Towards Benchmarks for Conceptual Graph Tools

Jesse-Francois Bayet, Olivier Carbone, Michel Chein, David Gentz, Alain Guichard, Michel Leclercq, Marie-Laure Nougier, Eric Salvat, and Roland Theremps

University of Nice-Sophia Antipolis

Abstract. This paper reports a collective reflection led in our team about conceptual graph benchmarks. We tackle four issues for which agreement should be obtained before benchmarking can be built; what are the fragments of CGs considered? How is information exchanged? What are the problems to be solved? What kind of tools property are evaluated by the benchmark? We define a basic building block defined upon simple conceptual graph. Finally we propose to provide a first benchmark adapted from an industrial case study. This benchmark is composed on very simple structures and should allow to focus on interoperability issues.

1 Introduction

This paper reports a collective reflection led in our team about conceptual graph benchmarks. We expect CGers to comment, support and complete this contribution.

Tackling the issue of CG tools interoperability leads to several questions:

1. What are the fragments of CGs handled by the tools? Answering that question involves an agreement on a classification of specific forms of conceptual graphs.
2. How can tools exchange pieces of information? Answering that question involves agreeing on an exchange format.
3. What are the basic problems to be solved by the tools? Answering that question involves defining common basic problems in a way precise enough to be able to define for a given instance of the problem the solution that should be found.
4. What kind of benchmark data should we provide? Answering that question involves defining what tool properties are to be evaluated by a benchmark.

In what follows we detail each above question, with identifying the difficulties that should be overcome. We make a first proposal, built upon the so-called simple conceptual graphs.

The tools we consider are those referenced on the wiki page maintained by Philippe Martin.

2 Fragments of conceptual graphs

Why is it essential to identify conceptual graph fragments? Conceptual graphs can be seen as a family of languages rather than as a language. A reason is that, since John Sowa's seminal book, definitions, even of basic notions, have evolved along the years and each team has added its own development.

Another reason, which is essential in the context of tool interoperability, is that no tool implements conceptual graphs in their generality. General conceptual graphs are equivalent to first-order-logic (FOL) but no tool implements them. No tool does even implement any form of negation.

Our idea before writing this paper was to consider on one hand the CG standard (ISO/TC1/SC32/WG29) that can be found on Sowa's website [1] and on the other hand the kind of constraints handled in existing tools for proposing a classification of CG fragments, that could represent the most common constraints. A constraint handled by a tool could then be defined by differences with elements of this classification. However the variations among tools are so big that we lowered our ambitions. As a first attempt, we restrict the proposal to a fragment built on simple conceptual graphs. This data model is indeed the simplest model shared by the CG community. In the perspectives, other candidate fragments are discussed.

Each CG fragment should be precisely defined and should possess a semantic in FOL. That is a minimal requirement. In our opinion, another requirement should be the definition of operators for dealing with this fragment. A subset of those operators should allow to compute deduction in a sound and complete way (see section 4).

Let us begin with the common unquestionable basic simple conceptual graphs (SCGs). All tools implement them [1]. SCGs are furthermore fundamental for comparison with other languages/formalisms as they correspond to the construction of a conceptual fragment of FOL. They are for instance equivalent to the conjunctive queries of databases. About the relationships with the semantic web, it

http://www.w3.org/2001/06/sw/cg


[2] The CG editor ChartCG allows to represent negative contexts but does not provide reasoning on them.


Note however that Amine restricts CG to binary relation types and the relations adjacent to a concept must be of different types.
is interesting to notice that SCGs subsume RDF (transformations from RDF to SCGs preserving the semantics have been proposed in [CMR06] [Bag06]).

According to the standard, SCGs may include informative links. Let us point out however that these links vary on the constraint that can be enforced on them. In Cognoit for instance concept names, concept nodes must have the same label (but this restriction should be relaxed in a next version). More generally, there are variations on the way of dealing with distinct nodes referring to the same entity (either by means of reference links or by identical individual markers). We thus propose to distinguish an even simpler subclass of CGs: SCGs in normal form, that is to say two nodes referring to the same entity.

