

# TRIPLE: A Logic for Reasoning with Parameterized Views over Semi-Structured Data.\*

Stefan Decker<sup>2</sup>, Michael Sintek<sup>1,2</sup>, and Wolfgang Nejdl<sup>2</sup>

<sup>1</sup> DFKI GmbH Kaiserslautern

<sup>2</sup> Stanford University Database Group

**Abstract.** In this paper we investigate and formalize parameterized views for semi-structured data by defining syntax and a model theoretic semantics of TRIPLE, a logic for dealing with views. A unique feature of the language is that views can be used as parameters for other views. The definitions are used to derive a query and reasoning language based on deductive database technology. Applications include querying of data with respect to multiple semantics and information integration.

**Keywords:** Databases and the Semantic Web; Deductive databases and knowledge bases; Logic and databases; Semistructured, XML, and Web data

## 1 Motivation

Semi-structured data (cf., [19][21]) has been suggested as a means to information interchange and integration from heterogeneous information sources, and has been proven useful when used for mediator systems.

Views have been introduced in database systems to increase the flexibility by adapting the data to user or application needs, and some query languages for semi-structured data support view mechanisms, e.g., LORE [1]. *Parameterized Views* as described in [23] for relational databases, have been used for mediation [14].

In this paper we investigate and formalize parameterized views for semi-structured data by defining syntax and a model theoretic semantics of TRIPLE, a logic for dealing with views, inspired by F-Logic [11]. The definitions are used to derive a query and reasoning language based on deductive database technology, which supports parameterized views. Applications include querying of data with respect to multiple semantics and information integration. A unique feature of our approach is that views may be used as parameters for other views, therefore enabling composition of views.

### 1.1 Views, Semi-Structured Data, and the Semantic Web

An application area of parameterized views is the Semantic Web: in the Semantic Web [3] many different communities are publishing their formal data, and it is unlikely that established data models for representing this data will disappear. Examples of already established data models include UML, TopicMaps, RDF Schema, Entity Relationship Models, DAML+OIL, and more, highly specialized data models. Integrating data based on these different data models has proven to be an error prone and expensive task: different storage and query engines have to be combined into one system, and data has to be constantly translated from one representation to another. A first step to improve this situation is the use of a common data model as a representation formalism for all data involved. The database community has suggested semi-structured data as the underlying syntactic representation of data for data integration [19]. The Resource Description Framework (RDF) is a standard data model for representing and exchanging semi-structured data. Based on RDF many higher level data modeling languages have been developed, e.g., RDF vocabularies for UML, TopicMaps, or RDF Schema, DAML+OIL.<sup>1</sup>

However, using a common representation, such as RDF, does not solve the complete problem: although many query languages and inference engines for RDF exist (e.g., SiLRI [7], RQL [10], SQUISH<sup>2</sup>), none of them is capable of

\* This work was supported by the German Ministry for Education and Research, bmb+f (Grant: 01 IW 901, Project FRODO: A Framework for Distributed Organizational Memories) and the DARPA DAML Program, Project OntoAgents.

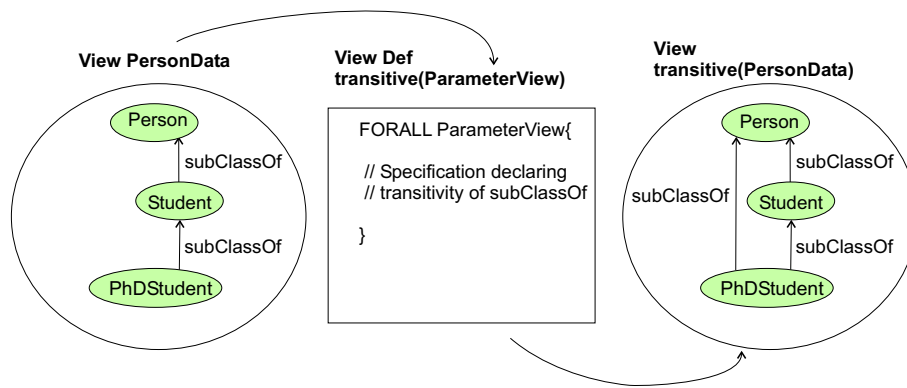
<sup>1</sup> See <http://www-db.stanford.edu/~melnik/rdf/uml/> for a representation of UML in RDF and [12] for a representation of TopicMaps in RDF.

<sup>2</sup> See <http://ilrt.org/discovery/2000/10/swsql/>

querying data with respect to multiple semantics as required by the different heterogeneous data modeling languages. E.g., when querying UML data the *Generalization* relation should be treated as a transitive relationship, as well as `rdfs:subClassOf` in RDF Schema. None of the cited query languages has the ability to query different data models with different kinds of semantics. Although some of them (RQL and SiLRI) have a built-in semantics for RDF Schema, this does not generalize to other data models. Since many different incompatible data models are already available this means existing query systems need to be extended with specialized reasoning engines for a particular data model.

## 1.2 Approach

*Parameterized views for semi-structured data* can be applied to remedy the situation. A view in our case is a subgraph of the overall data graph, similar to the notion of views as defined in [1]<sup>3</sup>. Enabling the use of views as parameters enables one to define the semantics of a particular data modeling language (e.g., RDF Schema) as a view over some source data.



