 \wedge \cdots \wedge \alpha_m$ is a conjunction of atoms, then*

$$\exists \mathbf{x}.((\alpha_1 \wedge \cdots \wedge \alpha_m) \wedge \beta(\mathbf{x}, \mathbf{y}))$$
$$\forall \mathbf{x}.((\alpha_1 \wedge \cdots \wedge \alpha_m) \rightarrow \beta(\mathbf{x}, \mathbf{y}))$$

   belong to $LGF(\Sigma)$, provided that for every variable x_i and every variable $z \in \{\mathbf{x}, \mathbf{y}\}$ there is an atom α_j such that x_i and z occur in α_j.

$LGF(\Sigma)$ (and hence GF(Σ)) is decidable. There is also an exact result for the complexity of the decision problem:

Fact 11 ([Grä98])
*The satisfiability problem both for GF and LGF is 2-*ExpTime *complete.*

If we restrict the signature Σ we get a considerably lower complexity:

Fact 12 ([Grä98])
Let Σ be a signature with a bound on the maximum arity on the relation symbols which appear in Σ. Then the satisfiability problem both for $GF(\Sigma)$ and $LGF(\Sigma)$ is ExpTime *complete. This is particularly the case if Σ is finite.*

The definition of the (loosely) guarded fragment of FO gives rise to the definition of a corresponding fragment of CGs; we will call this fragment the *(loosely) guarded fragment of CGs*. The restrictions defining this fragment guarantee

that all quantifiers in the FO translation of a guarded graph can either be eliminated, or are loosely guarded in the sense of Def. 10. The same must apply to any variable appearing free at any place in the FO translation of a guarded graph. To state the restrictions on the graphs, we have to identify the nodes representing free and bound variables in the contexts of a graph. These will be the *new* and *external* nodes from the next definition.

Definition 13
Let $G = \langle p, coref \rangle$ be a CG over $\mathcal{S}$. A concept node $c \in C(p)$ which is contained in a context q of p is called external *iff it is has a coreference link to a strictly dominating concept. It is called* old *iff it satisfies one of the following:*

- *c is an external or an individual concept node.*
- *c is linked by a coreference link to another old node in q.*

Nodes which are not old are called new.

In our example CG from Fig. 2 c_3, c_5, c_6 are external nodes while c_1, c_2, c_4 are new nodes. Note that c_4 is a new node even though linked by a chain of coreference links to the old node c_5. This is desired because coreference links inside a context express equality of concept node, while the coreference links from c_4 and c_5 to c_6 are used to express inequality of c_4 and c_5.

Definition 14 (The Guarded Fragment of Conceptual Graphs)
A CG $G = \langle p, coref \rangle \in CG(\mathcal{S})$ is called guarded *iff it satisfies the following:*

1. *For any two concept nodes c_1, c_2 contained in the contexts p_1, p_2: If $(c_1, c_2) \in coref$ and p_1 strictly dominates p_2, then for each context q such that q lies between p_1 and p_2 (i.e. p_1 strictly dominates q and q strictly dominates p_2) it holds that q is labeled by a simple graph g which contains no new nodes.*
2. *For each simple graph $g = \langle C, R, E, \ell \rangle$ labeling a context of G either g contains no new nodes or g satisfies the following: If $C = \{c\}$, then $type(c) \neq \top_C$ or there is an $r \in R$ such that $(c, r) \in E$. If C contains more then one node, then there is an $r \in R$ such that $C = \{c \mid (c, r) \in E\}$. In this case g is called* guarding.

The CG G is called loosely guarded, *if it satisfies Property (1) for gCGs and instead of (2) it satisfies:*

2' *For each simple graph $g = \langle C, R, E, \ell \rangle$ labeling a context of G either g contains no new nodes or g satisfies the following: If $C = \{c\}$, then $type(c) \neq \top_C$ or there is an $r \in R$ such that $(c, r) \in E$. If C contains more then one node, the for each pair $c, d \in C$ where $c \neq d$ and c is new and d is not an individual concept node, there is an $r \in R$ such that $\{(c, r), (d, r)\} \subseteq E$. In this case g is called* loosely guarding.

With $gCG(\mathcal{S})$ ($lgCG(\mathcal{S})$) we denote the set of all (loosely) guarded conceptual graphs over $\mathcal{S}$.

Both restrictions made in the above definition are necessary for the resulting fragment to correspond to the loosely guarded fragment of FO. An important example for an assertion which can not be expressed by a loosely guarded formula is transitivity of a binary relation symbol [Grä98]. Figure 5 shows two graphs that assert the transitivity of the binary relation hasOffspring. The upper graph is not loosely guarded because it violates Property (2), since c_1 and c_3 are both new nodes and are not adjacent to the same relation node. The lower graph is not loosely guarded because it violates Property 1, since c_4 and c_5 are linked by a coreference link which spans the context which contains the new node c_5. The two graphs correspond to the FO formulae

$$\varphi_1 = \neg(\exists x \exists y \exists z.((\mathsf{hasOffspring}(x,y) \wedge \mathsf{hasOffspring}(y,z)) \wedge \neg\mathsf{hasOffspring}(x,z)))$$
$$\varphi_2 = \neg(\exists x \exists y.(\mathsf{hasOffspring}(x,y) \wedge (\exists z.(\mathsf{hasOffspring}(y,z) \wedge \neg\mathsf{hasOffspring}(x,z))))) \ .$$

Note that even though the definition of (l)gCGs looks quite complex at first sight, it is a purely syntactical definition using easily testable properties of graphs.is a purely syntactic definition in the sense that it makes only use of properties of *G as a graph*. Indeed it is possible to decide in polynomial time whether a given CG is a gCG or a lgCG.

