Abstract

Rules interchanged using the Rule Interchange Format RIF may depend on or be used in combination with RDF data and RDF Schema or OWL ontologies. This document, developed by the Rule Interchange Format (RIF) Working Group, specifies the interoperation between RIF and the data and ontology languages RDF, RDF Schema, and OWL.

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May Be Superseded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 (this document)
8. OWL 2 RL in RIF
9. RIF Combination with XML data
10. RIF in RDF
11. RIF Test Cases
12. RIF Primer

Summary of Changes

There have been no substantive changes since the previous version. For details on the minor changes see the change log and color-coded diff.

W3C Members Please Review By 8 January 2013

The W3C Director seeks review and feedback from W3C Advisory Committee representatives, via their review form by 8 January 2013. This will allow the Director to assess consensus and determine whether to issue this document as a W3C Edited Recommendation.

Others are encouraged by the Rule Interchange Format (RIF) Working Group to continue to send reports of implementation experience, and other feedback, to public-rif-comments@w3.org (public archive). Reports of any success or difficulty with the test cases are encouraged. Open discussion among developers is welcome at public-rif-dev@w3.org (public archive).

No Endorsement

Publication as a Proposed Edited Recommendation 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.
1 Overview of RDF and OWL Compatibility

The Rule Interchange Format (RIF) is a format for interchanging rules over the Web. Rules that are exchanged using RIF may refer to external data sources and may be based on data models that are represented using a language different from RIF. The Resource Description Framework RDF [RDF-Semantics] is a Web-based language for the representation and exchange of data; RDF Schema (RDFS) [RDF-Semantics] and the OWL Web Ontology Language [OWL2-Syntax] are Web-based languages for representing and exchanging ontologies. This document specifies how combinations of RIF documents and RDF data and/or RDFS and OWL ontologies are interpreted; i.e., it specifies how RIF interoperates with RDF, RDFS, and OWL.

We consider here OWL 2 [OWL2-Syntax], which is an extension of OWL 1 [OWL References]. Therefore, the notions defined in this document also apply to combinations of RIF documents with OWL 1 ontologies.

We consider here the RIF Basic Logic Dialect (BLD) [RIF-BLD] and RIF Core [RIF-Core], a subset of RIF BLD. The RIF Production Rule Dialect (PRD) [RIF-PRD] is an extension of RIF Core. Interoperability between RIF and RDF/OWL is only defined for the Core subset of PRD. In the remainder, when speaking about RIF documents and rules, we refer to RIF Core and BLD.

RDF data and RDFS and OWL ontologies can be represented using RDF graphs. There exist several alternative syntaxes for OWL ontologies; however, for exchange purposes it is assumed they are represented using RDF graphs.

Several syntaxes have been proposed for the exchange of RDF graphs, the normative syntax being RDF/XML [RDF-Syntax]. RIF does not provide a format for exchanging RIF graphs; it is assumed that RDF graphs are exchanged using RDF/XML, or any other syntax that can be used for representing or exchanging RDF graphs.

A typical scenario for the use of RIF with RDF/OWL is the exchange of RDF graphs and/or RDFS and OWL ontologies: an interchange partner A has a rules language that is RDF/OWL-aware, i.e., it supports the use of RDF data, it uses an RDFS or OWL ontology, or it extends RDF(S)/OWL. A sends its rules using RIF, possibly with references to the appropriate RDF graph(s), to partner B. B receives the rules and references the referenced RDF graph(s). The rules are translated to the internal rules language of B and are processed, together with the RDF graphs, using the RDF/OWL-aware rule engine of B. The use case Vocabulary Mapping for Data Integration [RIF-UCR] is an example of the interchange of RIF rules that use RDF data and RDFS ontologies.