**Fig. 1.** Views as parameters for other views in semi-structured data

Figure 1 illustrates the approach: the first view (the view with the ID `PersonData`) is passed as a parameter to the view definition `transitive` which declares the transitivity of the `subClassOf`-relation (which might be a part of the semantics declaration of a data modeling language, e.g., RDF Schema, possibly with a richer set of axioms). The view definition takes a parameter, a set of semi-structured data. The result of applying the view definition `transitive` to the view `PersonData` results in a new view, which respects the semantics of the data modeling language. The ID of the new view is constructed (by means of a *skolem function*) out of the ID of the data, `PersonData`, and the ID of the view definition, here resulting in the ID `transitive(PersonData)`. The example just shows the approach—depending on the concrete implementation it may never be necessary to fully materialize the resulting view.

## 1.3 Contributions

The contributions of this paper are:

- We introduce the notion of parameterized views over semi-structured data as first class citizens, meaning that parameterized views can be parameters of other views.

<sup>3</sup> Please note that we don't distinguish between intensionally and extensionally defined subgraphs. Although only intensionally defined subgraphs are real views, we slightly abuse the notion of views to include also extensionally defined subgraphs.

- We define TRIPLE, a logical language for reasoning with parameterized views over semi-structured data, and define a model theoretic semantics for the language (sections 2 and 3). As a concrete instance of semi-structured data we chose the Resource Description Framework (RDF) [13], therefore TRIPLE serves also as a formalization of RDF and an approach for rules and axioms for RDF.
- Based on the full language, we sketch a language subset, analog to Horn-logic, and describe a prototypical implementation for this subset.
- Finally, two examples illustrate the language and the approach.

## 2 Syntax

To ensure applicability the logic is specifically tailored towards RDF. This has implications with respect to the allowed character sets, namespace identifiers, and reification.

The syntax definition starts with defining the alphabet of TRIPLE, listing all symbols used in TRIPLE-expressions. Using the alphabet terms are defined, which in turn are used to define formulae. The alphabet contains several components: resource constructors are used to construct resource-ids, which (in RDF) are URIs.

### 2.1 Basic Definitions

**Definition 1 (Alphabet).** *The alphabet of a TRIPLE-language,  $\mathcal{L}$ , consists of*

- a set of resource constructors,  $\mathcal{F}$ <sup>4</sup>;
- an infinite set of variables,  $\mathcal{V}$ ;
- an infinite set of strings,  $\mathcal{H}$ <sup>5</sup>;
- a set  $\mathcal{E}$  of language identifiers<sup>6</sup> plus a single identifier ‘NULL’;
- auxiliary symbols, such as  $(, ), [, ], :, <, >, \rightarrow, ::, XML$ ; and
- usual logical connectives and quantifier;  $\wedge, \vee, \neg, \leftarrow, \rightarrow, \leftrightarrow, \forall, \exists$ .

Resource constructors (the elements of  $\mathcal{F}$ ) play the role of function symbols of TRIPLE. Each function symbol has an *arity*—a nonnegative integer that determines the number of arguments this symbol can take. Symbols of arity 0 are also called *constants*; symbols of arity  $\geq 1$  are used to construct larger terms out of simpler ones.

The following definitions lists the terms in TRIPLE.

**Definition 2 (Terms).** *Terms are defined inductively as follows:*

- A variable  $v \in \mathcal{V}$  is a term.
- A constant (a resource constructor with arity 0) is a term.
- For all  $s \in \mathcal{H}$ ,  $e \in \mathcal{E}$ ,  $s :: e$ ,  $XMLs$ , and  $XMLs :: e$  are terms, also called literals.
- If  $f$  is an  $n$ -ary function symbol ( $n > 0$ ) in  $\mathcal{F}$ , and  $t_1, \dots, t_n$  are terms, then  $f(t_1, \dots, t_n)$  is a term.
- If  $n$  and  $l$  are terms, then  $n : l$  is a term, also called a resource-id. The term  $n$  is called the namespace part of the resource-ids, the term  $l$  is called the localname part of the resource-id.
- If  $s$ ,  $p$ , and  $o$  are terms, then  $\langle s[p \rightarrow o] \rangle$  is a term, also called a statement-id. A statement-id is also a resource-id.
- These are all terms.

Using the definition of terms we are able to give the definition of the Herbrand Universe, which will be used later for the definition of Herbrand Models.

**Definition 3 (Herbrand Universe).** *The Herbrand Universe  $G(\mathcal{L})$  is the set of all ground (i.e., variable-free) terms, including resource-ids and statement-ids.*

<sup>4</sup> See appendix A for the exact definition of allowed resource constructors

<sup>5</sup> See appendix A for the exact definition of allowed character sequences

<sup>6</sup> The set  $\mathcal{E}$  contains the set of all language identifiers as defined in [8], *Tags for the Identification of Languages*, or its successor on the IETF Standards Track (e.g., ‘en’ for English).

Similar to id-terms in F-logic [11], resource-ids act as logical object-ids. Resource-ids with a “complex” structure, such as *child(stefan,birgit,I)*, usually arise when a new resource is constructed out of already existing resources.

Throughout this paper  $\mathcal{F}$  and  $G(\mathcal{L})$  will be used to denote the set of function symbols and ground terms, respectively.

RDF data is usually associated with a particular *view expression*, which identifies a view. A view identified by a view expression may be either extensionally defined (e.g., a subgraph of the overall data graph that originates from a particular source), or intensionally defined, computing a new subgraph by means of rules. View expressions define a set algebra on views<sup>7</sup>.

**Definition 4 (View Expressions).** View expressions are defined inductively as follows:

- A term is a view expression.
- Given two view expressions  $m_1$  and  $m_2$ , then
  - $m_1 \cup m_2$  (Union of two views),
  - $m_1 \cap m_2$  (Intersection of two views), and
  - $m_1 \setminus m_2$  (Set difference of two views)
 are also view expressions.
- These are all view expressions.

