Stefan Decker, Michael Sintek, Andreas Billig, Nicola Henze
Peter Dolog, Wolfgang Nejdl, Andreas Harth, Andreas Leicher,
Susanne Busse, Jörn Guy Süß, Zoltán Miklós, Jose-Luis Ambite
Matthew Weathers, Gustaf Neumann, Uwe Zdun

TRIPLE –
an RDF Rule Language with
Context and Use Cases

Stefan.Decker@deri.org
Motivation: Why Rule Languages for the Web

• “Rule Languages for Interoperability” or “Rule Language Interoperability”?  
  – Focus on the former
• More and more data becomes available – Interoperability required  
  Need for:  
  – Querying  
  – Integration  
  – Transformation

=> “Time to Market”: Faster to write rules than code for transformation, integration
Guiding Requirements for an RDF Rule Language

- Support RDF (graph-structured data)
- Handle multiple modeling semantics
  - (OWL, UML, ER, TopicMaps, XML-Schema, Relational Data, special purpose data models)
  - Special query systems for all of them?
- Distributed, heterogeneous data sources
  - Not all data is created equal
RDF Contexts (e.g., Named Graphs)

- **Support for RDF contexts**
  - Source (is a restaurant vegetarian or not?)
  - Time stamps (e.g., sensor data)
  - Location (geospatial),
  - Culture (content rating)
  - ...

Stefan’s Data

Jim’s Data

Frank’s Data
TRIPLE: Brief Language Overview

- **Native support**
  - for Resources & namespaces,
  - Abbreviations
  - Object invention
  - Contexts (sets of RDF statements)
  - Reification

- **Rules with expressive bodies (full FOL syntax)**

- Inspired by F-Logic:
  - `subject[predicate→object]` (“molecule”)

- Extended by: context, context expressions, parameterized contexts:
  - `s[p→o]@m` “triple <s,p,o> in model m”
  - `s[p→o]@(m1 ∩ m2)` context intersection, union, diff
  - `s[p→o]@sf(m1, X, Y)` Skolem function
dc := “http://purl.org/dc/elements/1.0/”.
deri := “http://www.deri.ie/”.

Example: Querying and Inferencing Dublin Core with Object Invention

@deri:documents {
  deri:d_01_01 [ dc:title → TRIPLE;
  dc:creator → “Stefan Decker”;
  dc:subject → RDF;
  dc:subject → triples; ... ].
}

∀N p(N)[ rdf:type → xyz:Person;
  xyz:name → N ] ←
  ∃D D[dc:creator → N].

∀N ← ∃P P[rdf:type → xyz:Person;
  xyz:name → N]@deri:documents.
Semantic Spaces: Specifying Semantics via Parameterized Contexts

- RDF Schema, UML (and other frame/OO systems): semantics can be \textit{directly} defined in TRIPLE as a parameterized context
- OWL (i.e., expressive ontology languages, DL): requires interaction with \textit{foreign} reasoning components (e.g., DL classifier)
Example: RDF Schema Semantic Space

rdf := 'http://www.w3.org/.../rdf-syntax-ns#'.

rdfs := 'http://www.w3.org/.../PR-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,V   O[subClassOf->V] <-
                   EXISTS W   (O[subClassOf->W] AND W[subClassOf->V]).
    ...
}

namespace abbreviations
resource abbreviations
model block
"copy" triples from Mdl
Transitivity of subClassOf
Example: Cars Ontology

```prolog
@cars {
  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].
}
```

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

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

X = xyz:Van
X = xyz:Truck
X = xyz:PassengerVehicle
X = xyz:MiniVan
Example: UML Semantic Space

```prolog
rdf := 'http://www.w3.org/...rdf-syntax-ns#'.
uml := 'http://www.omg.org.uml/1.3/'.

FORALL Mdl @uml(Mdl) {
    FORALL O,P,V   O[P->V] <-
                   O[P->V]@Mdl.
    FORALL X,Z   g(X,Z)[rdf:type->uml:Generalization;
                        uml:Generalization.child->X;
                        uml:Generalization.parent->Z]<-
                   EXISTS Y,G1,G2
                        G1[uml:Generalization.child->X; uml:Generalization.parent->Y]
                        AND
                        G2[uml:Generalization.child->Y; uml:Generalization.parent->Z].

    ...}
```