A specialization of this scenario is the publication of RIF rules that refer to RDF graphs; publication is a special kind of interchange: one to many, rather than one-to-one. When a rule publisher A publishes its rules on the Web, there may be several consumers that retrieve the RIF rules and RDF graphs from the Web. The RIF rules to their respective rules languages, and process them together with the RDF graphs in their own rule engines. The use case Publishing Rules for Interlinked Metadata [RIF-UCR] illustrates the publication scenario.

Another specialization of the exchange scenario is the Interchange of Rule Extensions to OWL (RIF-UCR). The intention of the rule publisher in this scenario is to extend an OWL ontology with rules: interchange partner A has a rules language that extends OWL. A splits its ontology+rules description into a separate OWL ontology and a RIF document, publishes the OWL ontology, and sends (or publishes) the RIF document, which includes a reference to the OWL ontology. A consumer of the rules retrieves the OWL ontology and translates the ontology and rules description into a combined ontology+rules description in its own rule extension of OWL.

A RIF document that refers to (imports) RDF graphs and/or RDFS/OWL ontologies, or any use of a RIF document with RDF graphs, is viewed as a combination of a document and a number of graphs and ontologies. This document specifies how, in such a combination, the document and the graphs and ontologies interoperate in a technical sense, i.e., the conditions under which the combination is satisfiable (i.e., consistent), as well as the entailments (i.e., logical consequences) of the combination. The interaction between RIF and RDF/OWL is realized by connecting the model theory of RIF [RIF-BLD] with the model theories of RDF [RDF-Semantics] and OWL [OWL2-Semantics], respectively.

The notation of certain symbols in RIF, particularly IRIs and plain literals, is slightly different from the notation in RDF/OWL. These differences are illustrated in the Section Symbols in RIF Versus RDF/OWL.

The RDF Semantics specification [RDF-Semantics] defines four normative notions of entailment for RDF graphs: Simple, RDF, RDFS, and Datatype entailment. OWL 2 specifies two different semantics, with corresponding notions of entailment: the Direct Semantics [OWL2-Semantics] and the RDF-Based Semantics [OWL2-RDF-BasedSemantics]. This document specifies the interaction between RIF and RDF/OWL for all six notions. The Section RDF Compatibility is concerned with the combination of RIF and RDF/OWDF. The combination of RIF and OWL is addressed in the Section RDF Compatibility. The semantics of the interaction between RIF and the OWL 2 Direct Semantics is close in spirit to [SWRL].

RIF provides a mechanism for referring to (importing) RDF graphs and a means for specifying the profile of this import, which corresponds to the intended entailment regime. The Section Importing RDF and OWL in RIF specifies how such import statements are used for representing RIF-RDF and RIF-OWL combinations.

The Appendix Embeddings (Informative) describes how reasoning with combinations of RIF rules with RDF and OWL 2 RL (a subset of OWL 2 DL) can be reduced to reasoning with RDF documents. This reduction can be seen as an implementation hint for interchange partners who do not have RDF/OWL-aware rule systems, but want to process RIF rules that import RDF graphs and OWL ontologies. In terms of the aforementioned scenario: if the interchange partner B does not have an RDF/OWL-aware rule system, but B can process RIF rules, then the appendix explains how the rule system of B could be used for processing RIF-RDF/OWL combinations.

Throughout this document the following conventions are used when writing RIF and RDF statements in examples and definitions.
All RIF statements are written using the RIF presentation syntax [RIF-BLD]. Where possible, this document uses the shortcut syntax for IRIs and strings as defined in [RIF-DTB]. RDF triples are written using the Turtle syntax [Turtle]: triples are written as \( s \ p \ o \), where \( s \), \( p \), and \( o \) are blank nodes; in RDF, IRIs delimited with '<' and '>', compact IRIs prefix:localname, plain literals without language tags "literal", plain literals with language tags "literal@lang", or typed literals [literal^^datatype-IRI]. The following namespace prefixes are used throughout this document: ex refers to http://example.org/example, xs refers to http://www.w3.org/2001/XMLSchema, rdf refers to http://www.w3.org/1999/02/22-rdf-syntax-ns#, rdfs refers to http://www.w3.org/2000/01/rdf-schema#, owl refers to http://www.w3.org/2002/07/owl#, and rif refers to http://www.w3.org/2007/rif#.

