Semantics of Constraints in RDFS

Álvaro Cortés-Calabuig[1,2], Jan Paredaens[2]

[1] Vrije Universiteit Brussel, Belgium
alvaro.cortes@vub.ac.be
[2] University of Antwerp, Belgium
jan.paredaens@ua.ac.be

Abstract. We study constraints for RDF-Schema (RDFS) graphs. The syntax and semantics is defined for constraints in graphs that can contain RDFS properties and blank nodes. The proposal for constraint satisfaction closely resembles the possible world approach found in various contexts of incomplete databases and knowledge bases. Positive decidability results for checking satisfaction of RDFS constraints under blank nodes are given. In addition, we present deductive rules for different kinds of constraints and for several combination of them. Our approach resembles similar deductive rules of relational database contexts, but they are generalized and adapted to the RDFS data model using homomorphisms and embeddings.

1 Introduction

RDF (Resource Description Framework) [8] is a World Wide Web Consortium (W3C) recommendation for publishing structured data on the web. RDF-graphs arrange information in simple triples consisting of a subject, a property (sometimes referred to as predicate) and an object. Because of this structure, RDF-graphs are usually conceptualized as directed graphs, where the property of a triple is a directed edge between subject and object. Unlike standard graphs, however, in RDF the set of nodes and edges is not disjunctive, as the same symbol can represent both a property and a domain object.

RDF-graphs convey information about a certain domain under consideration. As with any other data format, in order to specify conditions that must be satisfied by an RDF-graph, constraints need to be imposed. In practical settings, constraints are commonly used for three main tasks: specifying properties of the data itself; handle contradictions within the database or with respect to the domain under consideration; or as a help for semantic query optimizations.

In recent years, important progress has been made towards extending the basic RDF framework with constraints. To the best of our knowledge, the proposals so far, however, exclude the use of RDFS properties. This simplification restricts the usability of such constraints in real world domains, where the use of RDFS coupled with blank nodes is common practice. Hence, one specific goal of this paper is to tackle this issue by extending the semantics of the RDF constraints introduced in [1] with RDFS properties and blank nodes.

In general terms, this paper is an effort towards a general theory of reasoning with constraints in RDF. The concrete contributions of this paper are as follows. 1. We define the semantics of several types of constraints for RDFS-graphs with schema properties and blank nodes; 2. We present a number of decidability results concerning the satisfaction of constraints under blank nodes in RDFS-graphs; 3. We provide sound and complete deductive rules for different kinds of constraints and for several combination of them; and 4. We study how RDFS properties can be expressed by a number of additional triple generating constraints. We deal with subclass, type, subproperty, domain and range.

This paper is organized as follows. In Section 2 we introduce the basic terminology and preliminary definitions we use throughout the paper. In Section 3 we study decidability problems on RDF constraints satisfaction and entailment. In Section 4, we extend our framework to RDFS graphs. Related work is surveyed in Section 5, which also contains our conclusions.

2 Preliminaries Constraints

In what follows, we use the following vocabulary:

- $\mathcal{U}$, is an infinite set of *URI*s;
- V, an infinite set of *variables* denoted by prefixing them by \$;
- $\mathcal{B}$ is an infinite set of *blank nodes* prefixed by _.

$\mathcal{U}, V$ and $\mathcal{B}$ are pairwise disjoint.

Let t_1 and t_2 be two terms. $\phi_{t_2 \leftarrow t_1}$ denotes the function that is equal to the identity, except that $\phi_{t_2 \leftarrow t_1}(t_1) = t_2$.

A *homomorphism* from a set of triples of terms S in a set of triples of terms S' is a total function[1] $h : V_S \cup \mathcal{U}_S \to V_{S'} \cup \mathcal{U}_{S'}$, such that

- $h(u) = u$, for each $u \in \mathcal{U}_S$;
- If $(t_1, t_2, t_3) \in S$ then $(h(t_1), h(t_2), h(t_3)) \in S'$;

Definition 1 (RDF-graph). *An* RDF-graph *(or* graph*)* $\mathcal{G}$ *is a finite set of triples* (s, p, o)*, subject, property, object,* $s, o \in \mathcal{U} \cup \mathcal{B}$*,* $p \in \mathcal{U}$*. A graph is* grounded *if it does not contain blank nodes.*

A *term* is an element of $V \cup \mathcal{U}$.
In the following we denote $\mathcal{U}_\mathcal{G}$ and $\mathcal{B}_\mathcal{G}$ the set of elements of $\mathcal{U}$ and $\mathcal{B}$ that occur in $\mathcal{G}$.

Definition 2 (Grounding). *A* grounding θ *of a graph* $\mathcal{G}$*, is a total function* $\theta : \mathcal{B}_\mathcal{G} \cup \mathcal{U}_\mathcal{G} \to \mathcal{U}$*, where* $\theta(u) = u$*, for* $u \in \mathcal{U}_\mathcal{G}$*. The* grounded graph $\mathcal{G}_\theta$ *of* $\mathcal{G}$ *under* θ *is defined as* $\mathcal{G}_\theta = \{(\theta(t_1), \theta(t_2), \theta(t_3)) \mid (t_1, t_2, t_3) \in \mathcal{G}\}$*. The set of grounded graphs of a graph* G *is denoted*

$$\mathfrak{G}_\mathcal{G} = \{\mathcal{G}_\theta \mid \theta \text{ is a grounding of } \mathcal{G}\}.$$

[1] h is not necessary a surjection.

Note that a grounded graph does not contain blank nodes and that every grounded graph is a graph with only one grounding, i.e. the identity.

Definition 3 (Embedding of a set of triples of terms in $\mathcal{G}_\theta$). *An* embedding *of a finite set S of triples of terms in a graph $\mathcal{G}_\theta$ is a total function $e : V_S \cup \mathcal{U}_S \to \mathcal{U}$[2], such that*

- *$e(u) = u$, for each $u \in \mathcal{U}$;*
- *If $(t_1, t_2, t_3) \in S$ then $(e(t_1), e(t_2), e(t_3)) \in \mathcal{G}_\theta$.*

3 Constraints in RDF

In this section we discuss four types of RDF-constraints: Equality Generating, Functional, Triple Generating and Forbidding Constraints.

In general an *RDF-constraint* is a condition which a graph can satisfy or not. We say that a graph $\mathcal{G}$ *satisfies* an RDF-constraint $\mathcal{C}$ if all the grounded graphs of $\mathfrak{G}_\mathcal{G}$ satisfy $\mathcal{C}$[3]. We denote that a graph $\mathcal{G}$ satisfies an RDF-constraint $\mathcal{C}$ as $\mathcal{G} \models \mathcal{C}$.

3.1 Types of Constraints

In [1] the following two types of constraints are introduced:

Definition 4 (EGC). *An* equality generating constraint *is a pair (S, E), where*

- *S is a finite set of triples of terms;*
- *E is a finite set of equalities, each of the form $(t_1 = t_2)$, with $t_1, t_2 \in V_S \cup \mathcal{U}$.*

A grounded graph $\mathcal{G}$ satisfies the EGC (S, E) iff for every embedding e of S in $\mathcal{G}$, and every $(t_1 = t_2) \in E$ holds that $e(t_1) = e(t_2)$.

Example 1. Consider the constraint $\mathcal{C}_1 = (\{(\$x, a, \$y)\}, \{(\$x = \$y)\})$.

$$\{(b, b, c), (b, b, {}_{-a})\} \models \mathcal{C}_1 \qquad \{({}_{-a}, a, c), (b, a, {}_{-a})\} \not\models \mathcal{C}_1$$
$$\{({}_{-a}, a, {}_{-b}), ({}_{-b}, a, {}_{-b})\} \not\models \mathcal{C}_1 \quad \{({}_{-a}, b, {}_{-b}), ({}_{-a}, b, {}_{-c})\} \models \mathcal{C}_1.$$

In a similar way as with EGCs, we proceed with defining functional constraints.