A language of TRIPLE (forthcoming called *T-language*, for brevity) consists of a set of formulae constructed out of the alphabet symbols. Formulae are built from simpler formulae by using the usual connectives  $\neg, \vee, \wedge$  and quantifiers  $\exists, \forall$ . The simplest kind of formulae are called *molecular TRIPLE-formulae* (abbr., *T-molecules* or just *molecules*).

Molecular formulae are defined using statements and model expressions:

**Definition 5 (Molecular Formulae).** Given a statement  $\langle s[p \rightarrow o] \rangle$  and a view expression  $m$ , the expression  $\langle s[p \rightarrow o] \rangle @m$  is called a molecule.

As a notional convention the angle brackets around the statement in a molecular formulae are usually omitted. General formulae are built from the molecular formulae by means of logical connectives and quantifiers:

**Definition 6 (Complex Formulae).** Formulae are defined as follows:

- A molecular formula is a formula.
- If  $\varphi$  and  $\psi$  are formulae, then so are  $\varphi \vee \psi, \varphi \wedge \psi, \varphi \rightarrow \psi, \varphi \leftarrow \psi, \varphi \leftrightarrow \psi, \neg\varphi$ .
- If  $X$  is a variable, and  $\varphi$  is a formula, then  $\forall X \varphi$  and  $\exists X \varphi$  are formulae.
- These are all formulae.

## 2.2 Notational Conventions

Some notional conventions have proven to be very useful when writing TRIPLE specifications.

**Complex Object Expressions** It is often convenient to combine several molecular formulae. A molecular expression  $s[p_1 \rightarrow o_1; p_2 \rightarrow o_2]@m$  is used as an abbreviation for  $s[p_1 \rightarrow o_1]@m \wedge s[p_2 \rightarrow o_2]@m$ .

A molecular expression  $s[p_1 \rightarrow o_1[p_2 \rightarrow o_2]]@m$  is used as an abbreviation for  $s[p_1 \rightarrow o_1]@m \wedge o_1[p_2 \rightarrow o_2]@m$ .

**View Blocks** *View blocks* are useful to group formulae. Instead of writing a reoccurring view expression several times in a set of formulae, a view block defines a default view expression for a set of formulae, which is used everywhere a view expression is omitted in a molecular formula. The example in figure 2 shows a view block, in which properties of a document are defined. The view expression `sem:documents` is the default view expression for every molecular formula in the view block, which is encapsulated by curly braces.

**Namespace Declarations** *Namespace declarations* mimic the namespace mechanism of XML [5] and especially their use in RDF. A namespace declaration possesses a left side and a right side. The left side defines an abbreviation, the right side is used as the substitution for the namespace abbreviation. The example in figure 2 defines three namespaces.

<sup>7</sup> Note that this notion of views is identical to the use of the term *RDF models* in the RDF literature.

```

rdf := 'http://www.w3.org/...rdf-syntax-ns#'.
dc := 'http://purl.org/dc/elements/1.0/'.
sem := 'http://triple.semanticweb.org/'.
@sem:documents {
  sem:d_01_01[
    dc:title → "TRIPLE";
    dc:creator → "Michael Sintek";
    dc:creator → "Stefan Decker";
    dc:subject → "RDF";
    dc:subject → "triples";... ].
}

```

**Fig. 2.** TRIPLE example using namespaces and blocks

**Blank Nodes** Blank nodes in RDF, as formalized in [9], are treated as existential variables. Since the full first-order language introduced here supports existential variables it is assumed that blank nodes are introduced with existential quantifiers.

### 3 Semantics

This section defines *semantic structures* for TRIPLE. Semantic structures are used to define fulfillment of formulae. In TRIPLE, semantic structures are called *T-structures*. T-structures define the usual constant universe, the set of all basic resources and statements, interpretation functions to construct resources and determining the truth value of molecular formulae.

**Definition 7 (T-structures).** Given a TRIPLE-language  $\mathcal{L}$ , a T-structure  $\mathcal{T}$  is a tuple  $\langle \mathcal{U}, \mathcal{R}, \mathcal{J}, \mathcal{S}, \mathcal{C}, \mathcal{M}, \mathcal{I} \rangle$ . The set  $\mathcal{U}$  is called the Constant Universe,  $\mathcal{R}$  is a set of Resources.  $\mathcal{J}$  is the set of 3-tuples, defined as  $\mathcal{H} \times \mathcal{E} \times \{\text{true}, \text{false}\}$  (as in definition 1),  $\mathcal{S}$  is constructed inductively as follows:

- $\mathcal{S}_0 = \mathcal{R} \times \mathcal{R} \times (\mathcal{R} \cup \mathcal{H})$
- $\mathcal{S}_{n+1} = (\mathcal{R} \cup \mathcal{S}_n) \times (\mathcal{R} \cup \mathcal{S}_n) \times (\mathcal{R} \cup \mathcal{H} \cup \mathcal{S}_n)$  for  $n \geq 0$ .

Then  $\mathcal{S}$  is defined as  $\bigcup_{i=0}^{\omega} \mathcal{S}_i$ .

