W3C Working Group Note

RIF In RDF

W3C Working Group Note 11 December 2012

This version:
http://www.w3.org/TR/2012/NOTE-rif-in-rdf-20121211/
Latest version:
http://www.w3.org/TR/rif-in-rdf/
Previous version:
http://www.w3.org/TR/2011/NOTE-rif-in-rdf-20110512/
Editors:
Sandro Hawke, W3C/MIT
Axel Polleres, DERI, NUI Galway

A color-coded version of this document showing changes made since the previous version is also available.

This document is also available in these non-normative formats: PDF version.

Abstract

This document specifies a reversible mapping (or transformation) from Rule Interchange Format (RIF) XML documents to Resource Description Framework (RDF) graphs. This mapping allows the contents of RIF documents to be interoperably stored and processed as RDF triples, using existing serializations and tools for RDF. When used with the standard mapping from RDF triples to RIF frames, this also provides a "reflection" or "introspection" mechanism, an interoperable way for RIF rules to operate on RIF documents.

Status of this Document

May Be Superseded

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

Set of Documents

This document is being published as one of a set of 12 documents:

1. RIF Overview
2. RIF Core Dialect
3. RIF Basic Logic Dialect
4. RIF Production Rule Dialect
5. RIF Framework for Logic Dialects
6. RIF Datatypes and Built-Ins 1.0
7. RIF RDF and OWL Compatibility
8. OWL 2 RL in RIF
9. RIF Combination with XML data
10. RIF In RDF (this document)
11. RIF Test Cases
12. RIF Primer

Document Unchanged

There have been no changes to the body of this document since the previous version. For details on earlier changes, see the change log.

Please Send Comments

Please send any comments to public-rif-comments@w3.org (public archive). Although work on this document by the Rule Interchange Format (RIF) Working Group is complete, comments may be addressed in the errata or in future revisions. Open discussion among developers is welcome at public-rif-dev@w3.org (public archive).

W3C Working Group Note

No Endorsement

Publication as a Working Group Note does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

Patents

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of Contents

- 1 Introduction
- 2 Use Cases
- 3 Requirements
- 4 Extensibility
- 5 Mapping from RIF XML to RDF Graphs
    - 5.1 Namespaces
    - 5.2 The <id> and <meta> Elements
    - 5.3 The <Var> and <Const> Elements
    - 5.4 General Mapping
- 6 The Reverse Mapping (Extracting RIF XML)
- 7 Importing RIF into RDF
- 8 Semantics of RIF in RDF
- 9 Acknowledgements
- 10 References
    - 10.1 Normative References
    - 10.2 Nonnormative References
- 11 Appendix Complete Example
- 12 Changelog
    - 12.1 Changes since 12 May 2011
    - 12.2 Changes since 22 June 2010

1 Introduction

The Rule Interchange Format (RIF) [RIF Overview] is an interlingua between rule systems. It is an overlapping family of XML languages (called "dialects") designed for transmitting and storing various kinds of computer-processable rules and related data. Three standard dialects have been defined: RIF Core [RIF Core], RIF Basic Logic Dialect (BLD) [RIF BLD], and RIF Production Rules Dialect (PRD) [RIF PRD]. RIF Core is a sublanguage of BLD and of PRD: every Core document is also a BLD document and a PRD document.

RIF was envisioned [RIF Charter] to be extensible, allowing third parties to define non-standard extensions which could be combined into new dialects, as needed, to support interchange of rule sets which include features not defined in the standard dialects. Despite this vision, no general mechanism for extensions has been detailed in the RIF specifications. The RIF Framework for Logic Dialects (FLD) [RIF FLD] does, however, specify a way to create more expressive logic dialects.

The Resource Description Framework (RDF) [RDF] is a standard abstract way to represent data. The units of data in RDF are triples consisting of a subject, property (or predicate), and value (or object). These triples are similar to (and compatible with) RIF Frames (see RIF-RDF Combinations [RIF RDF+OWL]). A set of triples can be viewed as a directed labeled graph, where the nodes are subjects and values and the arcs are labeled with property identifiers; we therefore speak of a set of RDF triples as an RDF graph. RDF graphs can be serialized in multiple equivalent syntaxes, including RDF/XML [RDF XML], RDFa [RDFa], and Turtle [Turtle]. RDF can be processed with a wide variety of software tools [RDF Tools].