Definition 5 (FC). *A* functional constraint *is a pair $(S, L \to R)$, where*

- *S is a finite set of triples of terms;*
- *$L, R \subseteq V_S$.*

[2] We denote $\mathcal{U}_S$ (resp. $\mathcal{B}_S$ and V_S) for the set of elements of $\mathcal{U}$ (resp. $\mathcal{B}$ and V) that occur in S.

[3] The notion of satisfaction for grounded graphs is separately defined below for each type of constraint.

A grounded graph $\mathcal{G}$ satisfies the FC $(S, L \to R)$ iff for every two embeddings of S in $\mathcal{G}$ that coincide[4] *on the variables of L, they also coincide on the variables of R.*

Example 2. Consider the constraint $\mathcal{C}_2 = (\{(\$x, a, \$y)\}, \{\$x\} \to \{\$y\})$.

$$\{(b, b, c), (b, b, {}_{-a})\} \models \mathcal{C}_2 \quad \{({}_{-a}, a, c), (b, a, {}_{-a})\} \not\models \mathcal{C}_2$$
$$\{({}_{-a}, a, {}_{-b}), ({}_{-b}, a, {}_{-b})\} \models \mathcal{C}_2 \; \{({}_{-a}, b, {}_{-b}), ({}_{-a}, b, {}_{-c})\} \models \mathcal{C}_2.$$

In [1] the following lemma is proved.

Lemma 1. *A functional constraint can be expressed in terms of equality generating constraints.*

Definition 6 (TGC). *A* triple generating constraint *is a pair (S, S'), where S and S' are both finite sets of triples of terms and $V_{S'} \subseteq V_S$. A grounded graph $\mathcal{G}$ satisfies the TGC (S, S') if every embedding e of S in $\mathcal{G}$ is also an embedding of S' in $\mathcal{G}$.*

Example 3. Consider the constraint $\mathcal{C}_3 = (\{(\$x, a, \$y)\}, \{(\$y, a, \$x)\})$.

$$\{(b, a, c), (c, a, b)\} \models \mathcal{C}_3 \; \{({}_{-a}, a, c), (c, a, {}_{-b})\} \not\models \mathcal{C}_3.$$

While EGCs, FCs and TGCs require properties for all possible embeddings, forbidding constraints forbid some embeddings.

Definition 7 (FBC). *A* forbidding constraint *has the form (S), where S is a finite set of triples of terms. A grounded graph $\mathcal{G}$ satisfies the FBC (S) iff there is no embedding of S in $\mathcal{G}$.*

Example 4. Consider $\mathcal{C}_4 = (\{(\$x, a, \$x)\})$

$$\{(b, a, b)\} \not\models \mathcal{C}_4 \quad \{(a, b, a)\} \models \mathcal{C}_4$$
$$\{({}_{-a}, b, {}_{-c})\} \models \mathcal{C}_4.$$

Since there is a finite number of embeddings of a finite set of triples into a grounded graph, by Lemma 1 we can easily prove

Theorem 1. *It is decidable whether a grounded graph $\mathcal{G}$ satisfies an EGC (S, E), an FC $(S, L \to R)$, a TGC (S, S') or a FBC (S).*

Let $\mathcal{SC}$ be a finite set of constraints and $\mathcal{C}$ be a constraint, we say that $\mathcal{C}$ is a *logical grounded consequence* of $\mathcal{SC}$ iff for all grounded graphs $\mathcal{G}$ holds

$$(\forall \mathcal{C}' \in \mathcal{SC}(\mathcal{G} \models \mathcal{C}')) \Rightarrow \mathcal{G} \models \mathcal{C}$$

[4] Two embeddings e and e' coincide on a variable $\$v$ iff $e(\$v) = e'(\$v)$.

3.2 Logical Grounded Consequences

Theorem 2. *Given a finite set $\mathcal{E}$ of EGCs. It is decidable whether an EGC (S, E) is a logical grounded consequence of $\mathcal{E}$.*

By Lemma 1 and Theorem 2 we have

Theorem 3. *Given a finite set $\mathcal{F}$ of FCs. It is decidable whether an FC $(S, L \rightarrow R)$ is a logical grounded consequence of $\mathcal{F}$.*

Theorem 4. *Given a finite set $\mathcal{T}$ of TGCs. It is decidable whether a TGC (S, S') is a logical grounded consequence of $\mathcal{T}$.*

Proof. Initialize the graph $\overline{S}$ as S where each variable $\$x$ is substituted by a new URI x. We denote this homomorphism $h_{\$}$. As long as there is a $(S_1, S_1') \in \mathcal{T}$ and an embedding e of S_1 in the graph $\overline{S}$ with $e(S_1') \not\subseteq \overline{S}$, substitute $\overline{S}$ by $\overline{S} \cup e(S_1')$. Consider now the final graph $\overline{S}$. $\overline{S} \models (S, S')$ iff (S, S') is a logical grounded consequence of $\mathcal{T}$. □

Theorem 5. *Given a finite set $\mathcal{FB}$ of FBCs. It is decidable whether an FBC (S) is a logical grounded consequence of $\mathcal{FB}$.*

Proof. Consider the graph $\overline{S}$ as S where each variable $\$x$ is substituted by a new URI x. $\exists (S') \in \mathcal{FB}$ with $\overline{S} \models (S')$ iff (S) is a logical grounded consequence of $\mathcal{FB}$. □

Let $\mathcal{T}$ be a finite set of TGCs and $\mathcal{E}$ be a finite set of EGCs. $\mathcal{T} \cup \mathcal{E}$ can have more logical grounded EGC consequences than $\mathcal{E}$. Indeed, $(\{(a, \$x, \$y)\}, \{(\$y = a)\})$ is a logical grounded consequences of $\{(\{(a, \$x, \$y)\}, \{(a, a, \$y)\}), (\{(a, a, \$y)\}, \{(\$y = a)\})\}$ but not of $\{(\{(a, a, \$y)\}, \{(\$y = a)\})\}$.
$\mathcal{T} \cup \mathcal{E}$ can also have more logical grounded TGC consequences than $\mathcal{T}$. Indeed, $(\{(a, \$x, \$y)\}, \{(a, a, \$x)\})$ is a logical grounded consequences of $\{(\{(a, \$x, \$y)\}, \{(\$y = a)\}), (\{(a, \$x, a)\}, \{(a, a, \$x)\})\}$ but not of $\{(\{(a, \$x, a)\}, \{(a, a, \$x)\})\}$.

Theorem 6. *Given a finite set $\mathcal{TE}$ of TGCs and EGCs. It is decidable whether a TGC (S, S') is a logical grounded consequence of $\mathcal{TE}$. It is decidable whether an EGC (S, E) is a logical grounded consequence of $\mathcal{TE}$.*

Let $\mathcal{T}$ be a finite set of TGCs and $\mathcal{FB}$ be a finite set of FBCs. $\mathcal{T} \cup \mathcal{FB}$ can have more logical grounded FBC consequences than $\mathcal{FB}$. Indeed, $(\{(a, \$x, \$y)\})$ is a logical grounded consequences of $\{(\{(a, \$x, \$y)\}, \{(a, a, \$y)\}), (\{(a, a, \$y)\})\}$ but not of $\{(\{(a, a, \$y)\})\}$.
$\mathcal{T} \cup \mathcal{FB}$ has exactly the same logical grounded TGC consequences as $\mathcal{T}$.