SCGs come with two kinds of equivalent operations: elementary inference rules (see [Bag06] for a discussion on several sets of rules) and projection/instance projection. Checking projection/inverse projection can be done more efficiently than building a derivation with elementary rules, essentially because it allows efficient preprocessing (though not, restricting the possible images for nodes of the source SCG). As a matter of fact, projection is provided in all tools. It is complete with deducibility only if the target graph is in normal form. Core-projection is a variant that is complete without this restriction [CMM05]. It is not implemented in any tool at the moment.

In what follows, classical mathematical definitions of SCGs are given and are reduced, whenever it is possible, to the abstract syntax of the CG standard.

1. SCG-module. A SCG-module is a tuple \( M = (V, I, B) \), where \( V \) is a SCG-vocabulary, \( I \) is a set of individual markers and \( B \) is a set of SCGs built over \( V \) and \( I \). (This is the same as the definition of a module in [FM03, p.10] of the standard where \( V \) is called type hierarchy, \( I \) catalog of individuals and \( B \) assertions. Furthermore, as in the SCG model, there is no context, a module is simply a triple of sets and not a context composed of three contexts.)

2. SCG-vocabulary. A SCG-vocabulary is composed of two (disjoint) sets \( V \) and \( I \), and \( B \) is a partially ordered set of concept types with a maximal element (base) as in Sect. 5.6, without neither defined types nor the abstract (ge) and \( T \) is a partially ordered set of relation types. Each relation has an arity \( > 0 \) (called valence) and a signature (same as in Sect. 6.3, without 0-ary relation, defined types, and actors. The signature is defined in Sect. 6.3.

3. Individual markers, \( I \) is a set of individual markers (as in Sect. 5.7).

4. Simple Conceptual Graph. A SCG is a 5-tuple \( (C, R, E, I, mcpu) \) where \( C \) is a set of concept nodes (as in Sect. 6.2 where a referent, of 6.3, is restricted to a designator that is either an individual marker or underlined - also called generic). If \( r \) is a concept node its label \( l(r) \) is the pair \(( \text{type}(r), \text{referent}(r)) \). It is a set of relation nodes (as in Sect. 6.1, without the above restrictions for relation types). If \( r \) is a relation node its label \( l(r) \) is the type of \( r, (U, R, E) \) is a bipartite multigraph (as in Sect. 6.1, but with multiple edges explicitly allowed between two nodes). The labelling function \( l \) of the nodes has been given above; the labelling function of the edges is such that for any relation node \( r \), has exactly \( k = \text{arity}(l(r)) \) incident edges which are labelled \( 1, 2, \ldots , k \). The correspondence \( c_{u} \) is a partition of \( C \) with each class being called a coreference set. (We have gathered in this definition of a SCG C, R, E, I, mcpu where the referent is restricted to the above designators, R, with the above restrictions. The "static" graph is also called the empty CG, and \( R, E \) where any concept must be in a coreference set and the dominant concept is not needed)

5. Normal Simple Conceptual Graph. A normal SCG is a SCG where every \( c_{u} \) is the identity. There are no distinct nodes representing the same entity.

6. SCG-Query. A SCG-query is a SCG. We only define the most basic form of query; a straightforward extension to add question markers on some generic concept nodes, is to consider a symbolic-SCG (see in the perspective); we then obtain the existential-quantifier query of databases.

As an example, let us consider a vocabulary where \( T_{s} \) contains the concept types Top, Object, Color, Ball, Attribute, Color, Shape, Rectangles, Square, Rhomb, partially ordered as follows: Object \( \subseteq \) Top, Color \( \subseteq \) Object, Ball \( \subseteq \) Object, Attribute \( \subseteq \) Top, Color \( \subseteq \) Attribute, Shape \( \subseteq \) Attribute, Rectangle \( \subseteq \) Shape, Rhomb \( \subseteq \) Shape, Square \( \subseteq \) Rectangle and Square \( \subseteq \) Rhomb. \( T_{s} \) contains the binary relations pop, peer, and on Top and the ternary relation between. We have an \( \text{on} \) \( \subseteq \text{more} \). Finally, this vocabulary contains the individual markers \( A \) and \( Nice \). We graphically represent the partial order \( T_{p} \) of this vocabulary in Fig. 1, as well as a simple conceptual graph built upon this vocabulary.