This specification defines a reversible mapping from RIF syntactic structures to RDF graphs. The definition is presented in tables where each row in the tables shows an XML template and a corresponding RDF graph template. The mapping is performed, roughly speaking, by finding the first matching XML template, then producing the corresponding graph. In some cases, the graph will require recursive translations of XML subtrees. The resulting graph has one node, called the focus node, which represents the XML root node, which in RIF Core, BLD, and PRD is rif:Document.

A reverse mapping, described in section 6, is possible for standard RIF by simply matching the RDF template and generating the corresponding XML. For extended RIF, so the reverse mapping can only done if the translator knows the XML grammar being generated.

Note that RDF serializations produced via this mapping are not standard RIF documents and cannot necessarily be understood by RIF implementations.

The rest of this document is structured as follows:

- design discussion, including use cases and requirements
- specification of the RIF-in-RDF mapping
- discussion of the reverse mapping
- specification of rif:usedWithProfile, allowing import of RIF into RDF
- a complete example

2 Use Cases

In designing this mapping a few use cases were considered:

- **UC1: Store RIF in an RDF Triplestore** — particularly when using RIF with RDF data, it may be useful to keep various RIF documents in the triplestore with the data, especially when there is associated metadata.
- **UC2: Access RIF Syntactic Structures with RDF Tools** — even more than storing whole RIF documents, it may be useful to be able to use RDF and Linked Data mechanisms to refer to and manipulate individual syntactic elements such as RIF rules, clauses, and groups.
- **UC3: Transform RIF Syntax using RIF Rules** — many logic programming techniques build on the idea of having rules which transform other rules. While RIF documents can be processed as XML, it may be desirable in some cases to process them as RDF triples or as RIF frames.
- **UC4: Provide Forward Compatibility via Fallback Rules** — it may be possible for RIF extensions to be completely or partially understood (used) by systems which do not directly implement the extensions, if the extensions are published with suitable fallback transformation rules. This is a special case of UC3.

3 Requirements

The following requirements were taken into account in this design:

- **Req1: All Standard RIF Documents Map to RDF** — Every syntactically valid RIF Core, RIF BLD, and RIF PRD has a well-defined mapping to RDF triples.
- **Req2: Extensions Can Be Written So They Will Be Automatically Mapped to RDF** — It is possible to write reasonable extensions which the mapping will handle, without the mapping being extended. It is not a requirement that *all* possible extensions be handled by this mapping.
- **Req3: Transformations Require No External Data** — The transformation can be done without any external information, such as dereferencing namespace URIs or otherwise obtaining schema information.
- **Req4: Stable Roundtrips Under RDF Simple Entailment** — A RIF document may be mapped to RDF, then the graph may be altered following RDF Simple Entailment [RDF Semantics] (including being reduced to a subgraph), and *if* the document can be extracted by the reverse mapping, it will have the same entailments and metadata. This is motivated by UC3 and especially UC4: without this property, incomplete running of transformation rules could undetectably result in incorrect results.
- **Req5: RDF View Conforms to RDF Best Practices** — the RDF form of the RIF constructs should appear as normal, well-constructed RDF data, not as some odd or surprising formation. Although this mapping uses rdf:List structures more than is common, for this application they are warranted.
- **Req6: RIF Extension are First Class in RDF View** — viewed as triples, there should be no indication of which features are in which dialects or extensions; the intent here is the allow the feature set to evolve and particular applications to use the appropriate set of features without regard to which features happen to be in RIF Core, RIF BLD, or RIF PRD.

4 Extensibility

Conceptually, in RIF there are two kinds of extensions, divided by how consumers which do not implement the extension are to handle them. Extensions which non-implementing consumers may ignore are encoded in RIF metadata; extensions which non-implementing consumers must understand before processing require altering the syntax so that the RIF document will not be schema valid, such as by introducing new XML elements.

In the RDF mapping, this distinction must be made somewhat differently. In RDF, may-ignore extensions can be encoded with additional triples about RIF syntactic elements, while must-understand extensions require *removing* or *replacing* properties required for non-extended decoding. This absence of a required arc prevents the decoding to a schema-valid XML RIF document, causing non-implementing consumers to abort.

These restrictions are necessary in order to meet the stated requirements. In particular, without these restrictions, a RIF document (in RDF graph form) being transformed by an incomplete reasoner into another RIF document (also in RDF graph form) could produce an unintended and incorrect result just because the reasoning was incomplete. With these restrictions, the result will not match the reverse-mapping until it is sufficiently complete.

