# Translations between RDF(S) and Conceptual Graphs

Jean-François Baget, Madalina Croitoru, Michel Leclère, Marie-Laure Mugnier

# Translations between RDF(S) and Conceptual Graphs

Jean-François Baget, Madalina Croitoru, Alain Gutierrez, Michel Leclère and Marie-Laure Mugnier

LIRMM (University of Montpellier II & CNRS), INRIA Sophia-Antipolis, France

**Abstract.** Though similarities between the Semantic Web language RDF(S) and languages of the Conceptual Graphs family have often been pointed out, the differences between these formalisms have been a source of difficulties while trying to translate objets of a language into the other. In this paper, we present two such transformations, that have been implemented into the CoGUI platform, and discuss their respective strengths and weaknesses.

## 1 Introduction

The scope of this paper is the problem of querying hybrid knowledge bases (KBs), i.e. with several components that can be expressed in different formalisms (Conceptual Graphs, RDF(S), OWL, relational model, etc.). The ontology itself can be described using different formalisms, but we make the assumption that the ontological knowledge it contains has the same meaning in all of the KBs considered (i.e. we do not address ontology alignment or mapping problems).

More specifically, we will focus on transformations between Conceptual Graphs (CGs) [10, 4] and the RDF(S) language [8], the standard for Semantic Web annotations. Given the scope of the paper, a fundamental property of such transformations is the preservation of the notion of semantic entailment (the basis for reasoning, hence querying). Other desirable properties are the natural aspect of the transformation, i.e. the conciseness and intuitiveness of the generated objects, as well as the preservation of some algorithmic properties of the language to be translated. Developing these transformations will not only provide a step towards querying hybrid KBs, but also benefit certain tasks on the Semantic Web [7] where structural, graph based optimisations (extensively addressed for Conceptual Graphs [9][4]) are needed.

In the following we detail two proposed transformations and study their properties. Both transformations preserve the semantic entailment, in a sense that we will precise, but they behave differently with respect to conciseness and intuitiveness. Both transformations are implemented in the tool CoGUI[1]. All graphs pictured in this paper are screenshots from an example designed with CoGUI and available on CoGUI's website.

## 2 Basic Conceptual Graphs: the core formalism of CoGUI

We recall in this section the main elements of the basic CG formalism. See [4] for details and tools CoGUI and CoGITaNT[2] for a faithful implementation of this framework. Ontological knowledge is encoded by a structure called the vocabulary, while factual knowledge is expressed by basic CGs.

[1] http://www.lirmm.fr/cogui/

[2] http://cogitant.sourceforge.net/

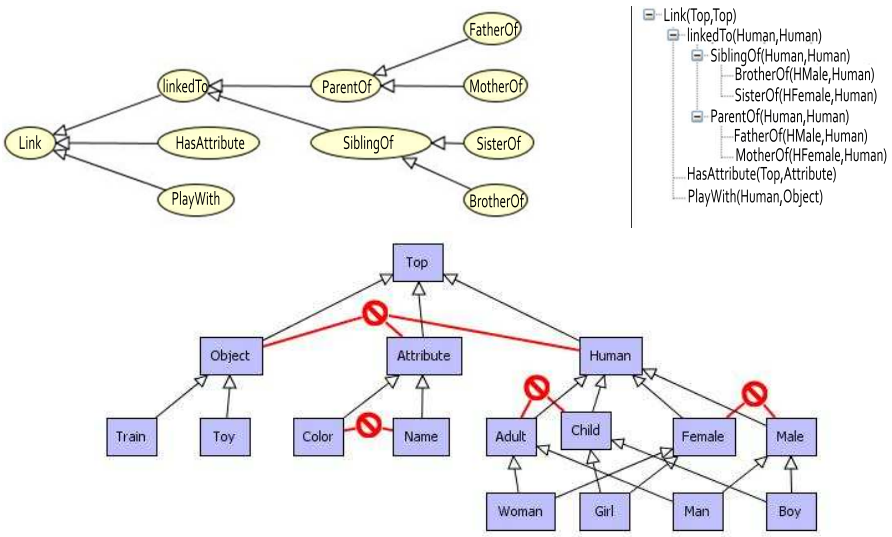


**Fig. 1.** Basic Conceptual Graphs (BG) Vocabulary

A *vocabulary* is basically a tuple $\mathcal{V} = (T_C, T_R = (T_R^1, \ldots, T_R^k), I)$ where $T_C$ is a partially ordered set of *concept types*, each $T_R^i$ is a partially ordered set of *relation types of arity* $i$, and $I$ is a set of *individual markers*. All these sets are pairwise disjoint and all partial orders are denoted by $\leq$. Other features may also appear in a vocabulary. A *conjunctive concept type* over a vocabulary $\mathcal{V}$ is a set $T = \{t_1, \ldots, t_p\}$ of concept types. If $T = \{t_1, \ldots, t_p\}$ and $T' = \{t'_1, \ldots, t'_q\}$ are two conjunctive concept types, then we also note $T \leq T' \Leftrightarrow \forall t'_i \in T', \exists t_j \in T$ such that $t_j \leq t'_i$. The *signature* $\sigma$ maps each relation type of arity $k$ to a $k$-tuple of conjunctive concept types, that encodes the maximal type of its arguments. The signature has the covariance property, meaning that if $r_2 \leq r_1$, then the $i^{th}$ argument of $r_2$ is a specialization of the $i^{th}$ argument of $r_1$. It is sometimes necessary, as in [1], to assert that an entity is an instance of several concept types. Finally, the vocabulary can be extended by adding *incompatibilities between (two) types*, i.e. asserting that a given conjunctive type is forbidden [6].