![A simple conceptual graph G](image)

Fig. 1. A partial order on \( T_{p} \) and a simple conceptual graph \( G \).

The logical semantics \( \mathcal{V} \) assigned to these constructs is well-known. A SCG is assigned to a conjunctive and existentially closed formula. If \( B \) is a set of SCGs, \( \mathcal{V}(B) \) is the set of formulas assigned to the elements of \( B \). Conjoining is translated by conjunction. The set of formulas assigned to the vocabulary translates.
the partial orders on types. Given a SCG module $M = \langle V, L, B \rangle$, $\Phi(M)$ is the
set of formulas resulting from the union of $\Phi(V)$ and $\Phi(B)$.

3 Exchange format

Agreement on an exchange format is the first requirement for benchmarking. To decide on a particular format, we considered the three following criteria:
1. It must be able to express items in the SCG format (vocabulary, graph, ...
2. It must be extensible enough to cope with more expressive CG fragments (e.g., rules) in forthcoming benchmarks;
3. To open the benchmark to a wide area of CG tools, this format must be already implemented in most tools.

A natural choice with respect to these criteria is CGIF. Since it is able to encode (at least) the whole FOIL, it naturally satisfies the two first criteria. According to Philippe Martin's wiki page, it is implemented in all CG tools except our tool Cycle, which is only able to write SCGs in CGIF (but not yet to read them).

Two versions of CGIF are available:
- the first is given in the "Conceptual graphs ISO/ITCI/SC32/WG2 N 050" proposed standard. We will refer to this version as CGIF 2001;
- the second version is given in the "Common Logic ISO/IEC/ITCI 1/SC32 1.177 proposed standard. The basic version presented is called Core CGIF; a more expressive version is called Extended CGIF.

In the following, we recall the grammar of these three formats by considering their respective sublanguages that correspond to SCGs, and characterize the SCGs represented in these formats. The semantics will be expressed via the translation to FOIL, and logical consequences between associated formulas is used to define the deduction problem of SCGs.

3.1 CGIF 2001

// Representation of a module

Module :=
[Module: TypeHierarchy RelationHierarchy CatalogIndividuals Assertion]

// Representation of the ordered set of concept types

TypeHierarchy :=
[TypeHierarchy: (TypeLabelOrdering) *]

TypeLabelOrdering :=
[GT [TypeLabel: Identifier *] [TypeLabel: Identifier *]]
\[ f(c_1) \ldots f(c_n) \] with \( f(c_i) = \{ \text{red} \} \) if \( c_i \) is an individual concept node with marker \text{red} and \( f(c_i) = \{ \text{blue} \} \) otherwise.

The CGIF 2001 files provided for the proposed benchmarks will be obtained by applying this transformation to vocabulary and SCGs in normal form. By reading these files, the normal form of the encoded SCGs will be retrieved. The semantics of the objects encoded in these CGIF 2001 files will be the semantics \( \Phi \) of the vocabularies and SCGs from which they were obtained.

As an example, here are the CGIF 2001 files associated with the vocabulary and graph of Sec. 1.