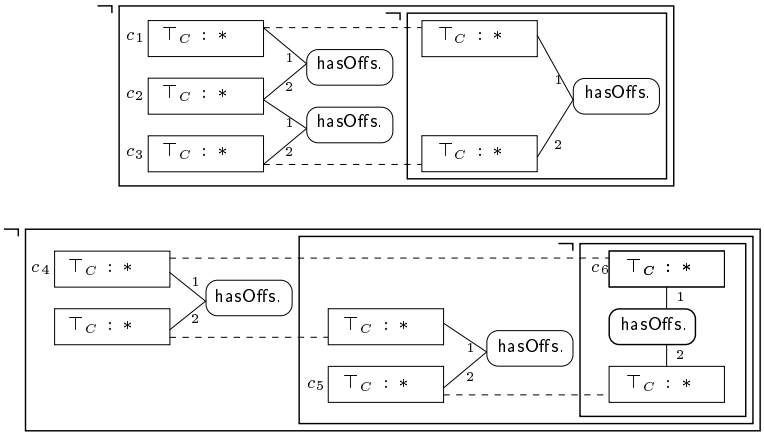


Figure 5: Two graphs that are not loosely guarded.

For the subsequent proofs we will need the fragment of conceptual graphs with (free) variables which corresponds to the guarded fragment of conceptual graphs with coreference links:

Definition 15
Let $G = \langle p, id, links\rangle \in CG(\mathcal{S}, \mathcal{V})$ be a CG with free variables. We define

$$\rightleftharpoons := \{(c_1, c_2) \mid id(c_2) \in links(c_1)\} \cup \{(c_1, c_2) \mid id(c_1) \in links(c_2)\}$$

A concept node $c \in C(p)$ which is contained in a context q of p is called external, *if there is an $s \in links(c)$ such that for all concepts c' in the same context q it holds that $id(c') \neq s$. It is called* old *iff it satisfies one of the following:*

- *c is an external or an individual concept node.*
- *There is an old node c' in the context q such that $c \rightleftharpoons c'$.*

Nodes which are not old are called new.

Definition 16
For a CG $G = \langle p, id, links \rangle \in CG(\mathcal{S}, \mathcal{V})$ we define.

$$\rightleftharpoons := \{(c_1, c_2) \mid id(c_2) \in links(c_1)\} \cup \{(c_1, c_2) \mid id(c_1) \in links(c_2)\}$$

The graph G is called guarded *iff it satisfies the following:*

1. *For any two concept nodes c_1, c_2 contained in the contexts p_1, p_2: If $c_1 \rightleftharpoons c_2$ and p_1 strictly dominates p_2, then for each context q such that q lies between p_1 and p_2 (i.e. p_1 strictly dominates q and q strictly dominates p_2) it holds that q is labeled by a simple graph g which contains no new nodes.*
2. *For each simple graph $g = \langle C, R, E, \ell \rangle$ labeling a context of G either g contains no new nodes or g satisfies the following: If $C = \{c\}$, then $type(c) \neq \top_C$ or there is an $r \in R$ such that $(c, r) \in E$. If C contains more then one node, then there is an $r \in R$ such that $C = \{c \mid (c, r) \in E\}$. In this case g is called* guarding.
3. *If $y \in links(c)$ for a concept c in context q and there is no concept $c' \in C(p)$ with $id(c') = y$, then there is no context q' which strictly dominates q and is labeled by a simple graph which contains a new node.*

The CG G is called loosely guarded, *if it satisfies Property (1) and (3) for gCGs and instead of (2) it satisfies:*

2' *For each simple graph $g = \langle C, R, E, \ell \rangle$ labeling a context of G either g contains no new nodes or g satisfies the following: If $C = \{c\}$, then $type(c) \neq \top_C$ or there is an $r \in R$ such that $(c, r) \in E$. If C contains more then one node, the for each pair $c, d \in C$ where $c \neq d$ and c is new and d is not an individual concept node, there is an $r \in R$ such that $\{(c, r), (d, r)\} \subseteq E$. In this case g is called* loosely guarding.

With $gCG(\mathcal{S}, \mathcal{V})$ ($lgCG(\mathcal{S}, \mathcal{V})$) we denote the set of all (loosely) guarded conceptual graphs with variables over $\mathcal{S}$.

Def. 14 and Def. 16 are very similar. Indeed, they express the same thing in the following sense:

Lemma 17
Let $G = \langle p, coref \rangle$ be a conceptual graph with coreference links and let $\widehat{G} = \langle p, id, links \rangle$ be an conceptual graph with variables such that $\widehat{G}$ is the canonical translation of G. Then G is (loosely) guarded in the sense of Def. 14, iff $\widehat{G}$ is (loosely) guarded in the sense of Def. 16.

Proof: Since $\widehat{G}$ is the canonical translation of G, it follows that $\rightleftharpoons = \mathit{coref}$. Hence c is an individual concept node in G iff it is an individual concept node in $\widehat{G}$. c is new (external) in G iff it is new (external) in $\widehat{G}$.

Also the Properties 1 and 2 of Def. 14 directly correspond to the Properties 1 and 2 of Def. 16. The only problem is Property 3 of Def. 16, but this property is trivially satisfied by any conceptual graph with variables (but without free variables) due to Property 3 of Def. 5. ∎

The set of (loosely) guarded annotated graphs with free variables has the inductive structure we need for inductive proofs:

Lemma 18
Let $G = \langle p, id, links\rangle, G' = \langle p', id', links'\rangle$ be a conceptual graph with free variables such that G' is a subgraph of G. If G is (loosely) guarded, then so is G'.

Proof: If $c \in C(p')$ is new in G' then it is new in G. Also if c is external in G' then it is external in G. Hence the Properties 1 and 2 of Def. 16 must be satisfied for G', otherwise they would also be violated in G.

Now assume Property 3 to be violated in G'. This implies that there is a concept node c in context q strictly dominated by context q' such that $y \in \mathit{varlinks}(c)$, $id(c') \neq y$ for all $c' \in C(p)$ and q' is labeled by a simple graph which contains a new node. This implies that in G there can not be a concept node c' in the outermost context such that $y = id(c')$, because otherwise Property 1 of Def. 16 would be violated. This implies that also $y \notin \{id(c') \mid c' \in C(p)\}$ and hence Property 3 is also violated by G which is a contradiction.