Fig. 1 depicts a hierarchy of relation types, their signature (e.g. $\sigma(MotherOf) = (HFemale, Human)$), and a hierarchy of concept types. The "forbidden" symbol encodes incompatibility (e.g. *Human*, *Attribute* and *Object* are pairwise incompatible).

A *basic conceptual graph* (BG) on a vocabulary $\mathcal{V} = (T_C, T_R, I)$ is a bipartite graph. The sets $C$ and $R$ contain respectively *concept* and *relation nodes*. A concept node $c \in C$ is labeled by a pair $(\textit{type}(c), \textit{marker}(c))$ where $\textit{type}(c)$ is a conjunctive concept type built on $T_C$ and *marker(c)* is either an individual marker of $\mathcal{I}$ – in that case $c$ is said *individual* – or the generic marker $*$, possibly named by a variable, as in $*x$ – then $c$ is said *generic*. A relation node $r \in R$ is labeled by $\textit{type}(r) \in T_R$ and is linked to $k$ concept nodes (its arguments), where $k$ is the arity of $\textit{type}(r)$.

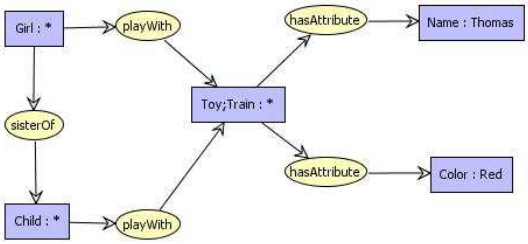


**Fig. 2.** BG Fact

Several concept nodes with the same individual marker or the same named generic marker denote the same entity. A BG is said to be *normal* when no two distinct concept nodes denote the same entity. Any BG can be transformed into a normal graph having the same logical semantics, called its normal form, by merging nodes that represent the same entity. The normal BG in Fig. 2, represents a fact stating that a child and its sister are playing with a toy train, which is has the name of Thomas and is red.

Note that the additional features of the vocabulary (signature and forbidden types) impose additional constraints on a BG. To respect the signature, the $i^{th}$ argument of a relation typed $r$ must be a specialization of the $i^{th}$ element of $\sigma(r)$; to satisfy forbidden types, no concept node can be of a type that is a specialization of a forbidden conjunctive type. Implemented in CoGUI, these features are used to check the validity of a BG and have no other impact on reasoning.

CGs can be translated into first-order logic by a mapping classically called $\Phi$ [10]. The BG fragment is equivalent to the existential positive conjunctive fragment of first-order logic [4]. The fundamental deduction problem in this fragment is as follows: given BGs $F$ and $Q$ defined over a vocabulary $\mathcal{V}$, is the formula $\Phi(Q)$ the logical deduction of the formulae $\Phi(F)$ and $\Phi(\mathcal{V})$ (noted $\Phi(\mathcal{V}), \Phi(F) \vdash \Phi(Q)$)? This problem can be recast as a query answering problem: is the conjunctive query $Q$ deducible from the KB composed of a set of facts $F$ and a lightweight ontology $\mathcal{V}$ ?

The basic notion for reasoning with BGs is a graph homomorphism ("projection" in the CG world). It provides a sound and complete reasoning mechanism:

**Theorem 1.** *[10] [5] Let $F$ and $Q$ be BGs on a vocabulary $\mathcal{V}$. Then $\Phi(\mathcal{V}), \Phi(F) \vdash \Phi(Q)$ iff there is a homomorphism from $Q$ to the normal form of $F$.*

Homomorphism checking (or deduction on BGs) is an NP-complete problem and is polynomial in data complexity (i.e. with respect to the size of $F$). Moreover, several cases with lower complexity can be obtained, mainly based on the structure of $Q$. Besides their interesting algorithmic properties, homomorphisms provide a visual way to express answers to a query $Q$. To compute homomorphisms, CoGUI relies upon a Client/Server architecture to communicate with the reasoning server CoGITaNT.

Apart from the vocabulary and the facts, CoGUI is also able to edit BG rules. These rules express knowledge of the form: "if hypothesis then conclusion", where the hypothesis and the conclusion are two BGs. In this paper, we will only consider simple

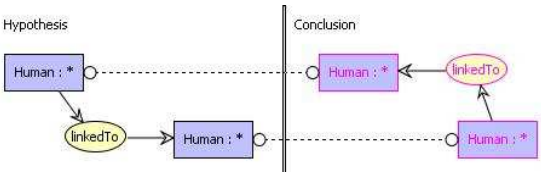


**Fig. 3.** BG+: Rule Example

rules, which do not increase the complexity of querying, i.e., rules that only add relation nodes or specialize the type of concept nodes (they are special cases of the so-called range-restricted rules in [3], [4]). See, for example, the rule in Fig. 3, where the relation "linked to" is deemed symmetrical. We denote by BG+ the BG fragment added with rules of the above mentioned form.

From now on, we simply note $K \vdash Q$ for $\Phi(K) \vdash \Phi(Q)$, where $K$ is a BG(+) KB (*i.e.* vocabulary, facts, and possibly rules) and $Q$ is a BG on the same vocabulary.