Theorem 7. *Given a finite set $\mathcal{TFB}$ of TGCs and FBCs. It is decidable whether a TGC (S, S') is a logical grounded consequence of $\mathcal{TFB}$. It is decidable whether a FBC (S) is a logical grounded consequence of $\mathcal{TFB}$.*

3.3 Deductive Rules for Grounded Graphs

Next we are giving a *sound, independent and complete set of deductive rules for grounded graphs*. A set of deductive rules defines when $\mathcal{SC} \vdash \mathcal{C}$, with $\mathcal{SC}$ is a set of constraints and $\mathcal{C}$ a constraint. We prove each time that

$$\mathcal{SC} \vdash \mathcal{C} \Rightarrow \forall \mathcal{G}_G((\forall \mathcal{C}' \in \mathcal{SC}(\mathcal{G}_G \models \mathcal{C}')) \Rightarrow \mathcal{G}_G \models \mathcal{C}), \text{soundness}$$

$$\mathcal{SC} \vdash \mathcal{C} \Leftarrow \forall \mathcal{G}_G((\forall \mathcal{C}' \in \mathcal{SC}(\mathcal{G}_G \models \mathcal{C}')) \Rightarrow \mathcal{G}_G \models \mathcal{C}), \text{completeness}$$

$$\forall \text{ Rule } r(\exists \mathcal{C}(\mathcal{SC} \vdash \mathcal{C} \wedge \mathcal{SC} \nvdash_r \mathcal{C})), \text{independent}$$

where $\mathcal{G}_G$ are grounded graphs and the subscript r means "without using Rule r".

Let $\mathcal{E}$ be a finite set of EGCs. In [1], the following set of deductive rules for EGCs Rules E0-E8 is proved to be sound, independent and complete for grounded graphs:

Rule E0 : $\mathcal{E} \vdash (S, E)$, for every $(S, E) \in \mathcal{E}$;
Rule E1 : $\mathcal{E} \vdash (S, \{(t = t)\})$, for every finite set S of triples of terms and $t \in V_S \cup \mathcal{U}_S$;
Rule E2 : $\mathcal{E} \vdash (S, \{(t_1 = t_2)\})$ implies $\mathcal{E} \vdash (S, \{(t_2 = t_1)\})$;
Rule E3 : $\mathcal{E} \vdash (S, \{(t_1 = t_2), (t_2 = t_3)\})$ implies $\mathcal{E} \vdash (S, \{(t_1 = t_3)\})$;
Rule E4 : $\mathcal{E} \vdash (S, E)$ and $E_1 \subseteq E$ implies $\mathcal{E} \vdash (S, E_1)$;
Rule E5 : $\mathcal{E} \vdash (S, E_1)$ and $\mathcal{E} \vdash (S, E_2)$ implies $\mathcal{E} \vdash (S, E_1 \cup E_2)$;
Rule E6 : $\mathcal{E} \vdash (S, E)$ and h is a homomorphism from S in S_1 implies $\mathcal{E} \vdash (S_1, h(E))$;
Rule E7 : $\mathcal{E} \vdash (S, \{(t = t')\})$ and $\mathcal{E} \vdash (\phi_{t \leftrightarrow t'}(S), E)$ implies $\mathcal{E} \vdash (S, E)$;
Rule E8 : $\mathcal{E} \vdash (S, \{(a = b)\})$ for $a, b \in \mathcal{U}_S$ and $a \neq b$ implies $\mathcal{E} \vdash (S, E)$ for all possible E.

Theorem 8 ([1]). *Let $\mathcal{E}$ be a finite set of EGCs. The set of deductive rules Rule E0-E8 for EGCs is sound, independent and complete for grounded graphs.*

There is no need for deduction rules for functional constraints, since they can be expressed as equality generating constraints.

Theorem 9. *Let $\mathcal{T}$ be a finite set of TGCs. The following set of deductive rules for TGCs is sound, independent and complete for grounded graphs:*

Rule T0 : $(S, S') \in \mathcal{T}$ implies $\mathcal{T} \vdash (S, S')$;
Rule T1 : $\mathcal{T} \vdash (S, S)$ for all S;
Rule T2 : $\mathcal{T} \vdash (S_0, S_1)$ and $\mathcal{T} \vdash (S_0, S_2)$ implies $\mathcal{T} \vdash (S_0, S_1 \cup S_2)$;
Rule T3 : $\mathcal{T} \vdash (S_0, S_1)$ and $\mathcal{T} \vdash (S_0 \cup S_1, S_2)$ implies $\mathcal{T} \vdash (S_0, S_2)$;
Rule T4 : h is a homomorphism from S in S_1 and $\mathcal{T} \vdash (S, S')$ implies $\mathcal{T} \vdash (S_1, h(S'))$.

Proof. Rules T0-T3 are clearly sound. For Rule T4, let $\mathcal{G} \models (S, S')$ and let e be an embedding of S_1 in $\mathcal{G}$. Hence $e \circ h$ is an embedding of S in $\mathcal{G}$, inducing that $e \circ h$ is also an embedding of S' in $\mathcal{G}$, concluding that e is an embedding of $h(S')$ in $\mathcal{G}$.
Rules T0-T4 are complete. Indeed, referring to the proof of Theorem 4, we can deduce from Rules T0-T4 that $\mathcal{T} \vdash (S, \overline{S})$. Furthermore we have that $\mathcal{T} \vdash (S_0, S_1)$ and $S_2 \subseteq S_1$ implies $\mathcal{T} \vdash (S_0, S_2)$, as a consequence of Rule T4.
Rules T0-T4 are independent. We show this by giving for each $0 \leq i \leq 4$ a set $\mathcal{T}_i$ and a TGC TG_i that can be deduced from $\mathcal{T}_i$ but for which we need Rule Ti:

- $\mathcal{T}_0 = \{(\{(a,b,c)\}, \{(d,e,f)\})\}$ and $TG_0 = (\{(a,b,c)\}, \{(d,e,f)\})$;
- $\mathcal{T}_1 = \{(\{(a,b,c)\}, \{(d,e,f)\})\}$ and $TG_1 = (\{(g,h,i)\}, \emptyset)$;
- $\mathcal{T}_2 = \{(\{(a,b,c)\}, \{(d,e,f)\}), (\{(a,b,c)\}, \{(g,h,i)\})\}$ and $TG_2 = (\{(a,b,c)\}, \{(d,e,f), (g,h,i)\})$;
- $\mathcal{T}_3 = \{(\{(a,b,c)\}, \{(d,e,f)\}), (\{(a,b,c), (d,e,f)\}, \{(g,h,i)\})\}$ and $TG_3 = (\{(a,b,c)\}, \{(g,h,i)\})$;
- $\mathcal{T}_4 = \{(\{(a,a,\$x)\}, \{(a,\$x,a)\})\}$ and $TG_4 = (\{(a,a,\$y)\}, \{(a,\$y,a)\})$;

□

Lemma 2. *1. $S_1 \subseteq S_0$ implies $\mathcal{T} \vdash (S_0, S_1)$;*
2. $\mathcal{T} \vdash (S_0, S_1)$ and $S_0 \subseteq S_2$ implies $\mathcal{T} \vdash (S_2, S_1)$;
3. $\mathcal{T} \vdash (S_0, S_1)$ and $S_2 \subseteq S_1$ implies $\mathcal{T} \vdash (S_0, S_2)$.

Proof. 1. follows from Rule T4; 2. follows from Rule T4; 3. follows from 1., 2. and Rule T3.□

Theorem 10. *Let $\mathcal{FB}$ be a finite set of FBCs. The following set of deductive rules for FBCs is sound, independent and complete for grounded graphs:*

Rule FB0 : $\mathcal{FB} \vdash (S)$, for every $(S) \in \mathcal{FB}$;
Rule FB1 : $\mathcal{FB} \vdash (S)$ and and h is a homomorphism from S in S_1 implies $\mathcal{FB} \vdash (S_1)$.