The next theorem justifies the name *guarded fragment* of conceptual graphs:

Theorem 19
Let $G \in (l)gCG(\mathcal{S}, \mathcal{V})$ and $\varphi = \Phi(G)$. Then there is a formula $\varphi' \in (L)GF(\Sigma_{\mathcal{S}})$ such that $\varphi' \equiv \varphi$. Furthermore φ' is computable from φ in polynomial time.

Proof: We will show this theorem for conceptual graphs with free variables to facilitate induction. We will use induction over the depth of p.

To simplify the induction base we assume that for each context q in G which is a leaf in p it holds that q is labeled by an empty simple graph. For this we replace all subpropositions of p of the form $[g]$ by $[g[\emptyset]]$. This does not change the semantics of the graph. Also the guardedness of G is not affected by this transformation, because an empty simple graph does not contain any new nodes.

For the base case of the induction we can now assume p to be of the form $[\emptyset]$. By the definition of Φ we get $\varphi = \Phi(G) = \mathit{true}$ because the quantifier prefix is empty as well as the conjunctions. This can of course be expressed by a guarded formula: Let P be an arbitrary concept type and set $\varphi' = \forall x.Px \rightarrow Px$. φ' is

(loosely) guarded and $\varphi' \equiv \mathit{true}$ holds. This concludes the proof of the basis case for the induction.

We now show the induction step: There are two possibilities for p. If p is of the form $\neg q$, we are done, because $\Phi\big[\neg q\big] = \neg\Phi\big[q\big]$ which is guarded due to the induction hypothesis. Now assume p to be of the form $p = \big[g\ p_1 \dots p_n\big]$ with $g = \langle C, R, E, \ell\rangle$. By the definition of Φ we have

$$\Phi\big[g\ p_1 \dots p_n\big] = \phi_p(g).\Big(\phi(g) \wedge \bigwedge_{j=1,\dots,n} \Phi(p_j)\Big)$$

Lemma 18 yields, that all sub-graphs of G are (loosely) guarded. Hence by the induction hypothesis, there exist (loosely) guarded formulae φ'_j such that $\varphi'_j \equiv \Phi\big[p_j\big]$, so there is a (loosely) guarded formula γ such that

$$\gamma \equiv \bigwedge_{j=1,\dots,n} \Phi(p_j)$$

due to the closure properties of the (loosely) guarded fragment. We define:

$$\chi = \phi_p(g).(\phi(g) \wedge \gamma)$$

Obviously $\chi \equiv \varphi$ holds. The formula χ is of the form

$$\chi = \exists x_1 \dots \exists x_m \exists z_1 \dots \exists z_l.\psi(x_1, \dots, x_m, z_1, \dots, z_l, y_1, \dots, y_n)$$

where the variables are partitioned as follows:

$$\begin{aligned}
\mathcal{X} &:= \{x_1, \dots, x_m\} = \{\mathit{id}(c) \mid c \text{ is new in } g\} \cap \mathcal{V} \\
\mathcal{Z} &:= \{z_1, \dots, z_l\} = \{\mathit{id}(c) \mid c \text{ is not new in } g\} \cap \mathcal{V} \\
\mathcal{Y} &:= \{y_1, \dots, y_n\} = \{y \mid y \text{ is free in } \chi\}
\end{aligned}$$

Claim 1: If y is a free variable from γ, then $\exists y$ appears in the quantifier prefix $\phi_p(g)$.

Proof of Claim 1: A free variable y in γ stems from a node c in a sub-graph G' of G for which $y \in \mathit{links}(c)$ but $\mathit{id}(c') \neq y$ for each concept node in G'. Hence there must be a node c' in C such that $y = \mathit{id}(c')$, because g contains a new node and otherwise Property 3 of Def. 16 would be violated. Hence y appears in the quantifier prefix $\phi_p(g)$ which proves Claim 2. □

The reflexive transitive closure of the restriction of $\rightleftharpoons$ to nodes from C induces a equivalence relation $\approx$ on the set

$$V_I = \mathcal{Y} \cup \mathcal{Z} \cup N_I$$

by setting

$$\begin{aligned}
&c_1, c_2 \in C \text{ are old and } (c_1, c_2) \in (\rightleftharpoons|_C)^* \\
\Rightarrow &\forall s \in \mathit{links}(c_2).\mathit{id}(c_1) \approx s \wedge \mathit{id}(c_1) \approx \mathit{id}(c_1).
\end{aligned}$$

It is easy to see that $\approx$ is an equivalence relation. Hence it partitions V_I into classes $[s]_\approx = \{t \in V_I \mid s \approx t\}$.

Claim 2:

1. For each variable $y \in \mathcal{Y}$, the class $[y]_\approx$ contains an element from $\mathcal{Z} \cup N_I$.
2. For each variable $z \in \mathcal{Z}$, the class $[z]_\approx$ contains an element from $\mathcal{Y} \cup N_I$.