Transitivity of Generalization
OWL Semantic Space

OWL can be handled by

- Vocabulary only
- DLP (only for the intersection between Horn and DL)
- Connection to an external OWL reasoner
  - Ship RDF to reasoner
  - Import result RDF back within the OWL Semantic Space
Example: Integrate Information from several Sources

∀ s1, s2 integ(s1, s2) {
    .......
}

Integr(source1, source2)

rules for integrating two sources
Compilation to Horn logic + NAF

- First implementation (and informal semantics) by mapping to Horn Logic / XSB system (Prolog with tabled resolution)
- Lloyd-Topor transformation for quantifiers etc.
- RDF-specific transformations given as rewrite rules:

\[
\begin{align*}
A : N & \quad \rightarrow \quad \text{resource}(A, N) \\
O[P \rightarrow V] & \quad \rightarrow \quad \text{statement}(O, P, V) \\
S@M & \quad \rightarrow \quad \text{true}(S, M) \quad \text{for statements } S \\
<S> & \quad \rightarrow \quad S \quad \text{for statements } S \\
O[P_1 \rightarrow V_1; P_2 \rightarrow V_2; \ldots]@M & \quad \rightarrow \quad O[P_1 \rightarrow V_1]@M \land \quad O[P_2 \rightarrow V_2]@M \land \ldots \\
\text{true}(S, M_1 \cap M_2) & \quad \rightarrow \quad \text{true}(S, M_1) \land \text{true}(S, M_2) \\
\text{true}(S, M_1 \setminus M_2) & \quad \rightarrow \quad \text{true}(S, M_1) \land \neg \text{true}(S, M_2) \\
X := Y. S(X) & \quad \rightarrow \quad \forall X \ (X = Y \land S(X)) \quad \text{for clause sequences } S(X)
\end{align*}
\]
Projects involving TRIPLE

• Conflict Analysis and Model Transformation based on TRIPLE (TU Berlin)
• Ontology Management with TRIPLE (Fraunhofer, Berlin)
• Ontology based matchmaking on the Grid (University of Southern California)
• Goods movement planning problems (University of Southern California)
• Personalization Services for the Semantic Web with TRIPLE (University of Hannover)
• Querying Semantic Web Resources Using TRIPLE Views (Technical University of Vienna)
• SmartWeb - Multi-modal access to the Semantic Web (DFKI)
Conclusions

• RDF data is contextualized and rule language need primitives to deal with context
• RDF as a Semantic Web foundation needs to express data with different semantics
• The TRIPLE context mechanism provides a flexible means to handle context
• Prototype available at http://triple.semanticweb.org
Syntax and Semantics Definition (for Peter)

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

- a set of resource constructors, \( \mathcal{F} \);
- an infinite set of variables, \( \mathcal{V} \);
- an infinite set of strings, \( \mathcal{H} \);
- a set \( \mathcal{E} \) of language identifiers plus a single identifier ‘NULL’;
- auxiliary symbols such as \( ( ) \), \( . \), \( : \), \( \leadsto \), \( \exists \), \( XML \); and
- usual logical connectives and quantifier, \( \land, \lor, \lnot, \leftrightarrow, \rightarrow, \forall, \exists \).

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 \( t_1, \ldots, t_n \) are terms, then \( f(t_1, \ldots, t_n) \) 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 local name part of the resource-id.
- If \( s, p, \) and \( o \) are terms, then \( s[p \rightarrow o] \) is a term, also called a statement-id. A statement-id is also a resource-id.
- These are all terms.

Definition 4 (Context Expressions) Context expressions are defined inductively as follows:

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

Definition 5 (Molecular Formulae) Given a statement \( s[p \rightarrow o] \) and a context expression \( m \), the expression \( s[p \rightarrow o] \circ m \) is called a molecule.