Proof. Sound: FB1. Let $\mathcal{G} \models FBC$ for all $FBC \in \mathcal{FB}$. So $\mathcal{G} \models (S)$ and there is no embedding e of S in $\mathcal{G}$. If there would be an embedding e_1 of S_1 in $\mathcal{G}$ then $e_1 \circ h$ would be an embedding of S in $\mathcal{G}$. So $\mathcal{G} \models (S_1)$.
Clearly FB0 and FB1 are independent.
Complete: Let $\mathcal{FB} \nvdash (S)$ and consider the graph $\overline{S}$ of Theorem 5. There is clearly an embedding of S in the graph $\overline{S}$. If there would be an $(S') \in \mathcal{FB}$ with an embedding e of S' in $\overline{S}$, there would be an homomorphism from S' in S, implying that $\mathcal{FB} \vdash (S)$. Hence the graph $\overline{S}$ is a counterexample proving that (S) is not a logical grounded consequence of $\mathcal{FB}$. □

Theorem 11. *Let $\mathcal{TE}$ be a finite set of TGCs and EGCs. The following set of deductive rules for TGCs and EGCs is sound, independent and complete for grounded graphs:*

Rules T0-T4[5]*;*
Rules E0-E8[6]*;*
Rule TE1 : $\mathcal{TE} \vdash (S, S')$ *and* $\mathcal{TE} \vdash (S', E)$ *implies* $\mathcal{TE} \vdash (S, E)$*;*
Rule TE2 : $\mathcal{TE} \vdash (S, \{(t = t')\})$ *and* $\mathcal{TE} \vdash (\phi_{t \hookleftarrow t'}(S), S')$ *implies* $\mathcal{TE} \vdash (S, S')$*;*
Rule TE3 : $\mathcal{TE} \vdash (S, \{(a = b)\})$ *with* $a, b \in \mathcal{U}_S$ *and* $a \neq b$ *implies* $\mathcal{TE} \vdash (S, S')$ *for all* S' *with* $V_{S'} \subseteq V_S$*. (Proof, Cfr. Appendix.)*

Theorem 12. *Let* $\mathcal{TFB}$ *be a finite set of TGCs and FBCs. The following set of deductive rules for TGCs and FBCs is sound, independent and complete for grounded graphs:*

Rules T0-T4[7]*;*
Rules FB0-FB1[8]*;*
Rule TFB0 : $\mathcal{TFB} \vdash (S, S')$ *and* $\mathcal{TFB} \vdash (S')$ *implies* $\mathcal{FB} \vdash (S)$*;*
Rule TFB1 : $\mathcal{TFB} \vdash (S)$ *implies* $\mathcal{TFB} \vdash (S, S')$ *for every* S' *with* $V_{S'} \subseteq V_S$*. (Proof, Cfr. Appendix.)*

3.4 Satisfiability of Constraints for Graphs

Two groundings θ_1 and θ_2 of an RDF-graph $\mathcal{G}$ are called *isomorphic* iff $\theta_1(a) = \theta_1(b) \Leftrightarrow \theta_2(a) = \theta_2(b)$ for every $a, b \in \mathcal{B}_\mathcal{G} \cup \mathcal{U}_\mathcal{G}$.

Lemma 3. *Let* θ_1 *and* θ_2 *be two isomorphic groundings of an RDF-graph* $\mathcal{G}$*.*

1. $\mathcal{G}_{\theta_1} \models (S, E)$ *iff* $\mathcal{G}_{\theta_2} \models (S, E)$*, with* (S, E) *an EGC;*
2. $\mathcal{G}_{\theta_1} \models (S, S')$ *iff* $\mathcal{G}_{\theta_2} \models (S, S')$*, with* (S, S') *an TGC;*
3. $\mathcal{G}_{\theta_1} \models (S)$ *iff* $\mathcal{G}_{\theta_2} \models (S)$*, with* (S) *an FBC.*

Since there are only a finite number of non-isomorphic groundings, $\mathfrak{G}_\mathcal{G}$ is finite for each graph $\mathcal{G}$ and hence we have:

Theorem 13. *It is decidable whether a graph* $\mathcal{G}$ *satisfies an EGC* (S, E)*, an FC* $(S, L \to R)$*, a TGC* (S, S') *or a FBC* (S)*.*

Let $\mathcal{SC}$ be a finite set of constraints and $\mathcal{C}$ be a constraint, we say that $\mathcal{C}$ is a *logical consequence* of $\mathcal{SC}$ iff for all graphs $\mathcal{G}$ holds

$$(\forall \mathcal{C}' \in \mathcal{SC}(\mathcal{G} \models \mathcal{C}')) \Rightarrow \mathcal{G} \models \mathcal{C}$$

The next theorem is straightforward consequence of Subsection 3.3:

Theorem 14. *Given a finite set* $\mathcal{SC}$ *of EGCs (resp. FCs, TGCs, FBCs). It is decidable whether an EGC (resp. FC, TGC, FBC)* $\mathcal{C}$ *is a logical consequence of* $\mathcal{SC}$*.*

[5] Substitute $\mathcal{T}$ by $\mathcal{TE}$
[6] Substitute $\mathcal{E}$ by $\mathcal{TE}$
[7] Substitute $\mathcal{T}$ by $\mathcal{TFB}$
[8] Substitute $\mathcal{FB}$ by $\mathcal{TFB}$

Let us consider now *sound, independent and complete sets of deductive rules for graphs*. Clearly :

$$\forall \mathcal{G}_G((\forall \mathcal{C}' \in \mathcal{SC}(\mathcal{G}_G \models \mathcal{C}')) \Rightarrow \mathcal{G}_G \models \mathcal{C}) \Leftrightarrow \forall \mathcal{G}((\forall \mathcal{C}' \in \mathcal{SC}(\mathcal{G} \models \mathcal{C}')) \Rightarrow \mathcal{G} \models \mathcal{C})$$

where $\mathcal{G}_G$ are grounded graphs and $\mathcal{G}$ are graphs.

Theorem 15. *The set E0-E8 (resp. T0-T4, FB0-FB1) of deductive rules for EGCs (resp. TGCs, FBCs) is sound, independent and complete for graphs.*
The set E0-E8, T0-4, TE1-2 of deductive rules for EGCs and TGCs is sound, independent and complete for graphs.
The set FB0-1, T0-4, TFB0 of deductive rules for FBCs and TGCs is sound, independent and complete for graphs.

4 Constraints in RDF Schema

4.1 S-Constraints

An RDF Schema (abbreviated as RDFS) is an extensible knowledge representation language providing basic elements for the description of ontologies, called RDF vocabularies, intended to structure RDF-resources. The RDF-properties of RDFS that we will consider are *subproperty* (**sp**), *subclass* (**sc**), *typing* (**type**), *domain* (**dom**) and *range* (**range**). In the literature this fragment of RDFS is called ρrdf [6, 3].