## 3 The Semantic Web language RDF(S)

RDF (Resource Description Framework) and its extension RDFS (RDF Schema) is a metadata model introduced by the W3C allowing the construction of semantic annotations given by a set of triples of the form *(subject, predicate, object)*. The RDF annotations are generally stored either in XML or in N3 files[3]. Fig. 4 shows a set of RDF triples in a simplified N3 notation, where names beginning with _ denote a *blank*, i.e. an anonymous resource; this set "naturally" corresponds to the BG in Fig. 2 (see Sect. 4.3). A set of RDF triples can be also visualized as a graph.

```
<:Red> <rdf:type> <:Color>.
<:Thomas> <rdf:type> <:Name>.
_:b1 <rdf:type> <:Toy>.
_:b1 <rdf:type> <:Train>.
_:b2 <rdf:type> <:Girl>.
_:b3 <rdf:type> <:Child>.
_:b1 <:hasAttribute> <:Red>.
_:b1 <:hasAttribute> <:Thomas>.
_:b2 <:playWith> _:b1.
_:b2 <:sisterOf> _:b3.
_:b3 <:playWith> _:b1.
```

**Fig. 4.** RDF Triples Corresponding to Fig. 2 Example

RDFS adds a lightweight ontological level structuring the vocabulary: it allows to declare classes and properties (binary predicates), to structure them by a preorder (*subClassOf* and *subPropertyOf* relations) and to define the signatures of properties via the notions of domain (*domain*) and co-domain (*range*). For instance, the following triples (in *(s,p,o)* form) "naturally" translate part of the BG vocabulary of Fig. 1. `(:Top,`

[3] The import / export tool of CoGUI uses the Jena (http://jena.sourceforge.net/) RDF parser for reading and outputting three formats: RDF/XML, N3 and TURTLE.

`rdf:type, rdfs:Class)`, `(:Human, rdf:type, rdfs:Class)` and `(:Human, rdfs:subClassOf, :Top)` express that `Top` and `Human` are classes (concept types) and that `Human` $\leq$ `Top`; while triples `(:MotherOf, rdf:type, rdf:Property)`, `(:MotherOf, rdfs:domain, :HFemale)`, `(:MotherOf, rdfs:range, :Human)` express that `MotherOf` is a property (relation) with signature `(HFemale, Human)`.

In this section, we define an RDF graph and three entailment relations of increasing preciseness: $\vdash_s$ (simple entailment), $\vdash_{rdf}$ (RDF entailment, which takes the so-called "RDF axiomatic triples" into account) and $\vdash_{rdfs}$ (RDFS entailment, which moreover takes the so-called "RDFS axiomatic triples" into account). Let us recall that $\vdash$ is the logical deduction in the BG fragment. The following definitions are reformulations of the ones provided by [8].

### 3.1 A Simple RDF

We first introduce a simplified version of RDF. Though its syntax remains the same, its semantics is weakened since no name is given any particular meaning. This RDF(S) fragment will be used as the building block in our transformations. On the other hand, we extend it to RDF$^*$, which allows to use variables/blanks as predicate names. This is an important feature in the perspective of implementing the SPARQL query language, whose basic graph patterns rely on such a possibility.

**Syntax** In what follows, we will consider 3 infinite pairwise disjoint sets of *terms*: the set $\mathcal{U}$ of *urirefs*, the set $\mathcal{L}$ of *literals*, and the set $\mathcal{B}$ of *blanks*. Among literals, we make a distinction between *plain literals*, and *typed literals*. A typed literal can be well-typed or ill-typed. The *value val*$(l)$ of a plain literal or an ill-typed literal $l$ is the literal itself, and the value of a well-typed literal is determined by its type. For example, the value of a typed literal whose type is `rdf:XMLLiteral` is the XML value of that literal. The only type that is currently taken into account in the RDF semantics is `rdf:XMLLiteral`. An *RDF vocabulary* is a subset of $\mathcal{U} \cup \mathcal{L}$.

**Definition 1 (RDF triple, RDF graph).** *An* RDF triple *is an element of* $(\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$. *An* RDF$^*$ triple *is an element of* $(\mathcal{U} \cup \mathcal{B}) \times (\mathcal{U} \cup \mathcal{B}) \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$. *The first element of a triple is called its* subject*, the second its* predicate*, and the third its* object*. An RDF graph is a set of RDF triples. An RDF$^*$ graph is a set of RDF$^*$ triples.*

Note that literals appearing as subject are classically forbidden both in RDF and RDF$^*$. This can be a problem since such triples can appear in reasonings. All further definitions and properties implicitly take that possibility in account.

If $G$ is an RDF$^*$graph, we call $\mathcal{U}$(G) (resp. $\mathcal{B}$(G), $\mathcal{L}(G)$, $\mathcal{T}(G)$) the set of *urirefs* (resp. *blanks*, *literals*, *terms*) appearing in $G$. An RDF$^*$ $G$ graph admits a natural graph representation: a node is assigned to each term appearing as a subject or object in $G$, and a directed edge to each triple of $G$; this edge admits for origin the node assigned to its subject, and for destination the node assign to its object.

**Semantics** In usual model-theoretic semantics, entities are mapped to elements of the interpretation domain and relations to a set of tuples of elements of the domain. Since RDF(S) does not consider a strict separation between entities and property names (which is considered as a requirement for the web), such an interpretation would lead to an important mathematical problem: an element of the domain could be asserted equal to a set of tuples containing it. By encoding the extension of a property into the interpretation structure, it is possible to lift that difficulty:

**Definition 2 (Interpretation).** *An* interpretation *of an RDF vocabulary $V$ is a triple $I = (\Delta, \iota, \epsilon)$ where $\Delta$ is a set of* resources *called the* interpretation *domain, $\iota : (V \cap \mathcal{U}) \rightarrow \Delta$ maps each uriref of the vocabulary to a resource, and $\iota : \Delta \rightarrow 2^{\Delta \times \Delta}$ maps each resource $d$ to a set of pairs of resources called the* extension *of $d$.*

**Definition 3 (Simple models).** *An interpretation $I = (\Delta, \iota, \epsilon)$ of a vocabulary $V$ is a* simple model *of an RDF or RDF* graph $G$ iff there exists a mapping $\pi : \mathcal{T}(G) \rightarrow \Delta$ that maps urirefs to their interpretation ($\forall u \in \mathcal{U} \cap V$, $\pi(u) = \iota(u)$); maps literals to their value ($\forall l \in \mathcal{L} \cap V$, $\pi(l) = \mathsf{val}(l)$); and preserves triples ($\forall (s, p, o) \in G$, $(\pi(s), \pi(o)) \in \epsilon(\pi(p))$).*