Proof of Claim 2:

1. From Claim 1 it follows that any free variable in χ cannot stem from γ, hence there must be a node $c \in C$ which causes y to occur in χ with $y \in \mathit{links}(c)$. For this c, it can not be the case that $\mathit{id}(c) \in \mathcal{X}$, because c is not a new node. Thus $\mathit{id}(c)$ is an element of $\mathcal{Z}$ or N_I.
2. Let $c \in C$ be the concept node with $\mathit{id}(c) = z$. We proof this claim by induction on the length of a path from c to an external or individual concept node. The claim holds obviously for the identifier of any external node and individual concept node. A node which is not an external or individual concept node, there exists an old node c' with $c \rightleftharpoons c'$ and c' is old. Since c' is one step closer to an external or individual concept node, we can use the induction hypothesis which yields an $s \in \mathcal{Y} \cup N_I$ such that $s \in [\mathit{id}(c')]_\approx$. Since $c \rightleftharpoons c'$ we have $s \in [z]_\approx = [\mathit{id}(c')]_\approx$. □

For each class which contains an element from $\mathcal{Y} \cup \mathcal{Z}$ we arbitrarily select one element as its representative, if possible an element from N_I. For each variable $x \in \mathcal{Y} \cup \mathcal{Z}$ we denote the chosen representative of its class $[x]_\approx$ by $\hat{x}$. By construction of $\approx$, $s \approx t$ implies $\psi \models s \doteq t$ and hence the formula

$$\chi' := \bigwedge_{y_j \approx y_k} y_j \doteq y_k \wedge \exists x_1 \ldots \exists x_m \psi(x_1, \ldots, x_m, \hat{z}_1, \ldots, \hat{z}_l, \hat{y}_1, \ldots, \hat{y}_n)$$

is equivalent to χ. We denote $\psi(x_1, \ldots, x_m, \hat{z}_1, \ldots, \hat{z}_l, \hat{y}_1, \ldots, \hat{y}_n)$ by ψ'. Note that all quantifiers for elements of $\mathcal{Z}$ could be eliminated, because these variables do no longer occur in the formula ψ' due to Claim 2.1. Also note that all equalities between elements of $\mathcal{Y}$ appear outside the scope of the quantifier prefix. Hence it is sufficient if we show that $\exists x_1 \ldots \exists x_m \psi'$ is a (loosely) guarded formula.

If g contains no new nodes, then we are finished at this stage, because $\mathcal{X} = \emptyset$ and hence we have eliminated all new quantifiers. This implies that χ' is guarded, because the substitution of variables does not effect the guardedness of ψ.

If g contains new nodes, then we have to show that the remaining quantifiers for variables from $\mathcal{X}$ are (loosely) guarded in the sense of Def. 10. We finish the proof of the induction step by showing that ψ' is (loosely) guarded. The first alternative is similar in both the loosely guarded and the guarded case. Assume $C = \{c\}$ with $\mathit{type}(c) = A \neq \top_C$. There are two possibilities for c:

- c is an external node. Then the quantifier for $\mathit{id}(c)$ has been eliminated in the step from χ to χ'. Since $\phi_p(g)$ consists only of this quantifier, ψ' does not contain any quantifiers apart from those which appear in γ and hence are (loosely) guarded. Hence ψ' is (loosely) guarded.

- If c is not external, then the only sources for free variables in ψ' can be free variables from γ. Claim 1 says that there is no free variable in γ which is not quantified by $\phi_p(g)$. Hence there are no free variables from $\mathcal{Z}$ in ψ'. This implies that the only free variable in ψ' might be $id(c)$ which is properly guarded in ψ' by the guarding expression $A(id(c))$.

The second alternative has to be handled differently for the guarded and the loosely guarded case. Firstly we will consider the guarded case. C contains more than one node. Then there must be an $r \in R$ such that

$$C = \{c \mid (c, r) \in E\}.$$

We claim that then quantifier prefix is guarded in ψ' by the atom α which corresponds the the conjunct $\phi(r)$ in the original formula ψ after the substitution step. Assume α does not guard the quantifiers. This can have two reasons:

- There is a bound variable x_j which occurs in the quantifier prefix but does not occur in α. Since x_j is still present in ψ' it must be the identifier of a new node in C and hence x_j occurs in $\phi(r)$ as well as in α which is a contradiction to the assumption.
- There is a free variable y which occurs in ψ' but does not occur in α. Since y still occurs in ψ' it must be the chosen representative of its class $[y]_\approx$. But construction $[y]_\approx$ can not contain a elements $i \in N_I$, because otherwise we would have chosen i to be the representative of $[y]_\approx$. Thus $[y]_\approx$ contains the identifier z of an external node c that is adjacent to r. This implies that z appears in $\phi(r)$ and thus y appears in α, because we have replaced z by y.

Both cases lead to contradictions and hence we have proved that α is a guard for the quantifiers in ψ'.

Secondly we deal with the loosely guarded case: Since G is loosely guarded, for each $c, d \in C$ with $c \neq d$, c is new, and d is not an individual concept node, there is an $r \in R$ with

$$\{(c, r), (d, r)\} \subseteq E.$$

Let R_{guard} be the set of these relation nodes and let $\alpha_1 \wedge \cdots \wedge \alpha_\mu$ be the conjuncts which correspond to the formulae $\phi(r)$ for $r \in R_{\text{guard}}$ after the substitution. We claim that the quantifier prefix in χ' is loosely guarded by these conjuncts. Assume there is a x_j in the prefix which is not properly guarded by these conjuncts. Since x_j has not been eliminated in the substitution step, it must belong to a new node c. There are two possibilities, why x_j is not properly guarded.

- There is another variable x bound by the quantifier prefix, which does not coexists with x_j in any of these guards. Since x is still in the quantifier prefix, it must also belong to a new node $d \in C$. For the pair c, d there

must be an $r \in R$ such that $\{(c,r),(d,r)\} \subseteq E$ hence $r \in R_{\text{guard}}$ by the definition of R_{guard}. This implies that x_j and x coexists in $\phi(r)$ after the substitution (since no variables which identify new nodes are effected) which we defined to be the guards of the quantifiers.

- There is a variable y occuring free in ψ' which does not coexists with x_j in any of these guards. As in the guarded case this implies the the existence of an external node $d \in C$ such that $y \in \mathit{links}(d)$. $d \in C$ implies the existence of relation node $r \in R_{\text{guard}}$ such that x_j and y coexists in the guard which corresponds to the formula $\phi(r)$ after the substitution.

In either case we have shown that the assumption, that x_j is not properly guarded, leads to a contradiction and hence x_j must be properly guarded.