$\mathcal{C}$  is a mapping from  $(\mathcal{U} \cup \mathcal{R} \cup \mathcal{S} \cup \mathcal{J}) \times (\mathcal{U} \cup \mathcal{R} \cup \mathcal{S} \cup \mathcal{J})$  to  $\mathcal{R}$ .  $\mathcal{M}$  is a mapping from  $\mathcal{R} \cup \mathcal{S}$  to  $2^{\mathcal{S}}$ .  $\mathcal{I}$  is an interpretation function defined as follows:

- Every constant  $c \in \mathcal{F}$  is mapped to an element  $c' \in \mathcal{U}$  from the universe (noted as  $\mathcal{I}(c) = c'$ );
- every  $n$ -ary ( $n > 0$ ) resource constructor is mapped to an  $n$ -ary function  $f' : \mathcal{U}^n \mapsto \mathcal{U}$  (noted as  $\mathcal{I}(f) = f'$ );
- every statement-id  $\langle s[p \rightarrow o] \rangle$  is mapped to a 3-tuple  $\langle \mathcal{I}(s), \mathcal{I}(p), \mathcal{I}(o) \rangle \in \mathcal{S}$  (noted as  $\mathcal{I}(\langle s[p \rightarrow] \rangle) = \langle \mathcal{I}(s), \mathcal{I}(p), \mathcal{I}(o) \rangle$ );
- every literal  $l$  is mapped to an element of  $\mathcal{J}$  as follows:
  - if  $l = XMLs :: e$  then  $\mathcal{I}(l) = \langle s, e, \text{true} \rangle$
  - if  $l = s :: e$  then  $\mathcal{I}(l) = \langle s, e, \text{false} \rangle$
  - if  $l = XMLs$  then  $\mathcal{I}(l) = \langle s, \text{null}, \text{true} \rangle$
  - if  $l = s$  then  $\mathcal{I}(l) = \langle s, \text{null}, \text{false} \rangle$

To be able to define truth values of formulae an adapted notion of the *variable assignment* notion is required.

**Definition 8 (Variable Assignment).** A variable assignment for a TRIPLE-language  $\mathcal{L}$  and a T-structure  $\mathcal{T}$  is a mapping  $B : \mathcal{V} \mapsto (\mathcal{U} \cup \mathcal{R} \cup \mathcal{S} \cup \mathcal{J})$ , which maps a variable to either an element of the universe, to a resource, or to a literal.

A variable assignment extends to terms in the usual way:

**Definition 9 (Term Denotation).** The denotation of a TRIPLE-term  $t$  with respect to a  $T$ -structure  $\mathcal{T}$  and a variable assignment  $B$  (noted as  $t^{\mathcal{T},B}$ ) is defined as follows:

- $v^{\mathcal{T},B} = B(v)$ , for all  $v \in \mathcal{V}$ ;
- $c^{\mathcal{T},B} = \mathcal{I}(c)$ , for all constants  $c \in \mathcal{F}$ ;
- $f(t_1, \dots, t_n)^{\mathcal{T},B} = \mathcal{I}(f)(t_1^{\mathcal{T},B}, \dots, t_n^{\mathcal{T},B})$ , for all terms  $t_1, \dots, t_n$  and all  $f \in \mathcal{F}$ ;
- $(n : l)^{\mathcal{T},B} = \mathcal{C}(n^{\mathcal{T},B} : l^{\mathcal{T},B})$  for all resource-ids  $n : l$ ;
- $\langle s[p \rightarrow o] \rangle^{\mathcal{T},B} = \mathcal{I}(\langle s^{\mathcal{T},B}[p^{\mathcal{T},B} \rightarrow o^{\mathcal{T},B}] \rangle)$  for all statement-ids  $\langle s[p \rightarrow o] \rangle$ ;
- $l^{\mathcal{T},B} = \mathcal{I}(l)$  for literals  $l$

The term denotation can be used to define the notion of molecular satisfaction.

**Definition 10 (Molecular Satisfaction).** The satisfaction of a molecular formula with respect to a  $T$ -structure  $\mathcal{T}$  and a variable assignment  $B$  is defined as follows:

- $\mathcal{T}, B \models \langle s[p \rightarrow o] \rangle @m$  iff  $\langle s[p \rightarrow o] \rangle^{\mathcal{T},B} \in \mathcal{M}(m^{\mathcal{T},B})$  if  $m$  is a resource-id;
- $\mathcal{T}, B \models \langle s[p \rightarrow o] \rangle @m_1 \cup m_2$  iff  $\mathcal{T}, B \models \langle s[p \rightarrow o] \rangle @m_1$  or  $\mathcal{T}, B \models \langle s[p \rightarrow o] \rangle @m_2$ ;
- $\mathcal{T}, B \models \langle s[p \rightarrow o] \rangle @m_1 \cap m_2$  iff  $\mathcal{T}, B \models \langle s[p \rightarrow o] \rangle @m_1$  and  $\mathcal{T}, B \models \langle s[p \rightarrow o] \rangle @m_2$ ;
- $\mathcal{T}, B \models \langle s[p \rightarrow o] \rangle @m_1 \setminus m_2$  iff  $\mathcal{T}, B \models \langle s[p \rightarrow o] \rangle @m_1$  and  $\mathcal{T}, B \not\models \langle s[p \rightarrow o] \rangle @m_2$

The meaning of complex formulae are defined in the standard way.