Encoding of the SOC vocabulary

\[
\begin{align*}
\text{Type Hierarchy:} & \\
& (\text{GT} \ [\text{TypeLabel:} \text{"Top"}] \ [\text{TypeLabel:} \text{"Object"}]) \\
& (\text{GT} \ [\text{TypeLabel:} \text{"Top"}] \ [\text{TypeLabel:} \text{"Attribute"}]) \\
& (\text{GT} \ [\text{TypeLabel:} \text{"Shape"}] \ [\text{TypeLabel:} \text{"Rhomb"}]) \\
& (\text{GT} \ [\text{TypeLabel:} \text{"Shape"}] \ [\text{TypeLabel:} \text{"Rectangle"}]) \\
& (\text{GT} \ [\text{TypeLabel:} \text{"Object"}] \ [\text{TypeLabel:} \text{"Cube"}]) \\
& (\text{GT} \ [\text{TypeLabel:} \text{"Object"}] \ [\text{TypeLabel:} \text{"Ball"}]) \\
& (\text{GT} \ [\text{TypeLabel:} \text{"Attribute"}] \ [\text{TypeLabel:} \text{"Shape"}]) \\
& (\text{GT} \ [\text{TypeLabel:} \text{"Attribute"}] \ [\text{TypeLabel:} \text{"Color"}]) \\
\end{align*}
\]

Relation Hierarchy:

\[
\begin{align*}
& (\text{RelationLabel:} \text{"prop"}) \\
& (\text{RelationLabel:} \text{"between"}) \\
& (\text{GT} \ [\text{RelationLabel:} \text{"near"}] \ [\text{RelationLabel:} \text{"onTop"}]) \\
\end{align*}
\]

Catalog of Individuals:

\[
\begin{align*}
& \text{Cube:} \text{"A"} \\
& \text{Color:} \text{"blue"} \\
\end{align*}
\]

Encoding of the SOC G

\[
\begin{align*}
& \text{prop [Ball \ \text{\"c2\"} \ Color: \text{"blue"}])} \\
& \text{prop [Cube \ \text{\"c6\"} \ \text{\"blue"}])} \\
& \text{onTop \ \text{\"c5\"} [\text{Cube:} \text{"A"})} \\
& \text{between \ [\text{\"A\"}] \ [\text{Ball:} \text{\"c2\"}] \\
\end{align*}
\]

3.2 Core/Extended CGIF

In this version of CGIF, information encoded in SCGs will be written in Core CGIF, while information encoded in the support will be encoded in Extended CGIF.

// Representation of a CG

CG :=

(Concept | ConceptualRelation)

Concept :=

(ExistentialConcept | ConjunctionConcept)

ExistentialConcept :=

[Identifier]

ConjunctionConcept :=

[If : CG [Then : CG]]

// Representation of the partial orders

PartialOrder :=

(If : CG [Then : CG])

To encode a SOC and its vocabulary in Core/Extended CGIF, we perform the following operations:

1. Let \( \Phi \) be the covering relation for the partial order on relation types of arity \( \leq 2 \).
2. For each pair of types \( (A_i, B_j) \) such that \( A_i \leq A_j \), we define the partial order \( (A_i, B_j) \) if \( A_i \) is in the SOC, otherwise \( (A_i, B_j) \) is in the SOC, otherwise \( (A_i, B_j) \) is not.
3. For each concept node \( c \) with marker \( m \) and type \( t \), we define \( f(c) = m \) if \( c \) is an individual, otherwise \( f(c) \) is a distinct identifier associated with each concept node in the SOC.
4. For each relation node \( r \) with marker \( m \) and type \( t \) in the SOC, \( G \) contains \( \{ \text{adj} \} \) and \( \{ \text{rel} \} \). For each relation node \( r \) in the SOC, \( G \) contains \( \{ \text{adj} \} \) and \( \{ \text{rel} \} \).
5. For each relation node \( r \) with type \( t \), \( m \) \neq 2 \), and \( n \) such that \( n \) is the number of arguments, \( G \) contains \( \text{prop} \) and \( \text{rel} \).

As for CGIF 2001, the SCGs retrieved by reading these files use the normal form of the encoded SCGs, and their semantics are defined by the translation \( \Phi \) to \( \Phi' \).

As an example, here are the Core/Extended CGIF files associated with the vocabulary and graph of Sec. 1.