In Def. 16 there is an alternative way for g to be a loosely guarding simple graph. If C contains only a single node and there is an $r \in R$ such that $(c,r) \in E$. But this implies that g is a guarding simple graph, because $C = \{c\} = \{c \in C \mid (c,r) \in E\}$.

This finishes the proof of the base case since we have shown that χ' is indeed (loosely) guarded and we can set $\varphi' := \chi'$.

Also this formula is effectively computable, because all steps, especially the substitution step from χ to χ' is effective. ∎

We now define a slight modification of Ψ which will map (loosely) guarded formulae to (loosely) guarded graphs:

Definition 20
Let $\mathcal{S}$ be a support and $\Sigma_{\mathcal{S}}$ be the corresponding first order signature. The graph representation Ψ' which maps close (loosely) guarded formulae to (loosely) guarded conceptual graphs with free variables is defined as follows:

We assume the formula φ to be in a normal form, which is built using only $\exists, \wedge$ and $\neg$, also we assume each variable to be bound exactly once. This normal form can easily be generated by applying well known transformations. We define $\Psi'(\varphi)$ inductively over the structure of φ. We will use the same functions to manipulate conceptual graphs as we have done in Def. 8.

Apart from the quantifier case, Ψ' is defined exactly as Ψ.

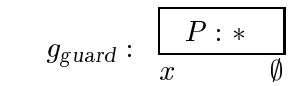
For a loosely guarded quantifier we define

$$\Psi'(\exists \mathbf{x}.((\alpha_1 \wedge \cdots \wedge \alpha_n) \wedge \beta(\mathbf{x},\mathbf{y}))) := \left[g_{guard}\ \Psi'(\beta')\right]$$

where $\beta' = \beta(\mathbf{x},\mathbf{z})$ and $\mathbf{z} = z_{y_1},\ldots,z_{y_m}$ contains a new variable for each component of $\mathbf{y} = y_1,\ldots,y_m$. $g_{guard} = \langle C,R,E,\ell\rangle$ is defined as follows:

If the guard consist of only an unary atomic expression of the from Px, then g_{guard} is defined as

g_{guard} :



If the guard consist of more then one atom or of an atom which is of higher arity, we define g_{guard} as follows:

For each variable x which appears in $\mathbf{x}$, C contains a concept node c_x of the form

$$c_x : \begin{array}{|c|}\hline \top_C : * \\ \hline x \qquad\quad \emptyset \\ \hline \end{array} .$$

For each y which appears in $\mathbf{y}$, C contains a concept node c_y of the form

$$c_y : \begin{array}{|c|}\hline \top_C : * \\ \hline z_y \qquad\quad \{y\} \\ \hline \end{array} .$$

For each constant i which appears in $\alpha_1, \ldots, \alpha_n$, C contains a concept node c_i of the form

$$c_i : \begin{array}{|c|}\hline \top_C : i \\ \hline i \qquad\quad \emptyset \\ \hline \end{array}$$

For each guarding expression $\alpha = Ss_1, \ldots, s_l$, R contains a node r_j. For each $j \in \{1, \ldots, l\}$ E contains the edges (c_{s_j}, r) and the label of this edge $\ell(c_{s_j}, r)$ contains j.

More formally we define:

- $C := \{c_x \mid x \in \mathbf{x}\} \cup \{c_y \mid y \in \mathbf{y}\} \cup \{c_i \mid i$ *appears in the guard*$\}$, $\ell(c_s)$ *is defined as in the pictures above.*
- $R := \{r_1, \ldots, r_n\}$, $\ell(r_j) := S$, *where S is the relation symbol of α_j.*
- $E := \bigcup_{j=1,\ldots,n} \{(c_s, r_j) \mid s$ *appears in* $\alpha_j\}$
- $\ell(c_s, r_j) := \{l \mid s$ *appears at the l-th position of* $\alpha_j\}$

The values of the functions id and links are defined as in the pictures above.

For a guarded quantifier we define $\Psi'(\exists \mathbf{x}.\alpha \wedge \psi(\mathbf{x}, \mathbf{y}))$ exactly as above as a special case of a loosely guarded quantifier.

The reader may validate the following fact:

Lemma 21
If $\varphi \in$ (L)GF($\Sigma_{\mathcal{S}}$) is in the normal form assumed in Definition 20, then $\Psi'(\varphi)$ is an conceptual graph with variables according to Def. 5.

At first we will have to show that the result of the translation of a (loosely) guarded formula is a (loosely) guarded conceptual graph with variables:

Lemma 22
If $\varphi \in (L)GF(\Sigma_{\mathcal{S}})$ is in the normal form required by Def. 20, then $\Psi'(\varphi)$ is a (loosely) guarded conceptual graph with free variables. Moreover, if φ is closed then $\Psi'(\varphi)$ is a (loosely) guarded conceptual graph (without free) variables.

Proof: We show this proof by induction over the structure of φ. All the base cases are trivially guarded, because the result of the translation is a conceptual graph with only a single context which contains no new nodes.

For the induction step we only have to consider the translation of a quantifier, because the constructions in the translation of $\wedge$ and $\neg$ trivially preserve the guardedness. The first two properties are the same for the guarded and the loosely guarded case, while the third one has to be proved separately. We will only show the most interesting case in this proof. Assume φ to be a formula with a loosely guarded quantifier, where the guard consists of more then one atom. All other cases are similar but simpler.

Let $\varphi = \exists \mathbf{x}.\alpha \wedge \beta(\mathbf{x}, \mathbf{y})$ where $\beta(\mathbf{x}, \mathbf{y})$ is guarded and $\mathit{free}(\beta) \subseteq \mathit{free}(\alpha) = \{\mathbf{x}, \mathbf{y}\}$. By the induction hypothesis, $G' = \Psi'(\beta')$ is guarded. Assume $G = \Psi'(\varphi)$ is not guarded. This implies that one of the Properties 1, 2 or 3 of Def. 16 is not satisfied by G.