**Definition 11 (Formula Satisfaction).** Given formulae  $\varphi, \psi$  and a variable  $x$ , the satisfaction of a non-molecular formula with respect a  $T$ -structure  $\mathcal{T}$  a variable assignment  $B$  is defined as follows:

- $\mathcal{T}, B \models \varphi \wedge \psi$  iff  $\mathcal{T}, B \models \varphi$  and  $\mathcal{T}, B \models \psi$ ;
- $\mathcal{T}, B \models \varphi \vee \psi$  iff  $\mathcal{T}, B \models \varphi$  or  $\mathcal{T}, B \models \psi$ ;
- $\mathcal{T}, B \models \varphi \rightarrow \psi$  iff if  $\mathcal{T}, B \models \varphi$  then also  $\mathcal{T}, B \models \psi$ ;
- $\mathcal{T}, B \models \varphi \leftarrow \psi$  iff if  $\mathcal{T}, B \models \psi$  then also  $\mathcal{T}, B \models \varphi$ ;
- $\mathcal{T}, B \models \varphi \leftrightarrow \psi$  iff  $\mathcal{T}, B \models \varphi$  if and only if  $\mathcal{T}, B \models \psi$ ;
- $\mathcal{T}, B \models \neg\varphi$  iff  $\mathcal{T}, B \not\models \varphi$ ;
- $\mathcal{T}, B \models \forall x\varphi$  iff for all  $d \in \mathcal{U} \cup \mathcal{R} \cup \mathcal{S} \cup \mathcal{H}$   $\mathcal{T}, B_x^d \models \varphi$ , where

$$B_x^d(y) := \begin{cases} d & \text{if } x = y \\ B(y) & \text{otherwise} \end{cases}$$

- $\mathcal{T}, B \models \exists x\varphi$  iff there exists  $d \in \mathcal{U} \cup \mathcal{R} \cup \mathcal{S} \cup \mathcal{H}$   $\mathcal{T}, B_x^d \models \varphi$ , with  $B_x^d$  defined as above.

These definitions conclude the syntax and semantics definition of TRIPLE. The core definition is definition 10. This definition shows how molecular satisfaction is defined for graphs with respect to a certain view. The next section sketches how a query and rule language can be constructed using the above model theory.

## 4 HORN-TRIPLE : The Query and Rule Language

In the previous sections a language and model theory was developed for TRIPLE-logic, similar to usual full first-order logic. Several fragments of first-order logic have been identified with preferable computational properties. In the case of first-order logic, Horn-logic is used as a basis for Logic Programming and Deductive Databases, and is extended with various forms of non-monotonic negation to serve representation needs.

Analogously to usual predicate logic it is possible to define skolemization, and Skolem's Theorem is easily transferable from the classical predicate logic calculus to TRIPLE.

Furthermore the notion of the *Herbrand-Base*, *Herbrand-Interpretation*, and *Herbrand Model* can be defined analogously to the predicate logic case based on ground molecules. These definitions justify the definition of the Horn subset of TRIPLE .

$A : N \longrightarrow \text{resource}(A, N)$	(1)
$O[P \rightarrow V] \longrightarrow \text{statement}(O, P, V)$	(2)
$S @ M \longrightarrow \text{true}(S, M)$ for statements $S$	(3)
$\langle S \rangle \longrightarrow S$ for statements $S$	(4)
$O[P_1 \rightarrow V_1; P_2 \rightarrow V_2; \dots] @ M \longrightarrow O[P_1 \rightarrow V_1] @ M \wedge$	(5)
$O[P_2 \rightarrow V_2] @ M \wedge \dots$	
$\text{true}(S, M_1 \cap M_2) \longrightarrow \text{true}(S, M_1) \wedge \text{true}(S, M_2)$	(6)
$\text{true}(S, M_1 \cup M_2) \longrightarrow$ two rules, one with $\text{true}(S, M_1)$	(7)
the other with $\text{true}(S, M_2)$ . The other atoms remain.	(8)

**Fig. 3.** The rewrite rules for the translation of TRIPLE to conventional horn rules

**Definition 12 (Horn TRIPLE).** *The Horn-subset HORN of a  $\mathcal{T}$ -language is defined as the set of all clauses with at most one positive molecule. The view expression part of the positive molecules is either a simple term or an intersection expression. Of the negative molecules the view expression part is either a simple term, intersection expression, or a union expression (no set-theoretic difference).*

The restriction to molecules with the specific view expressions is necessary to ensure the applicability of the usual type of fixpoint definition for the semantic characterization. The Set-difference operator on views causes the usual fixpoint procedure (direct consequence operator, see [15]) to be non-monotonic, therefore a least fixpoint cannot be guaranteed. The union-operator in the head of a clause invalidates the model intersection property, that guaranties that the intersection of two Herbrand models is again another Herbrand model. The model intersection property is also necessary to guarantee the existence of a least Herbrand model.

Using the definition of the Horn-subset and the Herbrand base the usual semantic characterizations for logic programs can be adapted for the Horn part of TRIPLE.

Using notions for non-monotonic negation developed in the logic programming community (e.g., well-founded negation [24]) it is possible to reintroduce set-difference again.

## 5 Implementation

We implemented a prototype for TRIPLE using an existing deductive database (XSB [20]) by applying a source transformation procedure. The source transformation applies the rewrite rules provided in figure 3. Resources and statements are replaced by function symbols `resource` and `statement`. A molecule is replaced by a dyadic predicate symbol `true`, indicating membership of a statement in a view. The intersection-operator is rewritten in a conjunction of two predicates. The union-operator generates two rules, where the original predicates are replaced with a new true predicate for each argument of the union-operator.

The implementation is available for download and experiments at <http://triple.semanticweb.org>.

## 6 Examples

### 6.1 Querying an RDF Schema with RDF Schema semantics

This section shows how a view defined by some rules axiomatizing (part of the) semantics of RDF Schema are implemented in TRIPLE. Figure 4 show the RDF Schema specification. The first lines define namespaces (for RDF and RDF Schema) and abbreviations (for type, subPropertyOf and subclassOf).