**Definition 4 (Simple Entailment).** *Let $F$ and $Q$ be RDF* graphs. We say that $F$* simply entails *$Q$ and note $F \vdash_s Q$ iff every simple model of $F$ is a simple model of $Q$.*

**Theorem 2.** *Let $F$ and $Q$ be RDF or RDF* graphs. Then $F \vdash_s Q$ iff there exists a mapping $\pi : \mathcal{T}(Q) \rightarrow \mathcal{T}(F)$ that maps urirefs to themselves ($\forall u \in \mathcal{U}(Q)$, $\pi(t) = t$); maps literals to literals with same value ($\forall l \in \mathcal{L}(Q)$, $\mathsf{val}(\pi(l)) = \mathsf{val}(l)$); and preserves triples ($\forall (s, p, o) \in Q$, $(\pi(s), \pi(p), \pi(o)) \in F$).*

### 3.2 RDF and RDFS axiomatic triples

RDF considers an infinite set $\mathcal{A}^{rdf}$ of triples said *axiomatic*, i.e. true for any RDF or RDF*graph. In the same way, RDFS considers the axiomatic set $\mathcal{A}^{rdfs}$. Both sets are infinite, due to the presence of an infinite set of properties `rdf:_i`. We can consider finite subsets by bounding the number of such properties allowed. If $k$ is a positive integer, we denote by $\mathcal{A}^{rdf}_k$ the *finite* subset of RDF axiomatic triples defined by $\mathcal{A}^{rdf}_k = \mathcal{A}^{rdf} \setminus \{(\texttt{rdf:\_i}, \texttt{rdf:type}, \texttt{rdf:Property}) | i > k\}$. The RDFS graph $\mathcal{A}^{rdfs}_k$ is defined in a similar way. Furthermore, RDF and RDFS consider a set of semantic conditions specifying the meaning embedded by special names of RDF(S).

**RDF semantics** We make here a simplification of RDFS semantics, since [8] modifies the structure of an interpretation by introducing a mapping $i' : \Delta \rightarrow \Delta$, used to define the extension of a class. But this is a redundant information, since $x \in i'(c)$ is defined as equivalent to "$x$ has type $c$", that we can already encode in RDF interpretations.

**Definition 5 (RDF and RDFS interpretations).** *An interpretation $I = (\Delta, \iota, \epsilon)$ of a vocabulary $V$ is an* RDF interpretation *of $V$ iff $I$ is a simple model for every RDF axiomatic triple, and $I$ satisfies each RDF semantic rule. If, moreover, $I$ is a simple model for every RDFS axiomatic triple and satisfies each RDFS semantic condition, then $I$ is said an RDFS interpretation.*

Before expliciting some of these semantic conditions, let us first define RDF and RDFS entailments:

**Definition 6 (RDF(S) Entailment).** *Let $F$ and $Q$ be RDF or RDF$^*$ graphs. We say that $F$* RDF entails *(resp.* RDFS entails*) $Q$ and note $F \vdash_{rdf} Q$ (resp. $F \vdash_{rdfs} Q$) iff every RDF (resp. RDFS) interpretation that is a model of $F$ is also a model of $Q$.*

*RDF semantic conditions* An interpretation $I = (\Delta, \iota, \epsilon)$ satisfies the RDF semantic condition iff:

1. for every resource $\delta \in \Delta$ with $\epsilon(\delta) \neq \emptyset$, $(\delta, \iota(\texttt{rdf:Property})) \in \epsilon(\iota(\texttt{rdf:type}))$;
2. for every typed literal $l$ whose type is `rdf:XMLLiteral`, $l$ is well-typed iff $(\mathit{val}(l), \iota(\texttt{rdf:XMLLiteral})) \in \epsilon(\iota(\texttt{rdf:type}))$;

*RDFS semantic rules* In the same way, an RDFS interpretation must satisfy some semantic conditions. A complete list of these conditions can be found in [8]. F.i., the two following semantic conditions state 1) that if a property $p$ has domain $c$ and $(s, p, o)$ is asserted, then $o$ has type $c$; and 2) that if $x$ has type $c$ and $c$ is a subclass of $c'$, then $x$ has type $c'$.

1. if $(p, c) \in \epsilon(\iota(\texttt{rdfs : domain}))$ and $(s, o) \in \epsilon(p)$, then $(s, c) \in \epsilon(\iota(\texttt{rdf : type}))$.
2. if $(x, c) \in \epsilon(\iota(\texttt{rdf : type}))$ and $(c, c') \in \epsilon(\iota(\texttt{rdfs : subClassOf}))$, then $(x, c') \in \epsilon(\iota(\texttt{rdf : type}))$.

**Computing RDF and RDFS entailment** When we have to compute whether $F \vdash_{rdf} Q$ (or $F \vdash_{rdfs} Q$), we will add to $F$ all necessary information to answer $Q$: first the axiomatic triples (at least a finite subset of them), then enrich it with all information that will force its simple model to be an RDF or RDFS interpretation. This can be done with the semantic rules of [8] that are used to generate a graph that respects all semantic conditions. A CG translation of one of these rules is presented in Fig. 7.

If $G$ is an RDF or RDF$^*$ graph, then its *saturation* $S^{rdf}_k(G)$ (or $S^{rdfs}_k(G)$) is obtained from $G$ as follows:

1. make the union of $G$ and $\mathcal{A}^{rdf}_k$ (or $\mathcal{A}^{rdfs}_k$);
2. enrich the obtained graph with RDF or RDFS rules until a fixpoint is obtained.