5 Mapping from RIF XML to RDF Graphs

The mapping from RIF XML to RDF Graphs is expressed as a function *Tr*:

```
Tr(rif-xml-tree) → <focus-node, triples>
```

For every standard RIF Document, and for certain subtrees of RIF documents and extended RIF documents, *Tr* maps to the pair of an RDF node and an RDF graph. The node, called the *focus* node represents the same syntactic element as the root of the given XML tree. The RDF graph is a standard RDF graph, a set of RDF triples, and always contains the focus node. The focus node is usually a fresh blank node, but it might have a IRI label in certain cases, as detailed below.

In this document, the Turtle [Turtle] RDF serialization syntax is used for expressing triples and graphs. Turtle has a very terse syntax for lists, *(item-1 ... item-n)* and for fresh blank nodes and the triples using them as the subject: *[property-1 value-1; ... property-n value-n]*. These constructs allow the mapping to be presented and examples to be shown with relative simplicity.

The mapping is defined recursively, with each application of *Tr* converting an XML *class element* to a focus node, with additional triples. Class elements in RIF XML have tags that begin with an uppercase letter and represent a particular syntactic entity. Except for rif:Var and rif:Const class elements (detailed in Table 1, below), all the class elements follow a general form, containing a sequence of *property elements*, each containing additional class elements. The mapping for these is detailed in Table 2 and Table 3, below.

For another example of a specification of a mapping to RDF graphs, which may lend insight into how to use this specification, see OWL 2 Mapping to RDF Graphs [OWL2 Mapping].

If a system performing the mapping has determined the input document to be in one of the three standard dialects (Core, PRD, BLD), the rdf:type of the root focus-node *may* be set corresponding to that dialect: rif:CoreDocument, rif:PRDDocument, or rif:BLDDocument, instead of rif:Document. These classes for the standard dialects are defined to be disjoint: rif:BLDDocument contains those BLD documents which are not Core documents, and rif:PRDDocument contains those PRD documents which are not Core documents. Extensions *may* result in additional rdf:type arcs on the root focus-node.

Note that RIF XML allows for relative IRIs, which are expanded to absolute IRIs using the xml:base directive. This expansion *must* be done before *Tr*.

5.1 Namespaces

All standard elements in RIF XML have the namespace "http://www.w3.org/2007/rif#", and the attributes have no namespace. Extensions are expected to use other namespaces for the elements and are not allowed to introduce new attributes.

The RIF-in-RDF mapping produces RDF graphs that use the **same namespace**, although they use that namespace name in the normal RDF way (as an IRI prefix) instead of in the XML way (as a disambiguator).

By keeping the namespace the same, transformation software can correctly operate, without modification, on all RIF documents, even ones containing extensions.

Note that this use of the same namespace means that in certain cases RIF and RDF/XML documents cannot be distinguished simply by their namespace use. Moreover, since the rdf:RDF root element is optional in RDF/XML, in some cases it is not possible to distinguish between RIF and RDF/XML documents just by schema-validating or RDF-parsing the XML. In those cases, additional inspection of the structure is necessary. In general, systems should therefore be careful to maintain external file type information. This is typically done with either the media types ("application/rif+xml" and "application/rdf+xml") or the suggested filename extensions (".rif" and ".rdf").

In the tables below, the followng XML DOCTYPE declaration is assumed, allowing for abbreviation of the RDF and RIF namespaces:

```
<!DOCTYPE rif:Document [
  <!ENTITY rdf   "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY rif   "http://www.w3.org/2007/rif#">
]>
```

Also, the default XML namespace is assumed to be "http://www.w3.org/2007/rif#" and for use in Turtle, the following prefix declarations are assumed to be in effect:

```
@prefix rif: <http://www.w3.org/2007/rif#>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix xs:  <http://www.w3.org/2001/XMLSchema#>
```

5.2 The <id> and <meta> Elements

Any RIF class element may have a first child of the form:

```
<id>
   <Const type="&rif;iri">id</Const>
</id>
```

When this child is present, it is ignored for other processing and the *id* text is used as the IRI (URI-Reference) label for the **focus_node**, instead of it being left as a blank node.

As a special case, if the rif:Document element does not have an <id> child, its **focus_node** *should* be given the Web address (IRI) of the input XML document, if it has one.

After this optional <id> element, any RIF class element may contain a <meta> element. This element is processed according to general element processing rules, below.

Systems performing this transformation *may* also attempt to convert the metadata to RDF using the standard Frame-RDF correspondence [RIF RDF+OWL] , and include it in the returned triples. The conversion is not always possible, because some frame formulas are not expressible in RDF; systems which attempt this transformation and encounter such frame formulas in metadata *should* issue a warning. Even if the frames are converted to RDF like this, implementations *must*, under default settings, still keep the rif:meta triples intact, to support stable roundtripping.