The rules are enclosed by a view specification block:  $\forall Mdl \text{ @rdfschema}(Mdl) \{ \dots \}$

The Skolem function `rdfschema(Mdl)` is the view identifier of all facts derived by the rules enclosed by the view specification block. The parameter  $Mdl$  denotes the RDF Schema specification. The view `rdfschema(Mdl)` contains all

statements from the model *Mdl* plus everything derived additionally by the rules. The rule

$$\forall O, P, V \ O[P \rightarrow V] \leftarrow O[P \rightarrow V]@Mdl.$$

specifies that every triple contained in the view *Mdl* is also element of the view with the identifier *rdfschema(Mdl)*. The next rule defines the inheritance of values from sub properties to super properties. The remaining rules define the semantics of transitive properties (*subPropertyOf* and *subClassOf*) and of the type property.

```

rdf := 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'.
rdfs := 'http://www.w3.org/2000/01/rdf-schema#'.
type := rdf:type.
subPropertyOf := rdfs:subPropertyOf.
subClassOf := rdfs:subClassOf.
FORALL Mdl @rdfschema(Mdl) {
  FORALL O,P,V O[P->V] <-
    O[P->V]@Mdl.
  FORALL O,P,V O[P->V] <-
    EXISTS S S[subPropertyOf->P] AND O[S->V].
  FORALL O,P,V O[subClassOf->V] <-
    EXISTS W (O[subClassOf->W] AND W[subClassOf->V]).
  FORALL O,P,V O[subPropertyOf->V] <-
    EXISTS W (O[subPropertyOf->W] AND W[subPropertyOf->V]).
  FORALL O,T O[type->T] <-
    EXISTS S (S[subClassOf->T] AND O[type->S]).
}

```

Fig. 4. RDF Schema in TRIPLE

In Figure 5, a simple RDF Schema for motor vehicles is given: the root class is *xyz:MotorVehicle*, which has the direct subclasses *xyz:PassengerVehicle*, *xyz:Truck*, and *xyz:Van*. *xyz:MiniVan* is defined as a common subclass of *xyz:Van* and *xyz:PassengerVehicle*.

```

@cars {
  xyz := 'http://www.w3.org/2000/03/example/vehicles#'.
  xyz:MotorVehicle[rdfs:subClassOf -> rdfs:Resource].
  xyz:PassengerVehicle[rdfs:subClassOf -> xyz:MotorVehicle].
  xyz:Truck[rdfs:subClassOf -> xyz:MotorVehicle].
  xyz:Van[rdfs:subClassOf -> xyz:MotorVehicle].
  xyz:MiniVan[
    rdfs:subClassOf -> xyz:Van;
    rdfs:subClassOf -> xyz:PassengerVehicle].
}

```

Fig. 5. RDF Schema Example

The following query searches for all direct and indirect subclasses of *xyz:MotorVehicle*, using the RDF Schema definition for *rdfs:subClassOf* as defined in the *rdfschema(Mdl)* model:

```
FORALL C <- C[rdfs:subClassOf->xyz:MotorVehicle]@rdfschema(cars).
```

This is achieved by passing the RDF Schema (*cars*) as a parameter to the RDF Schema rules, whereas the query

```
FORALL C <- C[rdfs:subClassOf->xyz:MotorVehicle]@cars.
```

results in just the direct subclasses of *xyz:MotorVehicle*. The view *rdfschema(cars)* could be passed as a parameter to another view. Similar view definitions for languages like TopicMaps or UML can be given.

## 6.2 Using Views to Integrate Information about Educational Resources

This section presents an example where resources and resource annotations are distributed widely over a large number of peers, in the field of educational resources and materials. The example is inspired by the EDUTELLA project [17], which is aiming at establishing a Peer-to-Peer infrastructure for exchanging educational resources based on metadata. The following example shows the possibilities of TRIPLE for the integration of different resources and resource metadata. We assume three different metadata schemata used for describing educational resources and materials:

- The standard LOM metadata schema [6], using an RDF binding. This metadata schema specifies how to describe educational resources, and is used as the core part of the SCORM [22] standard which has the goal to ensure that educational resources can be exchanged between different repositories. The LOM RDF Binding reuses several other metadata schemata where available, such as the Dublin Core schema for describing basic properties of digital documents and the vCard schema to define person properties (name, affiliation, address, etc.)
- A topic ontology for a specific area, in our case software engineering, based on the SWEBOK classification [16, 4] and our representation of it in RDF. We are using the lom\_cls:taxonomy and the lom\_cls:taxon field from the LOM RDF Binding to structure the topic ontology.
- A reviewing schema used for describing reviews of educational resources. As there is no such standard yet (though the need for reviewing learning objects has already led to several projects focusing on assessment issues for educational resources), we use a simple RDF schema to describe such reviews using a numeric rating system, reviewer name, reviewer affiliation, and reviewer report.

In TRIPLE, we have the following declarations and views: The first collection defines the set of educational resources, described using the LOM RDF Binding, plus an additional rule to derive dcq:hasPart automatically from dcq:isPartOf (inverse), and to automatically annotate learning objects already annotated by a specific topic by all its super topics (in order to retrieve this learning object also for more general thematically queries).

```
// dc := '...'.
// lom := '...'.
// dcq := '...'.
// l3s := '...'.
// dfki := '...'.

// collection of l3s documents
@l3s:learningobjects {
  l3s:L3P1[
    rdf:type -> l3s:PDF;
    dc:title -> "Introduction to Requirements Documents";
    dc:creator -> l3s:fs [vCard:fn -> "Friedrich Steimann";
                       vCard:org -> "University of Hannover"];
    lom_cls:taxonomy -> swelok:swelok;
    lom_cls:taxon -> swelok:ReqDocument;
    dcq:isPartOf -> l3s:L3].
  l3s:L3[dc:title -> "Requirements Engineering"].
  l3s:L3P1a[dc:title -> "L3P1a"; dc:creator -> l3s:nn1;
           dcq:isPartOf -> l3s:L3P1].
  l3s:L3P1b[dc:title -> "L3P1b"; dc:creator -> l3s:nn2;
           dcq:isPartOf -> l3s:L3P1].
  l3s:L3P1b1[dc:title -> "L3P1b1"; dc:creator -> l3s:nn3;
            dcq:isPartOf -> l3s:L3P1b].

  FORALL P1, P2
    P1[dcq:hasPart -> P2] <- P2[dcq:isPartOf -> P1].

  FORALL LO, SuperTopic, SubTopic
    LO[lom_cls:taxon -> SuperTopic] <-
      LO[lom_cls:taxon -> SubTopic]
  AND
```

```

        SuperTopic[lom_cls:sub_topic -> SubTopic]@swebok:swebok.
    }