The two previous conditions translating RDF semantics can be written as the following rules:

1. for each triple of form $(s, p, o)$, add the triple $(p, \texttt{rdf:type}, \texttt{rdf:Property})$;
2. for each well-typed literal $l$ of $G$ whose type is `rdf:XMLLiteral`, add the triple $(l, \texttt{rdf:type}, \texttt{rdf:XMLLiteral})$.

RDFS semantic conditions can be translated in the same way (that is indeed done in [8]), and our two example semantic conditions can now be expressed as rules:

1. if there is a triple $(p, \texttt{rdfs:domain}, c)$ and a triple $(s, p, o)$ in the graph, then add the triple $(s, \texttt{rdf:type}, c)$.

2. if there is a triple $(x, \texttt{rdf:type}, c)$ and a triple $(c, \texttt{rdfs:subClassOf}, c')$ in the graph, then add the triple $(x, \texttt{rdf:type}, c')$.

*Property 1 (Satisfiability).* An RDF or RDF$^*$ graph $G$ is *RDF-satisfiable* (resp. *RDFS-satisfiable*) iff $S_0^{rdf}(G)$ (resp. $S_0^{rdfs}(G)$) does not contain any triple of form $(l, \texttt{rdf:type}, \texttt{rdf:XMLLiteral})$, where $l$ is an ill-typed literal whose type is `rdf:XMLLiteral`.

**Theorem 3 (RDF(S) Entailment Connection).** *Let $F$ and $Q$ be RDF or RDF$^*$ graphs. Then $F \vdash_{rdf} Q$ (resp. $F \vdash_{rdfs} Q$) iff either $F$ is not satisfiable or $S_k^{rdf}(F) \vdash_s Q$ (resp. $S_k^{rdfs}(F) \vdash_s Q$) where $k \geq 1$ is the greater number such that* `rdf:_k` *appears in $F$ or $Q$.*

## 4 The RDF/BG Transformations

It was pointed out [4] that RDF and CGs share very similar characteristics. Fig. 5 summarizes the main points of the "natural" correspondence between RDF(S) and BGs, along with their logical translation. However, such an intuitive translation does not satisfy our main evaluation criterion, which is the equivalence between reasonings in the two formalisms.

| *RDFS Triple* | *Equivalent BG* | *Logical Translation* |
|---|---|---|
| $C$ rdf:type rdfs:Class | $C$ concept type | $C$ unary predicate |
| $R$ rdf:type rdf:Property | $R$ binary relation type | $R$ binary predicate |
| $C$ rdfs:subClassOf $D$ | $C \leq D$ | $\forall x(C(x) \rightarrow D(x))$ |
| $R$ rdfs:subPropertyOf $S$ | $R \leq S$ | $\forall x \forall y(R(x,y) \rightarrow S(x,y))$ |
| $R$ rdfs:domain $C$ | $\sigma(R) = (C, -)$ | $\forall x \forall y(R(x,y) \rightarrow C(x))$ |
| $R$ rdfs:range $D$ | $\sigma(R) = (-, D)$ | $\forall x \forall y(R(x,y) \rightarrow D(y))$ |

**Fig. 5.** Correspondences between RDFS, BG and logic

### 4.1 Problems with the intuitive translation

Assume we want to encode the simple entailment in RDF or RDF$*$ within the basic CG fragment. Let us note $\mathcal{T}_{basic}$ this transformation. An RDF graph $G$ is encoded into a BG $\mathcal{T}_{basic}(G)$ in normal form as follows. For each term $t$ that appears either as subject or object in a triple of $G$, we create a concept node whose type is $\top$ and whose marker is a named generic marker $*t$ if $t$ is a blank, the individual marker $t$ if $t$ is a uriref and the individual marker *val*$(t)$ if $t$ is a literal. Then we merge literals that have the same value. Finally, for every triple $(s, p, o) \in G$, we add a relation node whose label is $p$ if $p$ is a uriref and $\top_2$ (with $\top_2$ being the maximal relation type for binary relations) if $p$ is

[4] http://www.w3.org/DesignIssues/CG.html

a blank, and whose arguments are respectively the node associated with $s$ and the node associated with $o$. The case where $p$ is a blank can happen only in RDF$^*$.

The next theorem expresses that, even if RDF does not distinguish between entities and relations, that does not prevent the BG homomorphism to be complete w.r.t. RDF simple entailment.

**Theorem 4.** *Let $F$ and $Q$ be RDF graphs. Then $F \vdash_s Q$ iff $\mathcal{T}_{basic}(F) \vdash \mathcal{T}_{basic}(Q)$.*

Things change when we consider the RDF$^*$language. Consider for instance the following triples and the translation of each of them into a BG:

1. $t_Q = (a, \_x, \_x)$, where $\_x$ is a blank, translated into $Q = [\top : a]\text{->} (\top_2)\text{->} [\top : *x]$;
2. $t_1 = (a, b, c)$, translated into $F_1 = [\top : a]\text{->} (b)\text{->} [\top : c]$;
3. $t_2 = (a, b, b)$, translated into $F_2 = [\top : a]\text{->} (b)\text{->} [\top : b]$.

One has $t_2 \vdash_s t_Q$ but not $t_1 \vdash_s t_Q$ since models of $t_1$ that map $b$ and $c$ to distinct elements are not models of $t_Q$. However, there is a BG homomorphism from $Q$ to both $F_1$ and $F_2$. The trouble is that the translation of $t_Q$ into a BG does not keep the information that the object and the predicate of the triple have the same variable name.

We present here two solutions solving that problem: in the first, we change the structure of the built BG, while in the second we restrict the domain of the translation to a subset of RDF.

### 4.2 The RDF/BG "3-hypergraphs" Transformation

The transformation $\mathcal{T}_3$ described in this section relies on the work of [2]. It is sound and complete w.r.t. RDF(S) semantics.