5.3 The <Var> and <Const> Elements

Table 1, below, defines a portion of *Tr*. When a rif-xml-tree, X, matches the entry in column one, treating terms written *like-this* as metavariables, the result of *Tr*(X) is the pair <**focus_node**,G>, where G is the set of triples indicated by the second column.

Note that, although not shown in this table, <id> and <meta> child elements are allowed before the text in these elements. Additional elements after <id> and <meta> and before the character data may be allowed by extended schemas and *must* be processed as normal (extended) property elements.

Note that metadata about Consts is applied to the **focus_node**, not to the RDF literal produced. For example an explanation comment on a Const is understood to explain why that value is used in that spot, not state general properties of that value.

Table 1: RIF-RDF *Tr* Mapping Table for Var and Const

Input XML Pattern, X	Output RDF Triples, G
`<Var>`*variable-name*`</Var>`	**focus_node** rdf:type rif:Var **focus_node** rif:varname "*variable-name*"
`<Const type="&rif;iri">`*value*`</Const>`	**focus_node** rdf:type rif:Const **focus_node** rif:constIRI "*value*" Note that the value is an absolute IRI. Relative IRIs must be converted using the xml:base.
`<Const type="&rif;local">`*value*`</Const>`	**focus_node** rdf:type rif:Const **focus_node** rif:constname "*value*"
`<Const type="&rdf;PlainLiteral">`*text*`@</Const>`	**focus_node** rdf:type rif:Const **focus_node** rif:value "*value*"
`<Const type="&rdf;PlainLiteral">`*text*`@`*langtag*`</Const>`	**focus_node** rdf:type rif:Const **focus_node** rif:value "*value*"@*langtag*
`<Const type="`*type-iri*`">`*value*`</Const>`	**focus_node** rdf:type rif:Const **focus_node** rif:value "*value*"^^*type-iri*

5.4 General Mapping

Except as noted above, the *Tr* mapping for any class element (denoted as *parent* in Table 2) is to a new focus node and a set of triples which depend on each of the children of the class element. The exact dependency is detailed in this section.

Each child of the class element being mapped (except as noted above) is a property element (denoted as *child* in Table 2). There are four kinds of property elements:

Mode 0
: These elements have the **ordered="yes"** attribute. Their children are mapped to an RDF list (collection).

Mode 1
: These elements are required by the XML schema to appear exactly once. Their children are mapped directly to the value role of an RDF triple

Mode 2
: All the (zero or more) values of these elements are gathered, in document order, into an RDF list. When these elements do not appear in their class elements, an empty RDF list is generated.

Mode 3
: Special handling for the <slot> property, converting name/arg and key/value pairs into explicit pairs

The mapping for each mode is specified in Table 2 below. The mapping depends on the identity of an RDF property, written as *prop*, and the mode. Table 3 specifies special-case values for *prop* and mode, but otherwise they are determined as follows:

1. *prop* is the concatenation of the property element's tag's namespace IRI followed by its local part. For example, for the <rif:args> element, the RDF property *prop* has the IRI "http://www.w3.org/2007/rif#args".
2. If the element has an attribute "ordered" with the value "yes", it is Mode 0; otherwise, it is Mode 1. (As noted, RIF extensions must use required property elements, so Modes 2 and 3 are not available to them.)

Table 2: General *Tr* Mapping Table

<table>
<tr><th>Mode</th><th>Property Element XML Pattern (With Parent Shown)</th><th>RDF Triples added to Tr result</th></tr>
<tr><td>0</td><td>

```
. . .
<parent>
  . . .
  <child ordered="yes">
    item-1
    . . .
    item-n
  </child>
  . . .
</parent>
  . . .
```

</td><td>

```
focus_node rdf:type parent
focus_node prop ( id-1 . . . id-n )
triples-1
. . .
triples-n
where:
   <id-1, triples-1> = Tr(item-1)
   . . .
   <id-n, triples-n> = Tr(item-n)
```

</td></tr>
<tr><td>1</td><td>

```
. . .
<parent>
  . . .
  <child>item</child>
  . . .
</parent>
. . .
```

</td><td>

```
focus_node rdf:type parent
focus_node prop id
triples
where:
   <id, triples> = Tr(item)
```

As a special case, if item is merely character data (with no XML elements), id is an RDF plain literal with the same value and triples-n is empty. This occurs in RIF Core, for example, with the rif:location property element.