Definition 8. *An RDFS triple is a triple containing* **sp**, **sc**, **type**, **dom** *or* **range**. *An RDFS-graph is an RDF-graph that includes at least one RDFS triple.*

We will now express the semantics of RDFS-properties using triple generating constraints. We call the following set of TGCs, denoted by $\mathcal{C}^S$, *S–Constraints*. Consider the RDF-subjects or RDF-objects (that by the way can also be RDF-properties) S_1, S_2, S_3, the RDF-properties P_1, P_2, P_3, the classes of objects or subjects (or types) C_1, C_2 and C_3.
Subproperty
The RDFS subproperty is denoted by **sp**. $(P_1, \mathbf{sp}, P_2)$ indicates that P_1 is a subproperty of P_2. The semantics of **sp** says that

- if P_1 is a subproperty of P_2 and P_2 is a subproperty of P_3 then P_1 is a subproperty of P_3; this is expressed by the triple generating constraint:$(\{(\$p_1, \mathbf{sp}, \$p_2), (\$p_2, \mathbf{sp}, \$p_3)\}, \{(\$p_1, \mathbf{sp}, \$p_3)\})$
- if S_1 has as property P_1 with value S_2 and P_1 is a subproperty of P_2 then S_1 has also as property P_2 with value S_2, formally: $(\{(\$s_1, \$p_1, \$s_2), (\$p_1, \mathbf{sp}, \$p_2)\}, \{(\$s_1, \$p_2, \$s_2)\})$
- every property between a subject and an object is a subproperty of itself, formally: $(\{(\$s_1, \$p, \$s_2)\}, \{(\$p, \mathbf{sp}, \$p)\})$
- if P_1 is a subproperty of P_2 then P_1 and P_2 are each a subproperty of itself, formally: $(\{(\$p_1, \mathbf{sp}, \$p_2)\}, \{(\$p_1, \mathbf{sp}, \$p_1), (\$p_2, \mathbf{sp}, \$p_2)\})$

Subclass
The RDFS subclass is denoted by **sc**. $(C_1, \mathbf{sc}, C_2)$ indicates that C_1 is a subclass of C_2. The semantics of **sc** says that

- if C_1 is a subclass of C_2 and C_2 is a subclass of C_3 then C_1 is a subclass of C_3, formally:$(\{(\$c_1, \mathbf{sc}, \$c_2), (\$c_2, \mathbf{sc}, \$c_3)\}, \{(\$c_1, \mathbf{sc}, \$c_3)\})$
- if C_1 is a subclass of C_2 then C_1 and C_2 are each a subclass of itself, formally:$(\{(\$c_1, \mathbf{sc}, \$c_2)\}, \{(\$c_1, \mathbf{sc}, \$c_1), (\$c_2, \mathbf{sc}, \$c_2)\})$
- Finally, classes are subclasses of themselves:$(\{(\$S_1, \mathbf{type}, \$C_1)\}, \{(\$C_1, \mathbf{sc}, \$C_1)\})$

Typing, Domain, Range
The RDFS typing is denoted by **type**, the domain of a type by **dom**, its range by **range**. $(S_1, \mathbf{type}, C_1)$ indicates that C_1 is a type of S_1. $(P_1, \mathbf{dom}, C_1)$ indicates that C_1 is the domain of P_1. $(P_1, \mathbf{range}, C_1)$ indicates that C_1 is the range of P_1. Their semantics says that

- if C_1 is a subclass of C_2 and C_1 is a type of S_1 than C_2 is also a type of S_1, formally:$(\{(\$c_1, \mathbf{sc}, \$c_2), (\$s_1, \mathbf{type}, \$c_1)\}, \{(\$s_1, \mathbf{type}, \$c_2)\})$
- if C_1 is the domain P_1, and S_1 has property P_1, then C_1 is a type of S_1, formally:$(\{(\$p_1, \mathbf{dom}, \$c_1), (\$s_1, \$p_1, \$s_2)\}, \{(\$s_1, \mathbf{type}, \$c_1)\})$
- if C_1 is the range of the P_1, and S_2 is a value of the property P_1, then C_1 is a type of S_1, formally:$(\{(\$p_1, \mathbf{range}, \$c_1), (\$s_1, \$p_1, \$s_2)\}, \{(\$s_2, \mathbf{type}, \$c_1)\})$
- if C_1 is the domain of P_1, P_2 is a subproperty of P_1 and S_1 has property P_2 with value S_2 then C_1 is a type of S_1: $(\{\$p_1, \mathbf{dom}, \$c_1), (\$p_2, \mathbf{sp}, \$p_1), (\$s_1, p_2, \$s_2)\}, \{(\$s_1, \mathbf{type}, \$c_1)\})$
- if C_1 is the range of P_1, P_2 is a subproperty of P_1 and S_1 has property P_2 with value S_2 then C_1 is a type of S_2: $(\{\$p_1, \mathbf{range}, \$c_1), (\$p_2, \mathbf{sp}, \$p_1), (\$s_1, \$p_2, \$s_2)\}, \{(\$s_2, \mathbf{type}, \$c_1)\})$
- **sp**, **sc**, **dom**, **range**, **type** are each a subproperty of itself, formally: $(\emptyset, \{(\mathbf{v}, \mathbf{sp}, \mathbf{v})\}), \mathbf{v} \in \{\mathbf{sp}, \mathbf{sc}, \mathbf{dom}, \mathbf{range}, \mathbf{type}\}$
- every property that has a domain or a range is a subproperty of itself, formally: $(\{(\$p, \mathbf{v}, \$y)\}, \{(\$p, \mathbf{sp}, \$p)\}), \mathbf{v} \in \{\mathbf{dom}, \mathbf{range}\}$
- every class that is the domain, the range or a type is a subclass of itself, formally: $(\{(\$x, \mathbf{v}, \$c)\}, \{(\$c, \mathbf{sc}, \$c)\}), \mathbf{v} \in \{\mathbf{dom}, \mathbf{range}, \mathbf{type}\}$

4.2 Constraint satisfaction in RDFS

In this section we discuss constraint satisfaction in RDFS. We motivate the need for additional semantic considerations for constraints in RDFS with a simple example.

Example 5. Consider the EGC $\mathcal{C} = (\{(a, \mathbf{sp}, \$x)\}, \{(\$x = b)\})$ and the following graph:

$$\mathcal{G} = \{(a, \mathbf{sp}, b), (b, \mathbf{sp}, c)\}.$$

Observe first that $\mathcal{G} \models \mathcal{C}$. Indeed for every embedding of $\{(a, \mathbf{sp}, \$x)\}$ in $\mathcal{G}$ we have that $\$x = b$. Under RDFS semantics, however, the constraint should not

be satisfied. The reason is that RDFS implies the existence of triples that are not in the graph. In the current example, triple $(a, \mathbf{sp}, c)$, for instance, while not part of $\mathcal{G}$, is induced by the semantics of RDFS but does not satisfy the EGC.

Let us call $\mathcal{G}^I$ the RDF-graph that is defined (or constructed) from an initial graph $\mathcal{G}$ together with all its logical consequences according to the semantics of RDFS as defined by the *S–Constraints* in Section 4.1. In this new scenario, we check satisfiability of constraints against $\mathcal{G}^I$ instead of $\mathcal{G}$.

Example 6. Let $\mathcal{G}$ be the graph of Example 5. The intentional graph $\mathcal{G}^I$ is

$$\mathcal{G}^I = \{(a, \mathbf{sp}, b)(b, \mathbf{sp}, c), (a, \mathbf{sp}, c), \\ (a, \mathbf{sp}, a), (b, \mathbf{sp}, b), (c, \mathbf{sp}, c)\}.$$

$\mathcal{G} \models \mathcal{C}$, but $\mathcal{G}^I \not\models \mathcal{C}$ as triples $(a, \mathbf{sp}, a)$ and $(a, \mathbf{sp}, c)$ violate the constraint.

We now formalize these notions.