Let $G$ be an RDF or RDF$^*$graph. $\mathcal{T}_3$ encodes $G$ in a BG as follows. First, it assigns a concept node to every term appearing in $G$, with its marker being the same as in $\mathcal{T}_{basic}$; for now, we consider that its type is $\top$. Then, for every triple $(s, p, o)$ in $G$, it adds a relation node typed by `triple`, whose arguments are respectively the concept nodes assigned to $s$, $p$, and $o$.

It is immediate to check that an interpretation $I$ is a simple model of an RDF or RDF$^*$graph $G$ if and only if this interpretation is a model (in the CG sense) of the BG $\mathcal{T}_3(G)$. This transformation indeed encodes exactly the semantics of the language RDF. It follows that:

**Theorem 5.** *Let $F$ and $Q$ be RDF or RDF$^*$graphs. Then $F \vdash_s Q$ iff $\mathcal{T}_3(F) \vdash \mathcal{T}_3(Q)$.*

Now, let us enhance this transformation by adding relevant types to the concept nodes, thus translating the RDF vocabulary. The added hierarchy of concept types is depicted in Fig. 6 (it is indeed the same for all RDF or RDF$^*$input graphs). Some semantic rules require syntactic information in their hypothesis. This has to be taken into account in our syntactic transformations, thus, nodes translating an RDF term are typed according to the more specific syntactic category(ies).

To obtain the semantic completeness w.r.t. $\vdash_{rdf}$ and $\vdash_{rdfs}$, we add the translation of RDF(S) axiomatic triples as new facts, as well as the translation of the semantic rules

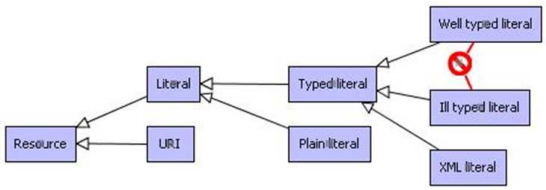


**Fig. 6.** $T_C$ depiction of RDF transformation $\mathcal{T}_3$

of RDF(S) as rules. For instance the RDFS "domain rule" presented previously can be translated into the BG rule pictured in Fig. 7. Note that such a rule relies upon generic concept nodes associated with a property, and thus uses the RDF* language.

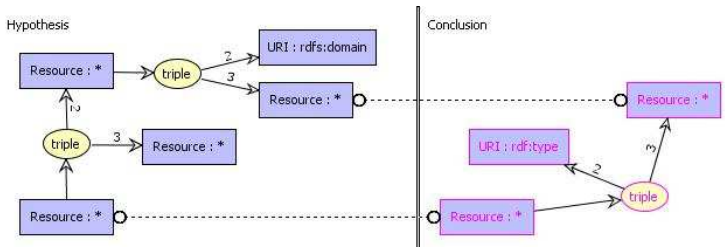


**Fig. 7.** BG+ depiction of an RDFS rule

Let us note $\mathcal{R}_{rdf}$ (resp. $\mathcal{R}_{rdfs}$) the set of BG rules associated with RDF (resp. RDFS) semantic rules. We can now express the equivalence between the BG and RDF(S) fragments based on transformation $\mathcal{T}_3$.

**Theorem 6.** *Let $F$ and $Q$ be RDF or RDF* graphs. Let $F_{rdf} = F \cup \mathcal{A}_k^{rdf}$, where $k \geq 1$ is the greatest integer such that* `rdf:_k` *appears in $F$ or $Q$. Then $F \vdash_{rdf} Q$ iff one of the following conditions is satisfied:*

- $\mathcal{T}_3(F_{rdf}), \mathcal{R}_{rdf} \vdash \mathcal{T}_3(\{(x, \texttt{rdf:type}, \texttt{rdf:XMLLiteral})\})$, *where $x$ is an ill-typed XMLLiteral typed literal.*
- $\mathcal{T}_3(F_{rdf}), \mathcal{R}_{rdf} \vdash \mathcal{T}_3(Q)$.

*The same property, obtained by substituting $_{rdf}$ with $_{rdfs}$, holds for RDFS entailment.*

The transformation $\mathcal{T}_3$ thus fulfills our main requirement: preserving the notion of entailment between RDF(S) and BGs. However, this transformation has severe drawbacks from a user perspective. First, the triples are harder to read than the binary relation they encode, and this default is made worse when saturating the graph by the application of rules. The second drawback is that $\mathcal{T}_3$, faithful to RDF, does not offer knowledge

structuring. This was already one of the main criticisms addressed to semantic networks, and CGs answered that by establishing a clear distinction between factual and ontological knowledge. These are the drawbacks addressed in the next transformation.

### 4.3 The RDF/BG Intuitive Transformation

The second transformation, called $\mathcal{T}_{nat}$ and outlined in Fig. 5, has several qualities:

- It is natural (hence the notation $\mathcal{T}_{nat}$), in sense that it respects the kinds of knowledge: it translates classes into concept types (both represent sets of entities), properties into binary relations, and instance into instances.
- It preserves the visual qualities of the graph.
- It allows for a clear distinction between ontological and factual knowledge.

Due to the latter quality, $\mathcal{T}_{nat}$ cannot not translate RDF(S) entirely. However, we claim that it allows to translate exactly the subset of RDFS used for representing knowledge with the purpose of querying factual knowledge, i.e. typical semantic annotations. We will denote this fragment of RDF(S) corresponding to BGs by RDFS-. Depending on the way we translate the signatures of properties, we will also obtain some rules that do not increase the complexity of deduction.