```

Furthermore, we have the swebok view, containing the complete swebok ontology, hierarchically structured by lom\_cls:sub\_topic:

```

// domain ontology
@swebok:swebok {
    swebok:ReqSpecification [lom_cls:sub_topic -> swebok:ReqDocument;
                            lom_cls:sub_topic -> swebok:DocQuality].
    // ...
}

```

Another view contains the reviewer database:

```

// reviewer database
@dfki:reviewer_db {
    l3s:L3P1 [l3s:reviewer -> stanford:stefan [vCard:fn -> "Stefan";
                                                vCard:org -> "Stanford University"];
             l3s:rating -> 1;
             l3s:review -> "review text ..."].
}

```

And finally, we define a parameterized view, which takes two views (the learning objects and the reviews) and three additional parameters (topic, rating, and affiliation), in order to search for specific learning objects on a certain topic, specified rating, and affiliation of the reviewer. Furthermore, we specify the hasPart relation as transitive and return all parts of found learning objects, and for each learning object, return its dc:creator and dc:title property, as well as the name of the reviewer.

```

// collection of good requirements engineering learning objects
FORALL LOModel, ReviewerModel, Topic, Rating, Affiliation
@GOOD_RE_LOs(LOModel, ReviewerModel, Topic, Rating, Affiliation) {
    FORALL LO, C, T, R, N
        LO [dc:creator -> C;
            dc:title -> T;
            l3s:reviewer -> R[vCard:fn -> N]
        ]
        <- LO [dc:creator -> C;
            dc:title -> T;
            lom_cls:taxon -> Topic
        ]@LOModel
        AND
        LO [l3s:reviewer -> R[vCard:fn -> N;
            vCard:org -> Affiliation];
            l3s:rating -> Rating]@ReviewerModel.

    FORALL LO, P
        LO[dcq:hasPart -> P] <-
            EXISTS A,B
                LO[A->B]
            AND
                LO[dcq:hasPart -> P]@LOModel.

    FORALL LO1, LO2, C, T, P
        LO1 [dc:creator -> C;
            dc:title -> T]

```

```

<-
  L02[dcq:hasPart -> L01]
  AND
  L01[dc:creator -> C;
      dc:title -> T]@LOModel.
}

```

A sample query then asks for all educational resources in the L3S repository, rated by reviews from the DFKI reviewer database, which have been rated by Stanford faculty with a rating of 1.

```

// query
FORALL O, P, Q <-
  O[P -> Q]@GOOD_RE_LOs(l3s:learningobjects, dfki:reviewer_db,
    swebok:ReqSpecification, 1, "Stanford University").

```

## 7 Related Work

### 7.1 Parameterized Views for Semi-structured Data

Other view mechanisms for semi-structured data have been investigated: [1] defines a view mechanism for Lorel. Lorel uses the select-from-where syntax of SQL and can be considered as extension to OQL that provides path expressions for traversing the data and extensive coercion rules for a more forgiving type system. The view definition in Lorel can import objects and edges from a source database into a view and new objects and edges can be created in the view. TRIPLE currently does not support regular path expressions, but rules can be used to implement these path expressions. However, Lorel does not support parameterized views.

### 7.2 Formalizations of RDF

Other formalizations of RDF include the RDF Model Theory developed by Pat Hayes [9]. The differences and commonalities with the model theory for TRIPLE presented in the previous sections (dubbed TRIPLE-M) are as follows:

- TRIPLE-M is a model theory for a complete logic based on RDF-like graph structures. It includes triples as well as formulae built with triples, whereas the Hayes' Model theory it only defined for the pure graph.
- TRIPLE-M defines the denotation for *RDF-models* (sets of RDF statements).
- Resource-identifiers are viewed as consisting of a namespace part and a localname part, whereas in the Hayes' model theory identifiers are monolithic.
- Literals have a language and a parse type.
- B-nodes in TRIPLE-M are treated in the same way—as existentially quantified variables. However, in contrast to the Hayes' interpretation of RDF, also existentially quantified predicates are allowed.
- The interpretation of reification is different in TRIPLE-M. In TRIPLE-M we assume that each statement has a unique ID.

## 8 Conclusion and Future Work

We presented a logical language and accompanied model theory, which formalized the notion of parameterized views for semi-structured data. Using the full language we identified a subset which is implementable using a deductive database. The language enables one to pass views as parameters to other views, realizing composition of views.

We illustrated the language with two examples: querying data with respect to a certain semantics, and integrating various metadata schemata. We expect both types of application scenarios to become common on the semantic web, and we foresee a need for parameterized views as presented and formalized in this paper.