</td></tr>
<tr><td>2</td><td>

```
. . .
<parent>
  . . .
  <child>item_1</child>
  . . .
  <child>item_n</child>
  . . .
</parent>
. . .
```

</td><td>

```
focus_node rdf:type parent
focus_node prop ( id-1 . . . id-n )
triples-1
. . .
triples-n
where:
   <id-1, triples-1> = Tr(item-1)
   ...
   <id-n, triples-n> = Tr(item-n)
```

</td></tr>
<tr><td>3</td><td>

As found in <Atom> and <Expr> in BLD (but not Core or PRD):

```
. . .
<parent>
  . . .
  <slot ordered="yes">
    <Name>name-1</Name>
    value-1
  </slot>
   . . .
  <slot ordered="yes">
    <Name>name-n</Name>
    value-n
  </slot>
  . . .
<parent>
. . .
```

</td><td>

```
focus_node rdf:type parent
focus_node rif:namedargs (
       [ rdf:type rif:NamedArg;
         rif:argname "name-1";
         rif:argvalue value-id-1 ]
       . . .
       [ rdf:type rif:NamedArg;
         rif:argname "name-n";
         rif:argvalue value-id-n ] )
triples-1
. . .
triples-n
where:
   <value-id-1, triples-1> = Tr(value-1)
   . . .
   <value-id-n, triples-n> = Tr(value-n)
```

</td></tr>
<tr><td>3</td><td>

```
...
<Frame>
  . . .
  <slot ordered="yes">
    key-1
    value-1
  </slot>
   . . .
  <slot ordered="yes">
    key-n
    value-n
  </slot>
  . . .
</Frame>
...
```

</td><td>

```
focus_node rdf:type rif:Frame.
focus_node rif:slots (
   [ rdf:type rif:Slot; rif:slotkey nk-1; rif:slotvalue nv-1 ]
   . . .
   [ rdf:type rif:Slot; rif:slotkey nk-n; rif:slotvalue nv-n ] )
tk-1
. . .
tk-n
tv-1
. . .
tv-n
where:
   <nk-1, tk-1> = Tr(key-1)
   . . .
   <nk-n, tk-n> = Tr(key-n)
   <nv-1, tv-1> = Tr(value-1)
   . . .
   <nv-n, tv-n> = Tr(value-n)
```

</td></tr>
</table>

W3C Working Group Note

This table specifies exceptions to the default rules for determining the value of *prop* and the mode of the property element:

Table 3: Mapping from RIF Parent/Child XML Elements to RDF Properties

Class Element (parent)	Property Element (child)	RDF Property (*prop*)	Mode
Document	directive	rif:directives	2
Group	sentence	rif:sentences	2
Forall	declare	rif:vars	2
Exists	declare	rif:vars	2
And	formula	rif:formulas	2
Or	formula	rif:formulas	2
Frame	slot	rif:slots	3
Atom	slot	rif:namedargs	3
Expr	slot	rif:namedargs	3

6 The Reverse Mapping (Extracting RIF XML)

We call the inverse mapping *XTr*:

XTr(*focus-node*, *triples*) → *rif-xml-tree*

For any RIF-XML document *D*, valid according to some RIF dialect XML schema, given the transformation:

D' = *Xtr*(*Tr*(*D*))

D' and *D* will differ only in presentation aspects (non-significant whitespace, XML comments, IRIs being made absolute using xml:base, etc), unrelated to the semantics. That is, the semantics of D will be preserved so the rule sets corresponding to D and D' will have identical entailments under the relevant RIF Dialect semantics.

The ordering of child property elements by *Xtr* is determined by the relevant dialect schema, if known; otherwise they are lexicographic, sorted first by namespace then by local part. *Xtr* implementations *may* accept schema parameters to constrain their extraction to conform to a particular dialect's schema, and handle non-lexicographic ordering, like:

XTr(*focus-node*, *triples*, *XML-schema*, *XML-root-element-in-schema*) → *rif-xml-tree*

If they do not do this, they will not be able to correctly extract RIF XML documents in non-standard schemas using non-lexicographic element ordering. (For this reason, extensions are advised to use lexicographic ordering of elements in their schema.) For the three standard dialects, *Xtr* implementations *must* emit the children in schema-valid order.

7 Importing RIF into RDF

RIF RDF and OWL Compatibility [RIF RDF+OWL] defines the entailments of combinations (R, G) where R (a RIF rule set) includes an import of G (an RDF graph).