1. Assume Property 1 is violated by by a concept node c in G. c can not be in the outermost context of G, because no node on this level can violated Property 1. It can also not be a node in an context in G' lying below a context which contains a new node, because then Property 1 would already be violated by G' which is a contradiction to the induction hypothesis. Hence c lies in a context q of G' such that between q and the outermost context of G there is no context which contains a new node. This implies (by the definition of Ψ') that c corresponds to the translation of a free variable z of β'. Since each free variable of β must occur in the guard expression, there are two possibilities for a free variable of β':

   - $z = x_j$ is a component of $\mathbf{x}$. Then g_{guard} contains a node c_{x_j} with $\mathit{id}(c_{x_j}) = x_j$, this implies that Property 1 is not violated.
   - $z = z_{y_j}$ is a component of $\mathbf{z}$. Then y_j is a free variable of both β and φ. This implies that y_j must occur in the guarding expression and hence g_{guard} contains the node c_{y_j} with $\mathit{id}(c_{y_j}) = z_{y_j}$ which again implies that Property 1 is not violated.

   In either case we get a contradiction to the initial assumption that G violates Property 1.

2. Property 2 can not be violated by G. This follows from the fact that G and all its subgraphs satisfy Property 1: Assume G would violated Property 2. The induction hypothesis implies that $G' = \Psi'(\beta')$ is (loosely) guarded and hence it satisfies both Property 1 and Property 2. If G violates Property 2, there must be nodes c_1, c_2 in context p_1, p_2 such that $y = \mathit{id}(c_1) \in \mathit{varlinks}(c_2)$, p_1 strictly dominates p_2 and there is a context q

which lies between p_1 and p_2 such that q contains a new node. p_1 must be the outermost context of G because otherwise G' would already violate Property 2. Since q lies between p_1 and p_2 it is also a context of G'. This implies that Property 1 is violated by c_2 in G' as follows: $y \in \mathit{varlinks}(c_2)$ and there is no node c' in G' such that $\mathit{id}(c') = y$, because in G there might be at most one concept node with identifier y and this is c_1. Since q strictly dominates p_2 and q contains a new node, G' violates Property 1, which is a contradiction.

3. Since $g_{\text{guard}} = \langle C, R, E, \ell \rangle$ contains new nodes, we must show that C has Property 2 (2') of Def. 16. The only interesting case is Property 2' for loosely guarded graphs:

   Assume there is a new node c with $\mathit{id}(c) = x$ and another node d such that c is new, d is not an individual concept node and there is no $r \in R$ such that $\{(c, r), (d, r)\} \subseteq E$. If d is new, then it is the result of the translation of a variable z which appears in the quantifier prefix. Hence there is an atom α_j such that x and z coexists in α_j. This implies that both c and d are adjacent to the node r_j, which is a contradiction. If d is not new, then it must be external. It may not be an individual concept node, because then Property 2' would not be violated. This implies, that d is the result of the translation of a free variable y which appears in $\mathbf{y}$. Each such y must coexists with x in at least one guarding expression and hence there is a node $r \in R$ such that both c and d are adjacent to r, which again is a contradiction to the assumption that no such r would exist.

We have thus shown that none of the conditions 1,2 or 3 is violated by G and hence G is (loosely) guarded.

If φ contains no free variables, then we have to show that for each variable z which appears in $\mathit{varlinks}(c)$ for a concept node $c \in C(p)$, there is a node c' in a dominating context such that $\mathit{id}(c') = z$. Such a z may either stem from the translation of a variable in φ for which there is a quantifier which binds z and hence there is also a node c' which corresponds to the translation the quantifier and for this node $\mathit{id}(c') = z$ holds. z may also stem from the translation of a guard expression which guards a free variable y . Then the simple graph which corresponds to the guard expression contains a then node c_y for which $\mathit{id}(c_y) = z$ holds. ∎

The next lemma is the guarded version of Theorem 9 and justifies the name *graph representation* for Ψ':

Lemma 23
Let $\mathcal{S}$ be a support and $\Sigma_{\mathcal{S}}$ the corresponding first order signature. For each formula $\varphi \in (L)GF(\Sigma_{\mathcal{S}})$ (possibly with free variables)

$$\Phi(\Psi'(\varphi)) \equiv \varphi.$$

Proof: We show this lemma by induction over the structure of φ. Since Ψ' is nearly defined as Ψ, we can use most of the arguments of Theorem 9. Since Ψ' agrees with Ψ in all the base cases, the base of the induction is shown.

For the induction step we only have to show the quantifier case, because Ψ' agrees with Ψ in all other cases. Hence we assume that φ is of the form

$$\varphi = \exists \mathbf{x}.((\alpha_1(\mathbf{x}, \mathbf{y}) \wedge \cdots \wedge \alpha_n(\mathbf{x}, \mathbf{y})) \wedge \beta(\mathbf{x}, \mathbf{y}))$$

where not all variables from $\mathbf{x}, \mathbf{y}$ need to occur in each of the guards. For such a formula, $\Psi'(\varphi)$ is defined by

$$\Psi'(\varphi) = \left[g_{\text{guard}}\ \Psi'(\beta')\right]$$

Where $\beta' = \beta(\mathbf{x}, \mathbf{z})$ and $\mathbf{z} = z_{y_1}, \ldots, z_{y_n}$ contains a new variable for every component of $\mathbf{y} = y_1, \ldots, y_n$. By the Definition of Φ we get

$$\Phi(\Psi'(\varphi)) = \phi_p(g_{\text{guard}}).\psi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

with

$$\psi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \phi(g_{\text{guard}}) \wedge \Phi(\Psi'(\beta'))$$

From the induction hypothesis we immediately get, that $\Phi(\Psi'(\beta')) \equiv \beta'$.