We define a transformation from RDFS- to BG(+) and a transformation from BG(+) to RDFS- in such way that these transformations are reciprocal one with respect to the other. Amongst the triples allowed in RDFS- we distinguish between *ontological* triples (corresponding to the vocabulary), *factual* triples (corresponding to the facts) and also *commentary* triples (corresponding to the elements not belonging to the CG formalism but present in CoGXML, the XML file format of CoGUI). The completeness result obtained states that, as long as the document to be entailed is composed only of RDF triples (as opposed to RDFS triples), then BG deduction is complete w.r.t. RDF(S) entailment. This means that this transformation is well-suited to the deduction of factual knowledge but not to ontological knowledge.

**The RDFS- Fragment** In the notations below, *B* stands for *Blanks*, *L* for *Literals* and *SU* for *simple URIs*, that is URIs not starting by `rdf:` or `rdfs:`. We further refine *SU* into *SUc* for the *SU* belonging to classes, *SUp* for properties and *SUi* for instances. The following triple patterns are allowed in RDFS-:

- *Ontological* triples: (*SUc*, `rdf:type`, `rdfs:Class`), (*SUc*, `rdfs:subClassOf`, *SUc*), (*SUp*, `rdf:type`, `rdf:Property`), (*SUp*, `rdfs:subPropertyOf`, *SUp*), (*SUp*, `rdfs:domain`, *SUc*) and (*SUp*, `rdfs:range`, *SUc*);
- *factual* triples: (*SUi*, `rdf:type`, *SUc*), (*B*, `rdf:type`, *SUc*) and (*s*, *SUp*, *o*), where *s* is either *B* or *SUi*, and *o* is either *B*, *SUi* or *L*;
- *commentary* triples: (*SU*, `rdfs:label`, *L*) and $(SU, \texttt{rdfs} : \texttt{comment}, L)$.

We do not allow for anonymous classes or properties: for this reason, we forbid ontological triples containing a blank either as a subject or as an object, as well as factual triples containing a blank as a property, or as an object when the predicate is rdf:type. Finally, the *separability* condition has to be fulfilled: a given *SU* can appear only in one of the categories "class" (*SUc*), "property" (*SUp*) and "instance" (*SUi*). In terms of CGs, this condition states that the sets $T_C$, $T_R$ and $I$ are pairwise disjoint.

**Transformation $\mathcal{T}_{nat}$: RDFS- towards BG(+)** In the description below, an element is added to the vocabulary or the fact graph only if it does not already exist. Any addition of a relation (obviously binary) is done by default with the signature (*rdfs:Resource*, *rdfs:Resource*). Further domain and range statements will induce a specialization of this signature. *rdfs:Resource* behaves as the top of the hierarchy. This specialization can be performed in two different ways: either by specializing the signature in the vocabulary directly, or by introducing a rule translating this specialization.

The transformation takes place as follows:

1. Creation of the concept types *rdfs:Resource*, *rdfs:Literal* and *rdfs:Datatype*, as well as the relation type *rdf:Property(rdfs:Resource, rdfs:Resource)*
2. Treatment of ontological triples:
   - (*SUc*, `rdf:type`, `rdfs:Class`) → addition of the concept type *SUc*
   - (*SUc1*, `rdfs:subClassOf`, *SUc2*) → addition of concept types *SUc1* and *SUc2*, where *SUc1* ≤ *SUc2*
   - (*SUp*, `rdf:type`, `rdf:Property`) → addition of the relation type *SUp*
   - (*SUp1*, `rdfs:subPropertyOf`, *SUp2*) → addition of binary relations *SUp1* and *SUp2* with *SUp1* ≤ *SUp2*
   - (*SUp*, `rdfs:domain`, *SUc*) → addition of the binary relation type *SUp*, of the concept type *SUc*, and treatment of the domain information (as explained above)
   - (*SUp*, `rdfs:range`, *SUc*) → similar to above
3. Treatment of factual triples (f.i. $\mathcal{T}_{nat}$ applied to the triples in Fig. 4 yields the BG in Fig. 2)
   - (*SUi*, `rdf:type`, *SUc*) → addition of the concept node *[SUc : SUi]*, addition to the vocabulary of the individual marker *SUi* and of the concept type *SUc*
   - (*B*, `rdf:type`, *SUc*) → similar to above with the only difference of the generic marker, i.e. the node *[SUc :*B]* is obtained
   - triples of form (*s*, *SUp*, *o*) → addition of the corresponding concept and relation nodes, along with the type insertions into the vocabulary
4. The commentary triples, i.e. containing `rdfs:comment` or `rdfs:label` are translated in labels and commentaries in CoGXML

When a document is violating the separability condition, there are several possibilities for dealing with the triples that are causing this violation. A drastic solution consists in rejecting the RDF(S) document. Another solution consists in only accepting a subset of the RDF(S) document that satisfies the separability condition: in this case, the choices made have to be independent of the order in which the triples have been analyzed, so that two RDFS documents with the same set of triples, thus semantically equivalent, are translated in the same way. Currently, the implemented solution consists in rejecting the RDF(S) documents violating the separability between the concept and relation type hierarchies. For the separation of individual markers with the concept / relation hierarchy, the priority is given to declarations concerning classes and properties.

