T-RDFS-Log – Syntax

Jakob Huber
DWS - University of Mannheim

January 2015

1 Introduction

T-RDFS-Log is a probabilistic (temporal) RDFS reasoner that is designed for knowledge base verification. In this document, we explain how to annotate RDFS statements with weights and intervals (see Section 2), and outline how to define domain-specific constraints and rules (see Section 3).

2 Annotating Statements

It is necessary to assign weights to uncertain facts in order to allow T-RDFS-Log to remove them from the knowledge base. For this purpose, we rely on reification as RDF does not support probabilistic or temporal statements.

Hence, one has to create a node that identifies the statement (see Fig. 1). To this node, one can assign weights, using the property `http://algebra.org/weight`, and the properties `http://algebra.org/start` and `http://algebra.org/end` to assign a temporal interval (see Fig. 2):

```
John  attended   University_1.

[ rdf:type      rdf:Statement ;
  rdf:subject   John ;
  rdf:predicate attended ;
  rdf:object    University_1 ;
  weight        "1.0"^^xsd:decimal ;
  start         "2002";
  end           "2006"
].
```

We also provide a shortcut to annotate a statements with multiple intervals. Therefore, it is possible to possible to attach a `rdf:Bag` with the property `http://algebra.org/hasNode` to a statement. The bag contains the different intervals and weights:

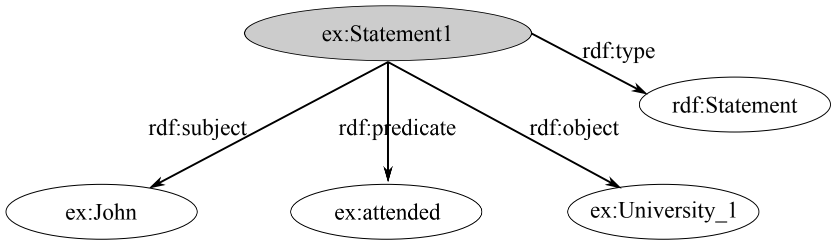


Figure 1: Statements: Reification.

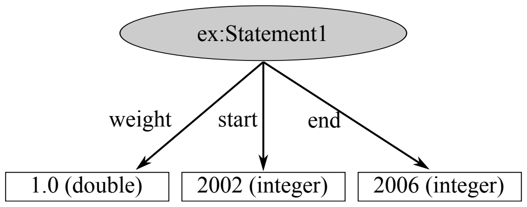


Figure 2: Statements: Direct Annotation.

```
John  attended   University_1.

[ rdf:type      rdf:Statement ;
  rdf:subject   John ;
  rdf:predicate attended ;
  rdf:object    University_1 ;
  hasNode       [  a rdf:Bag;
                   rdf:_1 [weight   "1.0"^^xsd:decimal ;
                           start    "2002";
                           end      "2004"];
                   rdf:_2 [weight   "0.5"^^xsd:decimal ;
                           start    "2005";
                           end      "2006"]
                ]
].
```

3 Defining Rules & Constraints

We also provide a possibility to define rules and constraints in a RDF document. For each, rule (and constraints) it is necessary to create a node of type `http://c.org/Constraint`. The information associated with the respective rule get attached to this node. A rule can either be a hard rule `http://c.org/isHard` or have a weight `http://c.org/weight`. The rule needs to be defined as a disjunction of literals. Hence, a bag containing the literals has to be attached to the node identifying the rule (see Fig. 3).