We hereby define an RDF predicate `rif:usedWithProfile` which enables an import to be specified from the graph G instead of from R. *This definition also appears in SPARQL 1.1 Entailment Regimes [SPARQL ER].*

In the simple usage the graph G is a plain RDF graph and `rif:usedWithProfile` is used to combine that graph with one or more externally defined RIF rule sets. In this usage each subject of a `rif:usedWithProfile` assertion should be the URI for a RIF rule set (which may be encoded in RIF-XML or RIF-in-RDF) and the object should be an import profile as defined in RIF RDF and OWL Compatibility [RIF RDF+OWL].

It is also possible for the graph G to itself contain an encoded ruleset along with additional RDF statements to which the ruleset is intended to apply. If graph G is obtained from a base IRI U_g then the statement:

U_g `rif:usedWithProfile P .`

within the graph causes it to be treated as both the graph G and the source of the rule set R in the combination (R, G). Syntactically such a statement can be made by using the empty relative IRI reference <> provided that the document base IRI has been set appropriately.

The semantics of `rif:usedWithProfile` is explained in the following section.

W3C Working Group Note

8 Semantics of RIF in RDF

Note: this definition also appears in SPARQL 1.1 Entailment Regimes [SPARQL ER].

A RIF-in-RDF-aware processor shall treat any RDF graph G as a RIF-RDF or RIF-OWL combination (cf. [RIF RDF+OWL]) as follows:

Let G' be the graph obtained from G by removing all triples with predicate `rif:usedWithProfile`.

Then G is to be treated by a RIF-in-RDF-aware processor as the ruleset R:

```
Document (
  Imports(R1')
  ...
  Imports(Rn')
  Imports(G' P1)
  ...
  Imports(G' Pn)
)
```

Where Ri and Pi are the subjects/objects respectively of triples of form:

```
Ri rif:usedWithProfile Pi
```

and Ri' denotes

- the RIF document Ri if Ri is a RIF/XML document, and
- the RIF document obtained from applying the inverse mapping XTr to the graph Gi if Ri denotes an RDF graph Gi.

Remark 1: Note that the fact that G' is treated as being imported with all profiles P1 ... Pn enforces G' to be treated according to the highest profiles among P1 ... Pn, cf. Section 5.2 of [RIF RDF+OWL].

Remark 2: If G also includes a `rif:usedWithProfile` statement referring to itself (i.e. with subject U_g where the graph G can be obtained from U_g) then the rules encoded in that document will be included in the rule imports but the encoding of the rules will remain visible within G'.

Remark 3: Note the discussion in the section 6 that the inversion of Tr is not a deterministic function.

Remark 4: Note that in the case where the graph G includes encoded RIF rules then, as a result of RDF graph merge, it may encode more than one RIF document. A RIF-in-RDF process MAY choose to combine all the rules in each documents into a single RIF document or MAY issue a warning. Note that future RIF dialects may have semantics which depend on rule ordering.

9 Acknowledgements

This document is the product of the Rules Interchange Format (RIF) Working Group (see below) whose members deserve recognition for their time and commitment. The editor extends special thanks to Dave Reynolds for his particularly attentive and insightful review comments.

The regular attendees at meetings of the Rule Interchange Format (RIF) Working Group at the time this document was developed were: Adrian Paschke (Freie Universitaet Berlin), Axel Polleres (DERI), Chris Welty (IBM), Christian de Sainte Marie (IBM), Dave Reynolds (HP), Gary Hallmark (ORACLE), Harold Boley (NRC), Jos de Bruijn (FUB), Leora Morgenstern (IBM), Michael Kifer (Stony Brook), Mike Dean (BBN), Sandro Hawke (W3C/MIT), and Stella Mitchell (IBM). [

10 References

10.1 Normative References

[RDF Concepts]
Resource Description Framework (RDF): Concepts and Abstract Syntax, G. Klyne, J. Carrol, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. Latest version available at http://www.w3.org/TR/rdf-concepts/.

[RIF Core]
RIF Core Dialect, Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres and Dave Reynolds (Editors), W3C Recommendation. Available at http://www.w3.org/TR/rif-core/.

[RIF RDF+OWL]
RIF RDF and OWL Compatibility, Jos de Bruijn (Editor), W3C Recommendation. Available at http://www.w3.org/TR/rif-rdf-owl/.

[Turtle]
Turtle - Terse RDF Triple Language, David Beckett and Tim Berners-Lee, Authors, W3C Team Submission 14 January 2008. Latest Version available at http://www.w3.org/TeamSubmission/turtle/ .