**Transformation $\mathcal{T}_{nat^-}$: BG towards RDFS-** We assume that all the relations are binary. If not, we can first "binarize" the BGs. Binary BGs can be easily translated in RDFS-:

1. The vocabulary is translated into ontological triples:
   - For all concept types $t \rightarrow$ ($t$, `rdf:type`, `rdfs:Class`)
   - For all concept types $t_1$ and $t_2$ s. t. $t_2 \leq t_1 \rightarrow$ ($t_2$, `rdfs:subClassOf`, $t_1$)
   - For all relations $r \rightarrow$ ($r$, `rdf:type`, `rdf:Property`)
   - For all signatures ($t_1$,$t_2$) of a relation $r \rightarrow$ ($r$, `rdfs:domain`, $t_1$), ($r$, `rdfs:range`, $t_2$)
   - For all relations $r_1$ and $r_2$ s. t. $r_2 \leq r_1 \rightarrow$ ($r_2$, `rdfs:subPropertyOf`, $r_1$)
2. The fact graphs are translated into factual triples (f.i. $\mathcal{T}_{nat^-}$ applied to the BG in Fig. 2 yields the triples in Fig. 4):
   - We assign a different blank to each generic concept node. The term assigned to a generic concept node is the above mentioned blank and the one assigned to an individual concept node is the URI corresponding to its individual marker;
   - For all concept nodes of type $t_1$,..., $t_n$ and associated term $e$, we have: for $i$ from 1 to $n \rightarrow$ ($e$, `rdf:type`, $t_i$)
   - For all relation nodes $r$ having as the first neighbor $c_1$ and second neighbor $c_2$ with the associated terms $e_1$ respectively $e_2 \rightarrow$ ($e_1$, $r$, $e_2$)
3. The commentaries and labels associated to concept and relation types are translated by commentary triples $\rightarrow$ ($t$, `rdfs:comment`, *literal*), ($t$, `rdfs:label`, *literal*).

Apart from n-ary relation types, the only element that we cannot translate into RDFS is the notion of forbidden conjunctive type (expressing that two concept types are disjoint). Note it can be translated by the OWL predicate owl:disjointWith (see Sect. 5).

Note that the rules associated with signatures could be translated into RDFS- (which is not implemented yet in the transformation provided by CoGUI).

**Properties of $\mathcal{T}_{nat}$ and $\mathcal{T}_{nat^-}$** These transformations are "essentially" reciprocal, in the following sense: their composition is the identity, up to a fixed set of axiomatic knowledge, which is made explicit by the transformation, but has no incidence on the semantics of the transformed knowledge. More precisely:

*Property 2.* Let $K$ be an RDF graph (resp. a vocabulary and a BG $F$). Let $f$ be $\mathcal{T}_{nat^-} \circ \mathcal{T}_{nat}$ (resp. $\mathcal{T}_{nat} \circ \mathcal{T}_{nat^-}$). Then $K' = f(K) = K \cup A$, where $A$ is a fixed set of triples (resp. $F \cup A$, where $A$ is a fixed part of the vocabulary), and $f(K') = K'$.

The following result specifies the kind of completeness obtained:

**Theorem 7.**
*Let $G_1$ and $G_2$ be RDFS- graphs such that $G_2$ contains solely factual triples.*
*Then $G_1 \vdash_{rdfs} G_2$ iff $\mathcal{T}_{nat}(G_1) \vdash \mathcal{T}_{nat}(G_2)$.*
*Let $F$ and $Q$ be BGs on a vocabulary $\mathcal{V}$.*
*Then $\mathcal{V}, F \vdash Q$ iff $\mathcal{T}_{nat^-}(\mathcal{V}) \cup \mathcal{T}_{nat^-}(F) \vdash_{rdfs} \mathcal{T}_{nat}(Q)$.*

## 5 Perspectives

Let us emphasize the interest of the second transformation, i.e. $\mathcal{T}_{nat}$. This transformation is in line with the knowledge representation vision of the Semantic Web, in the sense that it clearly distinguishes between different kinds of knowledge. Moreover, CoGUI allows to visualize the knowledge base obtained according to this separation. It defines a fragment of RDF(S) that can be provided with a semantics in classical first-order logics, which is compatible with most description logics, in particular the OWL-DL fragment. Hence, it is potentially extensible to take a richer ontology into account, that would be represented by description logics.

As a matter of fact, many RDFS files use simple OWL features. The combination of RDFS and OWL-DL in documents leads to a combinatorial explosion of the querying problem. Recently, restrictions of OWL-DL have been proposed to overcome this explosion (see OWL2 [5]). We are currently studying the precise relationships between these restrictions and fragments of CGs in the context of query answering. As a preliminary and pragmatic work, we have extended the $\mathcal{T}_{nat}$ translation to some OWL statements which can be expressed in BG+, thus without increasing complexity of reasoning.

## References


1. J.-F. Baget. Simple Conceptual Graphs Revisited: Hypergraphs and Conjunctive Types for Efficient Projection Algorithms. In *Proc. of ICCS'03*, volume 2746 of *LNAI*. Springer, 2003.
2. J. F. Baget. RDF Entailment as a Graph Homomorphism. In *Proc. of ISWC'05*, 2005.
3. J.-F. Baget and M.-L. Mugnier. The Complexity of Rules and Constraints. *JAIR*, 16:425–465, 2002.
4. M. Chein and M. Mugnier. *Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs*. Springer, 2009.
5. M. Chein and M.-L. Mugnier. Conceptual Graphs: Fundamental Notions. *Revue d'Intelligence Artificielle*, 6(4):365–406, 1992.
6. M. Chein and M.-L. Mugnier. Concept types and coreference in simple conceptual graphs. In K. E. W. et al, editor, *ICCS*, volume 3127 of *LNAI*, pages 303–318. Springer, 2004.
7. O. Corby, R. Dieng, and C. Hebert. A Conceptual Graph Model for W3C RDF. In *Proceedings of ICCS00*, volume 1867 of *LNAI*. Springer, 2000.
8. P. Hayes, editor. *RDF Semantics*. W3C Recommendation. W3C, 2004.
9. M.-L. Mugnier. Knowledge Representation and Reasoning based on Graph Homomorphism. In *Proc. ICCS'00*, volume 1867 of *LNAI*, pages 172–192. Springer, 2000.
10. J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.


[5] http://www.w3.org/TR/owl2-profiles/