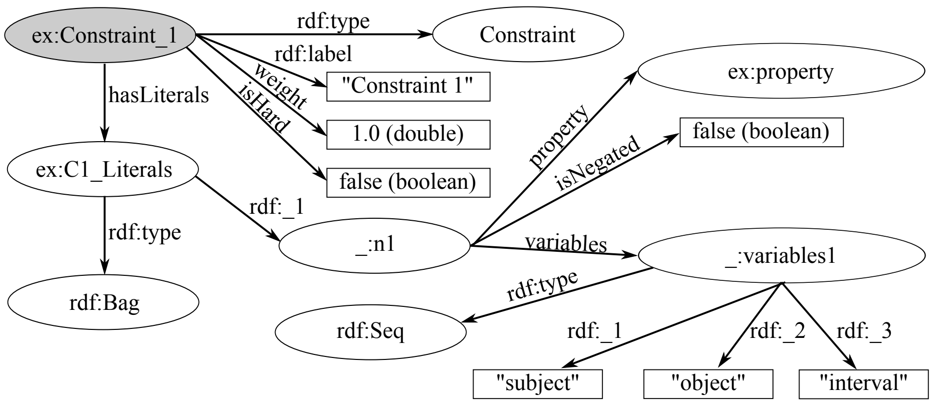


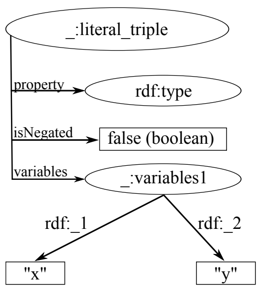
Figure 3: Constraints: RDF Model

Each literal can be negated or positive which has to be set with the property `http://c.org/isNegated`.T-RDFS-Log supports three types of literals. While it is required to state a property (which can also be a variable instead of a specific property) the number of variables (which can also be constants/URIs) varies (see Fig. 4). The property is specified by the property `http://c.org/property` and the sequence of variables by the property `http://c.org/variables`. Standard RDF triples have two variables (see Fig. 4(a)) as the (non-temporal) property is separately specified. Temporal statements have an additional variable for the interval (see Fig. 4(c)).

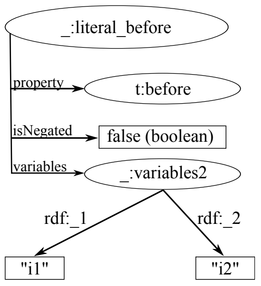
Moreover, T-RDFS-Log incorporates the temporal predicates of Allen's interval algebra that can also be used to define constraints. Hence, if the property of the literal is temporal and the literal has two variables, we transform it to a predicate of Allen's interval (see Fig. 4(b)). The following temporal properties exist: `http://algebra.org/allen/tBefore`, `http://algebra.org/allen/tMeets`, `http://algebra.org/allen/tOverlaps`, `http://algebra.org/allen/tStarts`, `http://algebra.org/allen/tDuring`, `http://algebra.org/allen/tFinishes`, and `http://algebra.org/allen/tEqual`.

Example The following constraint expresses that the `birthDate` of an entity cannot be annotated with the same interval as the `deathDate`.

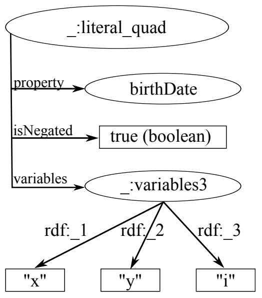
```
C1   rdf:type    Constraint;
     rdfs:label  "birthdate and deathdate cannot be annotated
                  with the same interval";
     isHard      "true"^^xsd:boolean;
     hasLiterals [
        a rdf:Bag;
        rdf:_1 [  isNegated "true"^^xsd:boolean;
                  isObserved "true"^^xsd:boolean;
```



(a) triple(x, “rdf:type”, y)



(b) tBefore(i1, i2)



(c) !quad(x, “birthDate”, y, i)

Figure 4: Constraints: Declaration of the different types of literals.

```
                property birthDate ;
                variables [ a rdf:Seq;
                            rdf:_1 "x"; rdf:_2 "y"; rdf:_3 "i"]
             ];
      rdf:_2 [  isNegated "true"^^xsd:boolean;
                property deathDate;
                variables [ a rdf:Seq;
                            rdf:_1 "x"; rdf:_2 "z"; rdf:_3 "i"]
             ]
   ].
```

This constraint will be converted to the following formula:

```
!quad(x,"birthDate",y,i) v !quad(x,"deathDate",z,i).
```