Definition 9 (Intensional Graph). *Let $\mathcal{G}$ be a graph. The intentional graph of $\mathcal{G}$, denoted as $\mathcal{G}^I$, is defined as the smallest graph with*

- $\mathcal{G} \subseteq \mathcal{G}^I$;
- $\mathcal{G}^I$ *satisfies the S–Constraints.*

Definition 10. *Let $\mathcal{C}$ be a constraint. We say that the graph $\mathcal{G}$ RDFS-satisfies C, denoted as $\mathcal{G} \models_{RDFS} \mathcal{C}$ iff $\mathcal{G}^I \models \mathcal{C}$.*

Theorem 16. *Let $\mathcal{G}$ be and RDF-graph (without containing S-properties) and $\mathcal{C}$ a constraint such that $\mathcal{G} \models \mathcal{C}$. Then, $\mathcal{G} \models_{RDFS} \mathcal{C}$.*

Proof. By definition $\mathcal{G} \models_{RDFS} \mathcal{C}$ iff $\mathcal{G}^I \models \mathcal{C}$. But since $\mathcal{G}$ contains no RDFS properties $\mathcal{G} \equiv \mathcal{G}^I$. We then have as an immediate consequence that $\mathcal{G} \models_{RDFS} \mathcal{C}$.

Example 7. Consider the following RDFS-graph:

$$\mathcal{G} = \{(a, \mathbf{sp}, b)(b, \mathbf{sp}, \_c)\}.$$

It is the case that: $\mathcal{G} \not\models_{RDFS} (\{(a, \mathbf{sp}, \$\mathbf{x})\}, \{(\$\mathbf{x} = \mathbf{b}\})\}$; $\mathcal{G} \not\models_{RDFS} (\{(a, \mathbf{sp}, \$\mathbf{x})\}, \{\mathbf{a}\} \rightarrow \{\$\mathbf{x}\})$; $\mathcal{G} \not\models_{RDFS} (\{(b, \mathbf{sp}, \$\mathbf{x})\}, \{(\mathbf{a}, \mathbf{sp}, \$\mathbf{x})\})$; $\mathcal{G} \not\models_{RDFS} (\{(a, \mathbf{sp}, \mathbf{c})\})$.

With the semantics machinery in order, we now turn to the problem of deciding constraint satisfaction in RDFS-graphs.

Lemma 4. *Let $\mathcal{G}$ be an RDFS-graph. Then $\mathcal{G}^I$ is computable.*

Theorem 17. *Let $\mathcal{G}$ be and RDFS-graph and $\mathcal{SC}$ a finite set of constraints. It is decidable checking whether $\forall \mathcal{C} \in \mathcal{SC}$ $(\mathcal{G} \models_{RDFS} \mathcal{C})$.*

We now deal with the use of some of the techniques and results of Section 3 in the context of RDFS-graphs.

Lemma 5. *Let $\mathcal{G}$ be an RDFS-graph and $\mathcal{SC}$ a finite set of constraints. It holds that $\forall \mathcal{C} \in \mathcal{SC}$, $\forall \mathcal{C}' \in \mathcal{C}^S$ $\mathcal{G}^I \models \mathcal{C} \cup \mathcal{C}'$ iff $\mathcal{G} \models_{RDFS} \mathcal{C}$.*

Theorem 18. *Let $\mathcal{C}$ be a finite set of EGC (resp. TGC, FC or FB) and $\mathcal{G}$ an RDFS-graph such that $\mathcal{G} \models_{RDFS} \mathcal{C}$. Let $\mathcal{C}'$ be an EGC constraint (resp. TGC, FC or FB). Then it holds that $(\mathcal{C} \cup \mathcal{C}^S) \vdash \mathcal{C}'$ iff $\mathcal{G} \models_{RDFS} \mathcal{C}'$.*

Proof. Assume $(\mathcal{C} \cup \mathcal{C}^S) \vdash \mathcal{C}'$. From Lemma 5 and $\mathcal{G} \models_{RDFS} \mathcal{C}$, we know that $\mathcal{G}^I \models \mathcal{C} \cup \mathcal{C}^S$. From Theorem [14] we have, in addition, that $\mathcal{G} \models \mathcal{C}'$. As a consequence, we have that $\mathcal{G} \models_{RDFS} \mathcal{C}'$. The other direction is trivial. □

5 Related Work and Conclusions

5.1 Related Work

RDF is a W3C recommendation and the reader interested in a thorough treatment of the language may find useful the official W3C document available at [9]. In [4], Gutierrez et al. provide formal foundations for RDF-graphs and propose a query language coupled with a study of its computational properties. Theoretical aspects of SPARQL, the de-facto query language for RDF, are investigated in [7, 2].

Constraints were first introduced into an RDF framework by Lausen at al. [5]. The authors' main motivation for adding constraints was the need to migrate a relational database with primary and foreign keys into an RDF-graph, without losing semantic information. The authors show how constraints such as keys and foreign keys can be encoded into the resulting RDF-graph by means of introducing additional nodes in an extended RDF vocabulary. FCs are defined in terms of properties of RDF interpretations over objects of the same type (or class). In [10], Smith et al. exploit the knowledge encoded in constraints (tuple generating and equality generating dependencies) in order to speed up the processing time for solving SPARQL queries. In [1], the authors study EGCs and FCs from a computational point of view. In particular, chasing algorithms that compute all the logical consequences of a given initial set of constraints are proposed.

5.2 Conclusions

We have studied constraints in RDFS-graphs in the presence of blank nodes. Included in this paper are decidability results for entailment of different types of constraints, sound and complete deductive rules for several combination of constraints, and the analysis of constraint satisfaction in RDFS-graphs. Future lines of research include the development of efficient methods for detecting constraints violation in RDFS-graphs and inconsistency handling.

References

1. Waseem Akhtar, Alvaro Cortés-Calabuig, and Jan Paredaens. Constraints in RDF. In *Semantics in Data and Knowledge Bases - 4th International Workshops, SDKB 2010, Bordeaux, France, July 5, 2010, Revised Selected Papers*, pages 23–39, 2010.
2. Renzo Angles and Claudio Gutierrez. The expressive power of SPARQL. In *Proceedings of the International Semantic Web Conference (ISWC08)*, pages 114–129, 2008.
3. Marcelo Arenas, Claudio Gutierrez, and Jorge Pérez. Foundations of RDF databases. In *Reasoning Web*, pages 158–204, 2009.
4. Claudio Gutierrez, Carlos A. Hurtado, and Alberto O. Mendelzon. Foundations of semantic web databases. In *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS04), June 14-16, 2004, Paris, France*, pages 95–106, 2004.
5. Georg Lausen, Michael Meier, and Michael Schmidt. Sparqling constraints for RDF. In *Proceedings 11th International Conference on Extending Database Technology (EDBT08), Nantes, France, March 25-29*, pages 499–509, 2008.
6. Sergio Muñoz, Jorge Pérez, and Claudio Gutiérrez. Minimal deductive systems for RDF. In *The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007, Proceedings*, pages 53–67, 2007.
7. Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3), 2009.
8. RDF primer. http://www.w3.org/TR/rdf-primer/, 2004.
9. RDF semantics. http://www.w3.org/TR/rdf-mt/, 2004.
10. Michael Schmidt, Michael Meier, and Georg Lausen. Foundations of SPARQL query optimization. In *Proceedings International Conference Database Theory (ICDT10), 13th International Conference*, 2010.