W3C Working Group Note

10.2 Nonnormative References

[GRDDL]
Gleaning Resource Descriptions from Dialects of Languages (GRDDL), Dan Connolly, Editors, W3C Recommendation, 11 September 2007, http://www.w3.org/TR/2007/REC-grddl-20070911/ . Latest version available at http://www.w3.org/TR/grddl/ .

[OWL2 Mapping]
OWL 2 Web Ontology Language: Mapping to RDF Graphs, Peter F. Patel-Schneider, Boris Motik, Eds., W3C Recommendation 27 October 2009, http://www.w3.org/TR/2009/REC-owl2-mapping-to-rdf-20091027/ . Latest version at http://www.w3.org/TR/owl-mapping-to-rdf .

[RDF Semantics]
RDF Semantics, Patrick Hayes, Ed., W3C Recommendation 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-mt-20040210/ . Latest version at http://www.w3.org/TR/rdf-mt/ .

[RDF Tools]
Semantic Web Development Tools, Website: http://www.w3.org/2001/sw/wiki/Tools retrieved on 21 June 2010.

[RDF XML]
RDF/XML Syntax Specification (Revised), Dave Beckett, Ed., W3C Recommendation 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/ . Latest version at http://www.w3.org/TR/rdf-syntax-grammar/ .

[RDFa]
RDFa in XHTML: Syntax and Processing, Ben Adida, Mark Birbeck, Shane McCarron, Steven Pemberton, Eds., W3C Recommendation 14 October 2008, http://www.w3.org/TR/2008/REC-rdfa-syntax-20081014/ . Latest version at http://www.w3.org/TR/rdfa-syntax/ .

[RDF]
Resource Description Framework (RDF), Website http://www.w3.org/RDF/ retrieved on 21 Jun 2010.

[RIF BLD]
RIF Basic Logic Dialect Harold Boley, Michael Kifer, eds. W3C Proposed Edited Recommendation, 11 December 2012, http://www.w3.org/TR/2012/PER-rif-bld-20121211/. Latest version available at http://www.w3.org/TR/rif-bld/.

[RIF Charter]
Rule Interchange Format Working Group Charter, Sandro Hawke, Ed., http://www.w3.org/2005/rules/wg/charter .

[RIF FLD]
RIF Framework for Logic Dialects Harold Boley, Michael Kifer, eds. W3C Proposed Edited Recommendation, 11 December 2012, http://www.w3.org/TR/2012/PER-rif-fld-20121211/. Latest version available at http://www.w3.org/TR/rif-fld/.

[RIF Overview]
RIF Overview Michael Kifer, Harold Boley, eds. W3C Working Group Note, 11 December 2012, http://www.w3.org/TR/2012/NOTE-rif-overview-20121211/. Latest version available at http://www.w3.org/TR/rif-overview/.

[RIF PRD]
RIF Production Rule Dialect Christian de Sainte Marie, Gary Hallmark, Adrian Paschke, eds. W3C Proposed Edited Recommendation, 11 December 2012, http://www.w3.org/TR/2012/PER-rif-prd-20121211/. Latest version available at http://www.w3.org/TR/rif-prd/.

[SPARQL ER]
SPARQL 1.1 Entailment Regimes, Birte Glimm and Chimezie Ogbuji, Editors, W3C Working Draft, 12 May 2011, http://www.w3.org/TR/2011/WD-sparql11-entailment-20110512. Latest version at http://www.w3.org/TR/sparql11-entailment/.

11 Appendix Complete Example

Examples, test cases, and links to implementations may be found at http://www.w3.org/2011/rif-in-rdf.