Encoding of the SOC vocabulary

\[
\begin{align*}
& \text{If:} \ [\text{\"A\"}] \ [\text{Object:} \text{\"A\"}] \ [\text{Then:} \text{\"Top\"}] \\
& \text{If:} \ [\text{\"A\"}] \ [\text{Attribute:} \text{\"A\"}] \ [\text{Then:} \text{\"Top\"}] \\
& \text{If:} \ [\text{\"A\"}] \ [\text{Rhomb:} \text{\"A\"}] \ [\text{Then:} \text{\"Shape\"}] \\
& \text{If:} \ [\text{\"A\"}] \ [\text{Rectangle:} \text{\"A\"}] \ [\text{Then:} \text{\"Shape\"}] \\
& \text{If:} \ [\text{\"A\"}] \ [\text{Square:} \text{\"A\"}] \ [\text{Then:} \text{\"Shape\"}] \\
& \text{If:} \ [\text{\"A\"}] \ [\text{Cube:} \text{\"A\"}] \ [\text{Then:} \text{\"Object\"}] \\
& \text{If:} \ [\text{\"A\"}] \ [\text{Ball:} \text{\"A\"}] \ [\text{Then:} \text{\"Object\"}] \\
& \text{If:} \ [\text{\"A\"}] \ [\text{Shape:} \text{\"A\"}] \ [\text{Then:} \text{\"Attribute\"}] \\
& \text{If:} \ [\text{\"A\"}] \ [\text{Color:} \text{\"A\"}] \ [\text{Then:} \text{\"Attribute\"}] \\
\end{align*}
\]
3.3 Discussion

As discussed above, a requisite for choosing a particular format (at least for the first installment of a benchmark) is that it should already be implemented in most CG platforms. Though CGIF is the best candidate for that purpose, we could not decide between the two versions (2001 and Core/Extended) for the following reasons:

1. On one hand, the adoption of the Core/Extended version of CGIF as an ISO standard should render obsolete the 2001 version;
2. On the other hand, it is doubtful that all CG platforms implementing CGIF have already migrated from the 2001 version to the Core/Extended version.

Should we want to decide between these two versions for further installments of a benchmark, we need to compare CGIF 2001 and Core/Extended CGIF:

- While CGIF 2001 exactly represents the objects (type hierarchies, SCCGs) we manipulate, Core/Extended CGIF represents (with a CG notation) the formulas associated by \( \Phi \) to those objects. Though the proximity of the Core/Extended version with logics could be useful to open CG tools to other KR formalisms, it also blurs the specificity of our graph-based approach.
- Core/Extended CGIF does not represent the signature (since the signature of a relation type is not a rule, but an integrity constraint in the database sense). Though the signature is not used to compute SCCGs deduction (it is not taken into account by the semantics \( \Phi \)), it is a useful guide when editing SCCGs.

- In both versions of CGIF, nested graphs are used to represent SCCGs and vocabularies. It can be a dangerous shiver since the logical semantics of these nested graphs is not equivalent to the semantics of the SCCGs and vocabularies they represent.

Ultimately, the choice of a format for our community’s benchmarks should come from a collective decision. However, this choice is not limited to the versions of CGIF used above, but we should consider XML-based formats (such that...
The basic problem we aim to solve is testing computational efficiency of tools. Each of these levels should be tested using different data, having different characteristics.

The first, auxiliary point concerns in coming to an agreement on basic categories of conceptual graphs and several formats, with precise specifications for each CG fragment (as discussed in sections 2 and 3).

Then we propose to collect several benchmark data, with each of them:
1. the CG fragment considered;
2. a precise definition of the problem to solve;
3. possibly the kind of difficulties represented by the benchmark (e.g. the size of the knowledge base, the density of the graphs, etc.).

As a first experiment, we propose to provide a benchmark issued from an industrial knowledge management tool. This benchmark is limited to the SCC fragment, and furthermore to a particular kind of SCGs; in particular, relation types are all binary and all concept nodes have individual markers. However, this simplicity can be an advantage for a first step, as it should not yield computational difficulties and thus allows to focus on interoperability aspects: the exchange formats, and the problems to be solved. In other words, we focus here on the two first levels of a benchmark.