6 Appendix

Proof of Theorem 11. The soundness of the rules is quite obvious.
Independence: The rules E0-E8 are independent, since they are independent in Theorem 8 (suppose there are no TGCs in $\mathcal{TE}$). The rules T0-T4 are independent, since they are independent in Theorem 9 (suppose there are no EGCs in $\mathcal{TE}$).
We need Rule TE1 to derive
$\{(\{(a,a,\$x)\},\{(b,b,\$x)\}),(\{(b,b,\$x)\},\{(\$x=a)\})\} \vdash (\{(a,a,x)\},\{(\$x=a)\})$.
We need Rule TE2 to derive
$\{(\{(b,b,a)\},\{(b,b,b)\}),(\{(b,b,\$x)\},\{(\$x=a)\})\} \vdash (\{(b,b,b)\},\{(\$x=b)\})$.
We need Rule TE3 to derive
$\{(\{(b,b,a)\},\{(a=b)\})\} \vdash (\{(b,b,a)\},\{(c,c,c)\})$.
Completeness: Without loss of generality we suppose that all EGCs have a singleton set of equalities and all TGCs have a singleton second set of triples. Let $\mathcal{TE} = \{(S_i,\{(t_i = t'_i)\}) \mid 1 \leq i \leq n\} \cup \{(S_i,\{s_i\}) \mid n+1 \leq i \leq q\}$, $\mathcal{TE} \nvdash (S,\{(t=t')\})$ and $\mathcal{TE} \nvdash (S,\{s'\})$. We will construct a graph $\overline{G}$ that satisfies all the constraints of $\mathcal{TE}$ but does not satisfy $(S,\{(t=t')\})$ nor $(S,\{s'\})$, inducing that $(S,\{(t=t')\})$ nor $(S,\{s'\})$ are a logical grounded consequences of $\mathcal{TE}$.
We first define a finite sequence of sets $\Sigma_0,\ldots,\Sigma_p$ and a finite sequence of functions $\Phi_0,\ldots,\Phi_p$ on $V_S \cup \mathcal{U}$:

1. $\Sigma_0 = S$ and Φ_0 is the identity on $V_S \cup \mathcal{U}$;
2. $\forall m, 0 \leq m < p$
   (2.1) $\exists j_m, 1 \leq j_m \leq n$ and $\exists\ h_m$ a homomorphism from S_{j_m} in Σ_m (a) with $h_m(t_{j_m}) \neq h_m(t'_{j_m})$ and $\{h_m(t_{j_m}), h_m(t'_{j_m})\} \not\subseteq \mathcal{U}$. If $h_m(t_{j_m}) \in V$ and $h_m(t'_{j_m}) \in \mathcal{U}$ interchange t_{j_m} and t'_{j_m}. Define $\Sigma_{m+1} = \phi_{h_m(t_{j_m}) \leftarrow h_m(t'_{j_m})}(\Sigma_m)$ (b) and $\Phi_{m+1} = \phi_{h_m(t_{j_m}) \leftarrow h_m(t'_{j_m})} \circ \Phi_m$; or
   (2.2) $\exists j_m, n+1 \leq j_m \leq q$ and $\exists\ h_m$ a homomorphism from S_{j_m} in Σ_m (a) with $h_m(s_{j_m}) \notin \Sigma_m$. Define $\Sigma_{m+1} = \Sigma_m \cup \{h_m(s_{j_m})\}$ (c) and $\Phi_{m+1} = \Phi_m$.
3. (3.1) $\exists j_p, 1 \leq j_p \leq n$ and $\exists\ h_p$ a homomorphism from S_{j_p} in Σ_p with $h_p(t_{j_p}) \neq h_p(t'_{j_p})$ and $\{h_p(t_{j_p}), h_p(t'_{j_p})\} \subseteq \mathcal{U}$ or
   (3.2) $\forall\ h_p$ a homomorphism from S_{j_p} in Σ_p holds that $\forall j_p, 1 \leq j_p \leq n$ holds that $h_p(t_{j_p}) = h_p(t'_{j_p})$ and $\forall j_p, n+1 \leq j_p \leq q$ holds that $h_p(s_{j_p}) \in \Sigma_p$.

Clearly for $\forall m, 0 \leq m \leq p$,
$\Phi_m = \phi_{h_{m-1}(t_{j_{m-1}}) \leftarrow h_{m-1}(t'_{j_{m-1}})} \circ \ldots \circ \phi_{h_i(t_{j_i}) \leftarrow h_i(t'_{j_i})} \circ \ldots \circ \phi_{h_0(t_{j_0}) \leftarrow h_0(t'_{j_0})}$ and Φ_m is a homomorphism from S in Σ_m.
We now prove that

- $\forall m, 0 \leq m \leq p$ and $1 \leq j_m \leq n$ holds that $\mathcal{TE} \vdash (S,\{(h_m(t_{j_m}) = h_m(t'_{j_m}))\})$ (d);
- $\forall m, 0 \leq m \leq p$ and $n+1 \leq j_m \leq q$ holds that $\mathcal{TE} \vdash (S,\{h_m(s_{j_m})\})$ (e).

We actually will prove by downward induction on k:
$\forall m, 0 \leq m \leq p$ and $\forall k, 0 \leq k \leq m$ holds:

- $\mathcal{TE} \vdash (\Sigma_k, \{(h_m(t_{j_m}) = h_m(t'_{j_m}))\})$, if $1 \leq j_m \leq n$ (f);
- $\mathcal{TE} \vdash (\Sigma_k, \{h_m(s_{j_m})\})$, if $n+1 \leq j_m \leq q$ (g).

Since $\mathcal{TE} \vdash (S_{j_m}, \{(t_{j_m} = t'_{j_m})\})$, if $1 \leq j_m \leq n$ and $\mathcal{TE} \vdash (S_{j_m}, \{s_{j_m}\})$, if $n+1 \leq j_m \leq q$ and h_m is a homomorphism from S_{j_m} in Σ_m, by (a) we have that
$\mathcal{TE} \vdash (\Sigma_m, \{(h_m(t_{j_m}) = h_m(t'_{j_m}))\})$ (h) if $1 \leq j_m \leq n$ by Rule E6 and
$\mathcal{TE} \vdash (\Sigma_m, \{h_m(s_{j_m})\})$ (i) if $n+1 \leq j_m \leq q$ by Rule T4.
Hence (f) and (g) hold for $k = m$.
By downward induction on k, if $1 \leq j_m \leq n$, using Rule E7,
$\mathcal{TE} \vdash (\Sigma_{k-1}, \{(h_{k-1}(t_{j_{k-1}}) = h_{k-1}(t'_{j_{k-1}}))\})$ by (h) and
$\mathcal{TE} \vdash (\phi_{h_{k-1}(t_{j_{k-1}}) \leftarrow h_{k-1}(t'_{j_{k-1}})}(\Sigma_{k-1}), \{(h_m(t_{j_m}) = h_m(t'_{j_m}))\})$ by (b) and by induction induce
$\mathcal{TE} \vdash (\Sigma_{k-1}, \{(h_m(t_{j_m}) = h_m(t'_{j_m}))\})$.
and if $n+1 \leq j_m \leq q$, using Rule TE1,
$\mathcal{TE} \vdash (\Sigma_{k-1}, \Sigma_{k-1} \cup \{h_{k-1}(s_{j_{k-1}})\})$ by (i) and
$\mathcal{TE} \vdash (\Sigma_k, \{(h_m(t_{j_m}) = h_m(t'_{j_m}))\})$ by (c) and by induction induce
$\mathcal{TE} \vdash (\Sigma_{k-1}, \{(h_m(t_{j_m}) = h_m(t'_{j_m}))\})$, proving (f). By downward induction on k, if $1 \leq j_m \leq n$, using Rule TE2,
$\mathcal{TE} \vdash (\Sigma_{k-1}, \{(h_{k-1}(t_{j_{k-1}}) = h_{k-1}(t'_{j_{k-1}}))\})$ by (h) and
$\mathcal{TE} \vdash (\Sigma_k, \{h_m(s_{j_m})\})$ by (c) and by induction induce
$\mathcal{TE} \vdash (\Sigma_{k-1}, \{(h_m(s_{j_m})\})$.
and if $n+1 \leq j_m \leq q$, using Rule T3,
$\mathcal{TE} \vdash (\Sigma_{k-1}, h_{k-1}(s_{j_{k-1}}))$ by (i) and
$\mathcal{TE} \vdash (\Sigma_k, \{h_m(s_{j_m})\})$ by (c) and by induction induce
$\mathcal{TE} \vdash (\Sigma_{k-1}, \{(h_m(s_{j_m})\})$, proving (g).
Taking $k = 0$ we have proved (d) and (e).
In case (3.1) we have that $\mathcal{TE} \vdash (S, \{(h_p(t_{j_p}) = h_p(t'_{j_p}))\})$, $h_p(t_{j_p}) \neq h_p(t'_{j_p})$ and $\{h_p(t_{j_p}), h_p(t'_{j_p})\} \subseteq \mathcal{U}$, so for all t, t' we have by Rule E8 $\mathcal{TE} \vdash (S, \{(t = t')\})$ and for all S' with $V_{S'} \subseteq V_S$ we have $\mathcal{TE} \vdash (S, S')$ by Rule TE3 .
In case (3.2) we take now $\overline{G}$ to be the graph obtained by substituting each variable $\$x$ in Σ_p by a new URI x. Clearly, by (3.2) we know that $\overline{G}$ satisfies all the EGCs and TGCs of $\mathcal{TE}$.
Suppose, by contradiction, that $\overline{G}$ satisfies $(S, \{(t = t')\})$, hence for every embedding e of S in $\overline{G}$ holds that $e(t) = e(t')$. Since Φ_p is such an embedding we have that $\Phi_p(t) = \Phi_p(t')$.
$\Phi_p = \phi_{h_{p-1}(t_{j_{p-1}}) \leftarrow h_{p-1}(t'_{j_{p-1}})} \circ \ldots \circ \phi_{h_i(t_{j_i}) \leftarrow h_i(t'_{j_i})} \circ \ldots \circ \phi_{h_0(t_{j_0}) \leftarrow h_0(t'_{j_0})}$.
This means that there is a sequence $t = \tau_0, \ldots, \tau_k, \ldots, \tau_m = t'$ with for $\forall k, 0 \leq k < m$:
$\exists i, 0 \leq i < p$:

- $\tau_k = h_i(t_{j_i})$ and $\tau_{k+1} = h_i(t'_{j_i})$ or
- $\tau_k = h_i(t'_{j_i})$ and $\tau_{k+1} = h_i(t_{j_i})$.

Hence $\mathcal{TE} \vdash (S, \{(\tau_k = \tau_{k+1})\})$, for all $0 \leq k < m$, and hence $\mathcal{TE} \vdash (S, \{(t = t')\})$, which is a contradiction.

Suppose furthermore, by contradiction, that $\overline{G}$ satisfies $(S, \{s'\})$, hence for every embedding e of S in $\overline{G}$ holds that $e(s') \in \overline{G}$. Since Φ_p is such an embedding we have that $\Phi_p(s') \in \overline{G}$. But $\overline{G} = \Sigma_p = S \bigcup_{1 \leq m \leq p, n+1 \leq j_m \leq q} \{h_m(s_{j_m})\}$, which implies that $\mathcal{TE} \vdash (S, \{s'\})$, which is a contradiction. □

Proof of Theorem 12. The soundness of the rules is straightforward. We need Rule TFB0 for $\{(\{(a,a,a)\}, \{(b,b,b)\}), (\{(b,b,b)\})\}$ implies $(\{(a,a,a)\})$. We need Rule TFB1 for $\{(\{(a,a,a)\})\}$ implies $(\{(a,a,a),(b,b,b)\})$.

Completeness: Let $\mathcal{TFB} = \{(S_i, S'_i) \mid 1 \leq i \leq n\} \cup \{(S_i) \mid n+1 \leq i \leq q\}$, $\mathcal{TFB} \nvdash (S)$ and $\mathcal{TFB} \nvdash (S, S')$. We will construct a graph $\overline{G}$ that satisfies all the constraints of $\mathcal{TFB}$ but does not satisfy (S) nor (S, S'), inducing that (S) nor (S, S') are a logical grounded consequences of $\mathcal{TFB}$.

Clearly we have that $\mathcal{TFB} \vdash (S, S')$ and $S_1 \subseteq S'$ then $\mathcal{TFB} \vdash (S, S_1)$ as a consequence of Rules ....

Let us define a finite sequence of sets $\Sigma_0, \ldots, \Sigma_p$:

1. $\Sigma_0 = S$;
2. $\forall m, 0 \leq m < p \; \exists j_m, 1 \leq j_m \leq n$ and $\exists\, h_m$ a homomorphism from S_{j_m} in Σ_m with $h_m(S'_{j_m}) \not\subseteq \Sigma_m$ (a); Define $\Sigma_{m+1} = \Sigma_m \cup h_m(S'_{j_m})$ (b);
3. $\forall j_p, 1 \leq j_p \leq n \; \forall h_p$ a homomorphism from S_{j_p} in Σ_p with $h_p(S'_{j_p}) \subseteq \Sigma_p$ (c).

We prove by induction that for $\forall m, 0 \leq m \leq p$ holds:

- if $S' \subseteq \Sigma_m$ then $\mathcal{TFB} \vdash (S, S')$ (d).
- if $\exists i, n+1 \leq i \leq q \; \exists h$, a homomorphism from S_i in Σ_m then $\mathcal{TFB} \vdash (S)$ (e);
- if $\exists i, n+1 \leq i \leq q \; \exists h$, a homomorphism from S_i in Σ_m then $\mathcal{TFB} \vdash (S, S')$, for all S' with $V_{S'} \subseteq V_S$ (f).

Clearly for $m = 0$ (d) holds because of Lemma 2, (e) holds because of Rule FB0 and FB1, (f) holds because of (e) and Rule TFB1.

By induction on m (d) holds because of Rules T0-T4, (e) holds because of Rule FB0 and FB1, (f) holds because of (e) and Rule TFB1.

Let ξ be the function that transforms each variable in Σ_p into a new URI, and let $\xi(\Sigma_p) = \mathcal{G}_S$.

We have to prove that the graph $\mathcal{G}_S$ satisfies all the constraints of $\mathcal{TFB}$ but does not satisfy (S) nor (S, S'):

- for $\forall i, 1 \leq i \leq n$ holds that $\mathcal{G}_S \models (S_i, S'_i)$, because of 3. above;
- for $\forall i, n+1 \leq i \leq q$ holds that $\mathcal{G}_S \models (S_i)$. Indeed, let $\exists i, n+1 \leq i \leq q$ with $\mathcal{G}_S \not\models (S_i)$; then there is an embedding e of S_i in $\mathcal{G}_S$; $\xi^{-1} \circ e$ is a homomorphism from S_i in Σ_p; hence by Rule FB1 $\mathcal{TFB} \vdash (\Sigma_p)$ and since $\mathcal{TFB} \vdash (S, \Sigma_p)$ we obtain $\mathcal{TFB} \vdash (S)$ by Rule TFB1, which is a contradiction.
- $\mathcal{G}_S \not\models (S, S')$; let on the contrary $\mathcal{G}_S \models (S, S')$, then every embedding of S in $\mathcal{G}_S$ is also an embedding of S' in $\mathcal{G}_S$; hence every homomorphism from S in Σ_p is also a homomorphism from S' in Σ_p and $S' \subseteq \Sigma_p$, but by (d) with $m = p$, we deduce $\mathcal{TFB} \vdash (S, S')$, a contradiction;
- $\mathcal{G}_S \not\models (S)$, since ξ is an embedding of S in $\mathcal{G}_S$.

□