For each guard expression $\alpha(\mathbf{x}, \mathbf{y}) = Ss_1, \ldots, s_m$, g_{guard} contain a relation node r such that $\ell(r) = S$, $(c_{s_j}, r) \in E$ and $j \in \ell(c_{s_j}, r)$. From the definition of ϕ it follows, that

$$\phi(r) = \alpha(\mathbf{x}, \mathbf{z})$$

For each variable x which appears in $\mathbf{x}$, g_{guard} contains a node c_x and hence $\phi_p(g_{\text{guard}})$ contains the quantifier $\exists x$ and $\phi(g_{\text{guard}})$ contains the conjunct $x \doteq x$ which is trivially satisfied and can be omitted. For each variable y in $\mathbf{y}$, g_{guard} contains a node c_y and hence $\phi_p(g_{\text{guard}})$ contains the quantifier $\exists z_y$ and $\phi(g_{\text{guard}})$ contains the conjunct $z_y \doteq y$. Thus for each y in $\mathbf{y}$ it holds that

$$\psi \models z_y \doteq y$$

as before we substitute each z_y by in ψ by y and thus get formula φ':

$$\varphi' := \phi_p(g_{\text{guard}}).\psi(\mathbf{x}, \mathbf{y}, \mathbf{y}) \equiv \phi_p(g_{\text{guard}}).\psi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

Since no z from $\mathbf{z}$ occurs in $\psi(\mathbf{x}, \mathbf{y})$, we can remove all quantifiers for these variables from the quantifier prefix and finally get the formula

$$\varphi'' := \exists \mathbf{x}.\psi(\mathbf{x}, \mathbf{y}, \mathbf{y}) \equiv \varphi'$$

A further inspection of φ'' shows that

$$\varphi'' \equiv \exists \mathbf{x}.((\alpha_1(\mathbf{x}, \mathbf{y}) \wedge \cdots \wedge \alpha_n(\mathbf{x}, \mathbf{y})) \wedge \beta(\mathbf{x}, \mathbf{y})) = \varphi$$

■

As a corollary we get the following theorem, which states, that Ψ' is a semantically correct translation from *(L)GF*$(\Sigma_\mathcal{S})$ to *(l)gCG*$(\mathcal{S}, \mathcal{V})$:

Theorem 24
For each formula $\varphi \in$ *(L)GF*$(\Sigma_\mathcal{S})$ *it holds that*

$$\Phi(\Psi(\varphi)) \equiv \Phi(\Psi'(\varphi))$$

Proof: From Theorem 19 we get that

$$\Phi(\Psi'(\varphi) \equiv \varphi$$

and from Theorem 9 we get that

$$\varphi \equiv \Phi(\Psi(\varphi))$$

■

We have established the correspondence between *(L)GF*$(\Sigma_\mathcal{S})$ and *(l)gCG*$(\mathcal{S}, \mathcal{V})$. Together with Lemma 22 this induces the same correspondence between closed formulae from *(L)GF*$(\Sigma_\mathcal{S})$ and graphs in *(l)gCG*$(\mathcal{S})$.

6 Applications

Note that, even though the definition of lgCGs may look quite complex at first sight, it is a purely syntactic definition using easily testable properties of graphs. Indeed, it is easy to show that membership of a given CG over $\mathcal{S}$ in *(l)gCG*$(\mathcal{S})$ can be tested in polynomial time:

Lemma 25
There is an algorithm `is-guarded` *which runs in cubic time in the size of the input, such that, given a graph* $G \in CG(\mathcal{S})$

$$\texttt{is-guarded}(C) = \begin{cases} \texttt{guarded}, & \text{if } G \text{ is guarded} \\ \texttt{loosely guarded}, & \text{if } G \text{ is loosely guarded} \\ \texttt{unguarded}, & \text{otherwise} \end{cases}$$

Proof: If we assume an appropriate coding of the input graph G, it is possible to decide in constant time, whether a node c is an individual concept node. To test whether a node is external takes linear time, because all other nodes in its simple graph have to be visited to examine their label *id*. Hence it is possible to mark all nodes in all simple graphs as *old* or *new* in quadratic time, by calculating the connected components of the simple graphs and by marking the nodes in a component as *old*, if the component does contain any individual

or external nodes. Otherwise the nodes in the component are marked as *new*. Also we mark all simple graphs which contain new nodes.

The algorithm will now assign the labels *guarding* and *loosely guarding* to the simple graphs in G. If it finds a violation of one of the properties from Def. 14, it will immediately answer `unguarded` and will exit. Otherwise it will answer based on the labels assigned to the simple graphs.

Based on markings of the nodes, it is tested, whether G satisfies the Properties 1 and 2 of Def. 14.

1. For each coreference links (c_1, c_2) linking concepts in the contexts p_1, p_2 test whether there is a context between p_1 and p_2 which contains a new node. If such a context exists, then it answers `unguarded` and exit. Each such test can be carried out in linear time.

2. For each simple graph $g = \langle C, R, E, \ell \rangle$ which contains a new node, test the following:

   - Test, if Property 2 is satisfied. If $C = \{c\}$, test whether $c \neq \top_C$. If this is the case, then mark g as *guarding*. This is possible in constant time. If this is not the case, test whether there is an $r \in R$ such that $C = \{c \in C \mid (c, r) \in E\}$. This can be done by iterating through R and takes at most quadratic time. If there is such an r then mark g as guarding.
   - If g is not already marked as *guarding*, then test, whether Property 2' is satisfied. For each new node c and each other node d which is not an individual concept node, test, whether there is an $r \in R$ such that both c and d are adjacent to r. This can be done in quadratic time. If Property 2' is satisfied, then mark g as *loosely guarding*, otherwise answer `unguarded` and exit.

   Once all simple graphs have been marked as *(loosely) guarding* and the algorithm has not exited with `unguarded`, then answer `guarded`, if all graphs are marked as *guarding*, otherwise answer `loosely guarded`.