The industrial tool at the source of the benchmark allows to manage different kinds of knowledge structures: ontologies, databases, annotation bases, etc. All this knowledge is stored in a repository based on Topic Maps.

Our team is developing a reasoning service for this tool. In this goal, we have defined a transformation from the network of topics contained in the repository to the conceptual graph format which maps ontologies into the type hierarchy of conceptual graph model and the theorems as well as the annotation bases into SCGs.

The first knowledge base on which we have applied this transformation is the demonstration base of Mendeley which describes explicitly concepts and relations in the sector of pharmacology. These descriptions are issued from an automated process of compilation based on NLP tools.

The obtained benchmark is composed of:
- a concept type hierarchy (tree-structured) of 88 types;
- a relation type hierarchy (flat) of 92 relations. All these relations are binary and are provided with a signature (2 concept types from preposing hierarchy);
- a set of 6506 individual markers;
- a SCC composed of 222 connected components and a big amount of isolated concept vertices. Each concept node is an individual one (i.e. there is no generic marker) and the entire graph is in normal form. It contains 6505 concept nodes corresponding to different markers and 5406 isolated vertices. Most of connected components are types of small size (approximately 30 nodes). Two components have more than 4000 nodes and contain cycles.

**Definition 2 (SCC Question Answering).** Given a SCC module $M = (V, I, R)$ and a SCC $Q$ defined on $V$ and $I$, give all answers to $Q$ in $M$.

The associated counting problem (how many answers are there?) might be of interest too. The above problem assumes a specific definition of an answer. Roughly, an answer describes a way of instantiating the nodes of the query onto $M$. More specifically, it is a mapping from the nodes of $Q$ to the nodes of $M$ such that if we replace each node of the query with its image we obtain a piece of knowledge deducible from the module. If $H$ is in normal form (that is the set of SCGs considered as a single SCC is in normal form), we obtain a subgraph of $B$ (the projection of $Q$ in $B$).

If $B$ is in normal form, an answer is a projection from $Q$ to $H$. More generally, an answer is a core-projection from $Q$ to $B$, that is a mapping from each core class of $Q$ to a core class of $B$ that satisfies similar properties on labels and edges as projection. Each core-projection from $Q$ to $B$ corresponds to a projection from the normal form of $Q$ to the normal form of $B$ if these normal forms exist (which depends on the constraints enforced on the coreference relation), and reciprocally.

**5 Benchmarks**

One can distinguish between several levels of benchmarks, depending on the tool properties that are to be evaluated. At first level, a benchmark can be used to check if tools accept a given format as input and as output. A second level is to test the ability of tools to solve a problem. As explained in previous section, we propose deduction and question answering as basic problems. When tools pass the two first levels, it is possible to consider a third level, which aims at...
We also propose a list of query graphs with associated results. For each graph, we give the boolean result for the deduction problem, the number of different projections from it into the base and the list of image graphs induced by those different projections.

Let us end this section with ideas related to the third benchmark level, that is testing tools efficiency.

Through a realistic, large scale knowledge base is a previous benchmark to evaluate the performance of our tools on practical instances, a more precise evaluation could use the notion of phase transition [MST92,SCS95]. Though this notion was introduced for constraint networks, it can be directly translated into the SCG framework: Let D be a random SCG generator. A parameter of this generator is its hardness. When generating an instance of the SCG-deduction problem with hardness close to 0, every graph in the query Q has less than one image in the knowledge base. With hardness close to 1, each star graph in the query has very few images in the KB. Provably, there are lots of projections of the query in the first case, and none in the second. By varying this parameter, we can find a value for which the expected number of projections is 1. This is where the phase transition stands. For this hardness value, all sound and complete SCG deduction algorithms will suffer from a transition peak of efficiency. This peak exists for all NP-complete problems, and is preserved by polynomial reductions between these problems.