Here is Example 8 from BLD converted via the RIF-in-RDF mapping to Turtle:

```
@prefix : <http://www.w3.org/2007/rif#> .
@prefix xs: <http://www.w3.org/2001/XMLSchema#> .

[]      a :Document;
        :directives ();
        :payload <http://sample.org>.

<http://sample.org>     a :Group;
     :meta  [
         a :Frame;
         :object  [
             a :Const;
             :constname "pd" ];
         :slots  (
         [
                 a :Slot;
                 :slotkey  [
                     a :Const;
                     :constIRI "http://purl.org/dc/terms/publisher"^^xs:anyURI ];
                 :slotvalue  [
                     a :Const;
                     :constIRI "http://www.w3.org/"^^xs:anyURI ] ]
         [
                 a :Slot;
                 :slotkey  [
```

```
                     a :Const;
                     :constIRI "http://purl.org/dc/terms/date"^^xs:anyURI ];
                 :slotvalue  [
                     a :Const;
                     :value "2008-04-04"^^xs:date ] ] ) ];
     :sentences  (
     [
             a :Forall;
             :formula  [
                 a :Implies;
                 :if  [
                     a :And;
                     :formulas  (
                     [
                             a :Atom;
                             :args  (
                             [
                                     a :Var;
                                     :varname "item" ] );
                             :op  [
                                 a :Const;
                                 :constIRI "http://example.com/concepts#perishable"^^xs:anyURI ] ]
                     [
                             a :Atom;
                             :args  (
                             [
                                     a :Var;
                                     :varname "item" ]
                             [
                                     a :Var;
                                     :varname "deliverydate" ]
                             [
                                     a :Const;
                                     :constIRI "http://example.com/John"^^xs:anyURI ] );
                             :op  [
                                 a :Const;
                                 :constIRI "http://example.com/concepts#delivered"^^xs:anyURI ] ]
                     [
                             a :Atom;
                             :args  (
                             [
                                     a :Var;
                                     :varname "item" ]
                             [
                                     a :Var;
                                     :varname "scheduledate" ] );
                             :op  [
                                 a :Const;
                                 :constIRI "http://example.com/concepts#scheduled"^^xs:anyURI ] ]
                     [
                             a :Equal;
                             :left  [
                                 a :Var;
                                 :varname "diffduration" ];
                             :right  [
                                 a :External;
                                 :content  [
                                     a :Expr;
                                     :args  (
                                     [
                                             a :Var;
                                             :varname "deliverydate" ]
                                     [
                                             a :Var;
                                             :varname "scheduledate" ] );
                                     :op  [
                                         a :Const;
:constIRI "http://www.w3.org/2007/rif-builtin-function#subtract-dateTimes"^^xs:anyURI ] ] ] ]
                     [
                             a :Equal;
                             :left  [
                                 a :Var;
                                 :varname "diffdays" ];
                             :right  [
                                 a :External;
```

```
                              :content  [
                                  a :Expr;
                                  :args  (
                                  [
                                          a :Var;
                                          :varname "diffduration" ] );
                                  :op  [
                                      a :Const;
:constIRI "http://www.w3.org/2007/rif-builtin-function#days-from-duration"^^xs:anyURI ] ] ] ]
                      [
                              a :External;
                              :content  [
                                  a :Atom;
                                  :args  (
                                  [
                                          a :Var;
                                          :varname "diffdays" ]
                                  [
                                          a :Const;
                                          :value 10 ] );
                                  :op  [
                                      a :Const;
:constIRI "http://www.w3.org/2007/rif-builtin-predicate#numeric-greater-than"^^xs:anyURI ] ] ] ) ];
                  :then  [
                      a :Atom;
                      :args  (
                      [
                              a :Const;
                              :constIRI "http://example.com/John"^^xs:anyURI ]
                      [
                              a :Var;
                              :varname "item" ] );
                      :op  [
                          a :Const;
                          :constIRI "http://example.com/concepts#reject"^^xs:anyURI ] ] ];
              :vars  (
              [
                      a :Var;
                      :varname "item" ]
              [
                      a :Var;
                      :varname "deliverydate" ]
              [
                      a :Var;
                      :varname "scheduledate" ]
              [
                      a :Var;
                      :varname "diffduration" ]
              [
                      a :Var;
                      :varname "diffdays" ] ) ]
      [
              a :Forall;
              :formula  [
                  a :Implies;
                  :if  [
                      a :Atom;
                      :args  (
                      [
                              a :Var;
                              :varname "item" ] );
                      :op  [
                          a :Const;
                          :constIRI "http://example.com/concepts#unsolicited"^^xs:anyURI ] ];
                  :then  [
                      a :Atom;
                      :args  (
                      [
                              a :Const;
                              :constIRI "http://example.com/Fred"^^xs:anyURI ]
                      [
                              a :Var;
                              :varname "item" ] );
                      :op  [
                          a :Const;
```

W3C Working Group Note

```
            :constIRI "http://example.com/concepts#reject"^^xs:anyURI ] ] ];
:vars  (
[
        a :Var;
        :varname "item" ] ) ] ) .
```

12 Changelog

12.1 Changes since 12 May 2011

Reference were updated as part of publishing RIF Second Edition

12.2 Changes since 22 June 2010

The mapping was changed in several ways. In particular, rdf:type arcs are now used, and some properties were renamed to be closer to the RIF/XML names.

Two sections were added to define a mechanism for importing RIF documents into RDF documents, using a rif:usedWithProfile property.

Placeholder appendices were removed because we did not develop the anticipated extra materials.