All tests and markings this algorithm performs can be carried out in at most quadratic time and hence this lemma is proved. ∎

Validity of conceptual graphs has always been an important topic. Sowa [Sow84] proposes a set of rules which can be used to derive valid graphs. Later Wermelinger [Wer95a] made slight modifications to these rules and proves them to be sound and complete for his definition of CGs with higher order features in [Wer95b]. A different approach was used in [KS97], where a tableaux system is used to proof the validity of a CG. Since CGs are able to express full first order logic (as, for example, has been shown in, Theorem 9) it is not possible to give a decision algorithm for validity, because this would imply a decision algorithm for validity of first order logic. For the (loosely) guarded fragment of

conceptual graphs validity is decidable, which makes is feasible to use them in fully automatic reasoners.

Since we want to use the first order semantics of guarded graphs to decide their validity, we will need to code the support by a guarded FO formula as well. Thus we define:

Definition 26
Let $\mathcal{S} = \langle \mathcal{T}_C, \mathcal{T}_R, N_I \rangle$ be a support. We define $\Phi(\mathcal{S})$ by:

$$\Phi(\mathcal{S}) := \bigwedge_{P \leq_C Q} \forall x.Qx \to Px \wedge \bigwedge_{S_1 \leq_R S_2} \forall \mathbf{x}.S_2\mathbf{x} \to S_1\mathbf{x}$$

This is the standard translation, which, for example, is also used in [CMS98]. Note that $\Phi(\mathcal{S})$ is a guarded formula.

We define validity of graphs with respect to their FO semantics.

Definition 27
A graph $G \in CG(\mathcal{S})$ is called valid, *iff*

$$\Phi(\mathcal{S}) \to \Phi(G) \equiv \mathit{true}.$$

Theorem 28
Let $\mathcal{S}$ be a finite support (i.e., all components of $\mathcal{S}$ are finite). The set Valid *with*

$$\mathsf{Valid} := \{G \in lgCG(\mathcal{S}) \mid G \text{ is valid}\}$$

is decidable. Furthermore Valid $\in$ ExpTime *holds.*

Proof: Let $G \in lgCG(\mathcal{S})$. To decide, whether G is valid, proceed as follows: Compute $\widehat{G}$, the canonical translation of G into a CG with variables. Apply Φ to $\widehat{G}$ and use Theorem 19 to get a $\varphi' \in LGF(\sigma_{\mathcal{S}})$ such that $\varphi' \equiv \Phi(G)$. By Fact 11, it is decidable, if $\neg(\Phi(\mathcal{S}) \wedge \varphi')$ is satisfiable. It holds that

$$\Phi(\mathcal{S}) \wedge \neg\varphi' \text{ is not satisfiable iff } G \text{ is valid}$$

and hence we have shown the set Valid to be decidable.

Since all steps in this algorithm except the last on can be computed in polynomial time and Fact 12 states, that satisfiability for the loosely guarded fragment is decidable in exponential time if only a finite signature is used, we also get that that Valid $\in$ ExpTime holds. ∎

Another important problem for conceptual graphs is the problem of subsumption. Subsumption between two simple graphs is usually defined for in terms of the existence of a certain homomorphism between the to graphs. In [CMS98] it is shown, that there is an equivalent definition of subsumption using the FO semantics of simple graphs. This definition can be generalized to full CGs:

Definition 29
Let $\mathcal{S}$ be a support and $G, H \in CG(\mathcal{S})$ be two conceptual graphs with coreference links. H subsumes G *($G \leq H$), iff*

$$\Phi(\mathcal{S}), \Phi(G) \rightarrow \Phi(H) \equiv true$$

While subsumption for simple graphs is decidable in NP, it is undecidable for all CGs, because such a decision procedure would imply the existence of a decision procedure for entailment in FO. Yet, for loosely guarded graphs, subsumption is decidable.

Theorem 30
Let $\mathcal{S}$ be a finite support and let Subsumption *be defined by*

$$\mathsf{Subsumption} := \{(G, H) \mid G, H \in lgCG(\mathcal{S}), G \leq H\}.$$

Subsumption *is decidable and* Subsumption $\in$ ExpTime *holds.*

Proof: We use the same translation as in Theorem 28 to get two guarded formulae $\varphi_G \equiv \Phi(G)$ and $\varphi_H \equiv \Phi(H)$.

$$G \leq H \text{ iff } (\Phi(\mathcal{S}) \wedge \varphi_G \wedge \neg\varphi_H) \text{ is not satisfiable}$$

The latter can be decided because of Fact 11 and we can use the same complexity arguments as in the proof of Theorem 28 to get that Subsumption $\in$ ExpTime holds. ∎

References

[CMS98] M. Chein, M. L. Mugnier, and G. Simonet. Nested Graphs: a Graph-based Knowledge Representation Model with FOL Semantics. In A.G. Cohn, L.Schubert, and S.C. Shapiro, editors, *Proceedings of the 6th International Conference on Knowledge Representation (KR'98)*, pages 524–534, Trento, Italy, June 1998. Morgan Kaufman Publishers Inc.

[Grä98] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 1998. to appear.

[KS97] G. Kerdiles and E. Salvat. A sound and complete CG proof procedure combining projections with analytic tableaux. In D. Lukose, H. Delugach, M. Keeler, L. Searle, and J. Sowa, editors, *Proceedings of the 5th Internation Conceference on Conceptual Structures, ICCS'97*, volume 1257 of *LNCS*, pages 371–385. Springer, 1997.

[Sow84] John F. Sowa. *Conceptual Structures*. Addison-Wesley, Reading, MA, 1984.

[Wer95a] M. Wermelinger. Conceptual graphs and first-order logic. *Lecture Notes in Computer Science*, 954:323–??, 1995.

[Wer95b] M. Wermelinger. Teoria básica das estruturas conceptuais. Master's thesis, Universidade Nova de Lisboa, 1995.