As a notational 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 \( \varphi \lor \psi \), \( \varphi \land \psi \), \( \varphi \rightarrow \psi \), \( \varphi \leftarrow \psi \), \( \lnot \varphi \), and \( (\varphi) \).
- If \( X \) is a variable, and \( \varphi \) is a formula, then \( \forall X \varphi \) and \( \exists X \varphi \)
  are formulae.
- These are all formulae.
Model Structure

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

- \( S_0 = R \times R \times (R \cup H) \)
- \( S_{n+1} = (R \cup S_n) \times (R \cup S_n) \times (R \cup H \cup S_n) \) for \( n \geq 0 \).

Then \( S \) is defined as \( \bigcup_{i=0}^{\omega} S_i \).

\( C \) is a mapping from \((U \cup R \cup S \cup J) \times (U \cup R \cup S \cup J)\) to \( R \).

\( M \) is a mapping from \( R \cup S \) to \( 2^S \).

\( I \) is an interpretation function defined as follows:

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

Definition 8 (Variable Assignment) A variable assignment for a TRIPLE-language \( L \) and a T-structure \( T \) is a mapping \( B : V \rightarrow (U \cup R \cup S \cup J) \), 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 \( T \) and a variable assignment \( B \) (noted as \( I^{T,B} \)) is defined as follows:

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

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

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

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

- $T, B \models \varphi \land \psi$ iff $T, B \models \varphi$ and $T, B \models \psi$;
- $T, B \models \varphi \lor \psi$ iff $T, B \models \varphi$ or $T, B \models \psi$;
- $T, B \models \varphi \rightarrow \psi$ iff $T, B \models \varphi$ then also $T, B \models \psi$;
- $T, B \models \varphi \leftrightarrow \psi$ iff $T, B \models \varphi$ then also $T, B \models \varphi$;
- $T, B \models \lnot \varphi$ iff $T, B \not\models \varphi$;
- $T, B \models \forall x \varphi$ if for all $d \in \mathcal{U} \cup \mathcal{R} \cup \mathcal{S} \cup \mathcal{H}$, $T, B_x^d \models \varphi$, where
  
  $B_x^d(y) := \begin{cases} 
  d & \text{if } x = y \\
  B(y) & \text{otherwise}
  \end{cases}$

- $T, B \models \exists x \varphi$ iff there exists $d \in \mathcal{U} \cup \mathcal{R} \cup \mathcal{S} \cup \mathcal{H}$, $T, B_x^d \models \varphi$, with $B_x^d$ defined as above.
Use-Case: Personalization Services for the Semantic Web

Problem
• Personalization functionality is not reusable but hidden in applications
• currently: no plug & play approaches offering dynamically personalized services
• to reach this goal: distributed sources (user profiling, context detection, resource usage, etc.) have to be collected and evaluated accordingly

Nicola Henze, L3S, Hannover

Approach
• Personalization rules reason over distributed resources: TRIPLE
• Personalization functionalities are encapsulated in Web services
• Available personalization services are registered in a Web service registry
• Connector service orchestrates communication
• Visualization services syndicate results according to currently used device

Results
• Realized Personal Readers:
  • e-Learning: for Java, and Semantic Web
  • Personal Publication Browser for the NoE REWERSE – Reasoning on the Web
• Prototypes available at
  • www.personal-reader.de

N. Henze: “Personal Readers: Personalized Learning Object Readers for the Semantic Web” (AIED’05)
Use Case: Ontology Management

**Approach**

- Using **TRIPLE/RDF** to represent *classes*, instances and their interrelations as *first class objects*
- Using **TRIPLE contexts** to partition the artefact set (req. 1)
- Using **TRIPLE rules** for derivations (req. 2)
- Using **TRIPLE parameterized contexts** to support transformations (req. 3)

**Goal:** integrative management and usage of *domain artefacts*

**Requirements**

A repository language has to support

1. **Partitioning** of the whole artefact set into artefact groups
2. **Deriving** of new artefact relations
   - (competenceIn)
3. **Transformation** between the elements of the artefact groups

**Results**

- **Repository prototype ODIS** at Fraunhofer ISST (available on demand)
- **Enhancements** of **TRIPLE**:
  - reintroduction of the *multi-value-construct* (~ F-Logic)
  - enabling *mapping-chains* ((a <- b) <- c)
  - distinction between *intensional* and *extensional statements* (~ deductive DB)

Andreas Billig, Fraunhofer, Berlin