To evaluate CC tools deduction efficiency for various hardness values, we could use a concrete network random generator, and translate the networks obtained into SCGs with the polynomial reduction RGC presented in [MGG90]. Such a range of SCG-deduction instances allows a more precise comparison of efficiency: some tools can be better for some hardness values, and worse for others.

6 Perspectives

In this paper, we have defined a basic building block for which we propose to provide benchmarks if there is an agreement on it among CG tool developers. This building block is built upon normal SCGs, which have a well defined logical semantics and are equipped with a community implemented operation, namely projection. We have proposed a restriction of the CGP schema corresponding exactly to this fragment. The basic problems are deduction and query answering.

For the first installment of these CG benchmarks, we will be able to evaluate if CG tool developers really agree on the syntax and semantics of simple CGs, that are always presented as the common model for our community. Further benchmarks should be proposed each year, to compare and improve our tools on this common formalism, as well as to agree on the extensions we should cope with for the next benchmarks. This would ultimately lead to an increased expressivity of the formalisms handled by our tools, and to an enhanced common ground, necessary to publish our results and tools outside the CG community.

In our opinion, the most data model to consider after SCGs would be rules, expressing knowledge of form "If Hypothesis then Conclusion" as they are essential in all knowledge-based systems.

Let us make a try. A SCC-rule is classically defined as a pair of lambda SCCs, a lambda SCC is a pair composed of a SCG and a list of distinguished concepts of the graph (formal parameters), which are generic concepts. A SCC rule is a pair of lambda SCCs with a bijection between the two sets of distinguished concepts. The first SCC is the hypothesis of the rule, the second is the conclusion. Now, the base H of a module is composed of a set, of SCGs and a set of rules.

The logical semantic does not lead to any problem. In lambda SCCs variables assigned to distinguished nodes are kept free. The formula assigned to a rule is the universal closure of the formula of \(\text{hypothesis} \rightarrow \forall \text{conclusion}\).

The difficulty arises for defining rule application. A simple rule application consists of finding SCCs that have the same type in the hypothesis and in the conclusion, and correspondence links are not allowed inside the conclusion. These restrictions lead to a simple definition of a rule application. Rules are processed by a forward chaining mechanism; this was preserved sound and complete. In ANITA there is no restriction on the form of the rules (except that relation types are binary) and rules are processed by a Prolog-like backward chaining.

When rules are involved the deduction problem becomes the following: Given a module \(M(V \cup I, B)\) where \(B\) is composed of SCGs and rules, and a SCC \(Q\) defined on the same \(V\) and \(I\), is \(Q\) deducible from \(B\)?

A SCC is said to be deducible from \(B\) if it can be obtained from the SCGs of \(B\) by applying (a finite number of times) rules of \(B\). An answer is now a mapping (projection or exact-projection) from \(Q\) to a SCC deducible from \(B\).

Deduction on rules is not decidable (we could not obtain a computability model).

One can distinguish a specific case of rules, which is decidable and exactly corresponds to rules used in Datalog, called range-restricted or safe. In these rules no variable appear in conclusion expressed in CSGs (the conclusion part (excluding connection nodes) has no generic node). Examples of such rules are rule expressing properties of relations, as symmetry or transitivity.

Another data fragment that should be considered is that of (positive) nested conceptual graphs, if they are widely present in tools but unfortunately with variations. The nested description is either a kind of reference or a third field of the concept label. In CoGraf there is in addition a type of notations. A trouble concerning nested conceptual graphs is their logical semantic. The conclusion part proposed by Shaw not is in Datalog (as the special predicate expressing description has an argument which is a formula - the formula representing the nested sub-graph). A FOIL equivalent has been proposed in [CNS98]. As there is no general agreement on the logical semantics of nested CGs, defining the deduction and query answering problems is not easy.
Acknowledgements

We thank Vincent for allowing the CG community to freely use their topic map demonstration base.

References