No work has so far been done in the optimization of the execution of rules—the current implementation is a straightforward translation to an ordinary deductive database, thus no specialized optimization strategies are exploited. Future work also includes the investigation of distributed queries and data sources, which is a generalization of the work done in the MSL context [18].

## References

1. Serge Abiteboul, Roy Goldman, Jason McHugh, Vasilis Vassalos, and Yue Zhuge. Views for semistructured data. In *Proceedings of the Workshop on Management of Semistructured Data, Tucson, Arizona*, May 1997.
2. T. Berners-Lee, R. Fielding, and L. Masinter. RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax. Technical report, IETF, 1998.
3. Tim Berners-Lee. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. Harper San Francisco, September 1999.
4. Pierre Bourque, Robert Dupuis, Alain Abran, James W. Moore, and Leonard Tripp. The guide to the software engineering body of knowledge. *IEEE Software*, 16:35–44, November 1999.
5. Tim Bray, Dave Hollander, and Andrew Layman. Namespaces in XML. Technical report, Recommendation, W3C, 1999.
6. IEEE Learning Technology Standards Committee. Draft standard for learning object metadata, ver. 6.4. <http://ltsc.ieee.org/wg12/index.html>, March 2002.
7. Stefan Decker, Dan Brickley, Janne Saarela, and Juergen Angele. A query and inference service for RDF. In *QL'98 — The Query Languages Workshop*, Boston, USA, 1998. WorldWideWeb Consortium (W3C).
8. Harald Alvestrand (ed.). RFC 1766: Tags for the Identification of Languages. Technical report, IETF, 1995.
9. Patrick Hayes. RDF Model Theory W3C Working Draft 14 February 2002. Technical report, W3C, 2002.
10. G. Karvounarakis, V. Christophides, D. Plexousakis, and S. Alexaki. Querying CommunityWeb Portals, 2001.
11. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, July 1995.
12. Martin S. Lacher and Stefan Decker. RDF, topic maps, and the semantic web. *Markup Languages: Theory and Practice*, 2002. Accepted for publication.
13. Ora Lassila and Ralph R. Swick. Resource description framework (RDF) model and syntax specification. Technical report, Recommendation, W3C, 1999.
14. A. Y. Levy, A. Rajaraman, and J. D. Ullman. Answering queries using limited external query processors. *Journal of Computer and System Sciences*, 58(1):69–82, 1999.
15. J. W. Lloyd. *Foundations of Logic Programming*. Springer Verlag, 1987.
16. James W. Moore, Pierre Bourque, Robert Dupuis, Alain Abran, and Leonard Tripp. Software engineering body of knowledge project. In *Encyclopedia of Software Engineering*. John Wiley & Sons, 2002. <http://www.mrw.interscience.wiley.com/ese/articles/sof312/frame.html>.
17. Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. Edutella: A p2p networking infrastructure based on RDF. In *Proceedings of the 11th International World Wide Web Conference (WWW2002)*, Hawaii, USA, June 2002.
18. Y. Papakonstantinou. Query processing in heterogeneous information sources, 1996.
19. Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. In P. S. Yu and A. L. P. Chen, editors, *11th Conference on Data Engineering*, pages 251–260, Taipei, Taiwan, 1995. IEEE Computer Society.
20. Konstantinos Sagonas, Terrance Swift, and David S. Warren. XSB as an efficient deductive database engine. In *Proc. of the 1994 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'94)*, pages 442–453, 1994.
21. D. Suci. An Overview of Semistructured Data. *SIGACT News*, 29(4):28–38, December 1998.
22. ADL Technical Team. Scorm version 1.2 documents. <http://www.adlnet.org/index.cfm?fuseaction=scorm12>, October 2001.
23. Motomichi Toyama. Parameterized view definition and recursive relations. In *Proceedings of the Second International Conference on Data Engineering, February 5-7, 1986, Los Angeles, California, USA*, pages 707–712. IEEE Computer Society, 1986.
24. Allen van Gelder, Kenneth Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

## Appendix A: Definitions

The following definitions are taken from the RDF Test Cases document<sup>8</sup>.

<sup>8</sup> See <http://www.w3.org/TR/rdf-testcases/>. For the definitions in this section the following copyright notice applies: *Copyright © 2000 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved.*

## Strings

Strings are defined to be sequences of characters in the range of [#x20-#x7E] (US-ASCII) with the following escape sequences:

Escape sequence	Encodes Unicode character
\\	Backslash character (decimal 92, #x5c)
\"	Double quote (decimal 34, #x22)
\n	Line feed (decimal 10, #xA)
\r	Carriage return (decimal 13, #xD)
\t	Horizontal tab (decimal 9, #x9)
\u xxxx	4 required hexadecimal digits xxxx encoding character [#x0-#x8],[#xB#xC],[#xE-#x1F],[#x7F-#xFFFF]
\U xxxxxx	8 required hexadecimal digits xxxxxx encoding character [#x10000-#x10FFFF]

Strings are enclosed in double quotes—these double quotes are not considered to be part of the string.

## Resource Constructors

Resource constructors are used to built URI references, and therefore the same character encoding rules as in the URI specification apply: resource constructors are sequences of US-ASCII characters. The standard escaping procedure is described in [2] using UTF-8 as the character encoding. Disallowed characters are represented in UTF-8 and then encoded using the %HH format, where HH is the byte value expressed using hexadecimal notation.

Characters above the US-ASCII range are made available by the \u or \U escapes as described in section Strings for ranges [\x80-\xFFFF] and [\x10000-\x10FFFF] respectively.

Characters [\x0-\x1F] and \x7F are forbidden in URI references. Resource constructors may be inclosed in single quotes.