2 Symbols in RIF versus RDF/OWL (Informative)

Where RDF/OWL has four kinds of constants: URI references (i.e., IRIs), plain literals without language tags, plain literals with language tags and typed literals (i.e., Unicode sequences with datatype IRIs [RDF-Concepts]), RIF has one kind of constants: Unicode sequences with symbol space IRIs [RIF-DTB]. Symbol spaces can be seen as groups of constants. Every datatype is a symbol space, but there are symbol spaces that are not datatypes, for example the symbol space rif:iri groups all IRIs. The correspondence between constant symbols in RDF graphs and RIF documents is explained in Table 1.

<table>
<thead>
<tr>
<th>RDF Symbol</th>
<th>Example</th>
<th>RIF Symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain literal without language tag</td>
<td>&quot;literal string&quot;</td>
<td>Constant in the rdf:PlainLiteral symbol space</td>
<td>&quot;literal string&quot;^^rdf:PlainLiteral</td>
</tr>
<tr>
<td>Plain literal with language tag</td>
<td>&quot;literal string@en&quot;</td>
<td>Constant in the rdf:PlainLiteral symbol space</td>
<td>&quot;literal string@en&quot;^^rdf:PlainLiteral</td>
</tr>
<tr>
<td>Typed literal</td>
<td>&quot;1&quot;^^xs:integer</td>
<td>Constant with symbol space</td>
<td>&quot;1&quot;^^xs:integer</td>
</tr>
</tbody>
</table>

The shortcut syntax for IRIs and strings [RIF-DTB], used throughout this document, corresponds to the syntax for IRIs and plain literals in Turtle [Turtle], a commonly used syntax for RDF.

- IRIs, i.e., constants of the form "IRI"^^rif:iri, may be written as <IRI> or as compact IRIs [CURIE], i.e., as prefix:localname, where prefix is understood to refer to an IRI namespace [IRI] and prefix:localname stands for the IRI [IRI] obtained by concatenating names prefix and localname.
- Plain literals without language tags, i.e., constants of the form "my string"^^"" rdf:PlainLiteral may be written as "my string".

RIF does not have a notion corresponding exactly to RDF blank nodes. RIF local symbols, written symbolname have some commonality with blank nodes; like the blank node label, the name of a local symbol is not exposed outside of the document. However, in contrast to blank nodes, which are essentially existentially quantified variables, RIF local symbols are constant symbols. In many applications and deployment scenarios, this difference may be inconsequential. However the results will differ when such symbols are used in a non-assertional context, such as in a query pattern or rule body.

Finally, variables in the bodies of RIF rules or in query patterns may be existentially quantified, and are thus similar to blank nodes; however, RIF BLD does not allow existentially quantified variables to occur in rule heads.

3 RDF Compatibility

This section specifies how a RIF document interacts with a set of RDF graphs in a RIF-RDF combination. In other words, how rules can "access" data in the RDF graphs. There is a correspondence between statements in RDF graphs and certain kinds of formulas in RIF. Namely, there is a correspondence between RDF triples of the form \( s \ p \ o \) and RIF frame formulas of the form \( \exists \ p' \rightarrow \ o' \). Under such a correspondence, the RDF graph consisting of all triples \( \exists \ p' \rightarrow \ o' \) is satisfied, and vice versa.

Consider, for example, a combination of an RDF graph that contains the triples

\[
\text{ex:john ex:brotherOf ex:jack.} \\
\text{ex:jack ex:parentOf ex:mary.}
\]

saying that ex:john is a brother of ex:jack and ex:jack is a parent of ex:mary, and a RIF document that contains the rule

\[
\text{forall } x \ y \ z \ (x[ex:uncleOf \rightarrow \ y] \land x[ex:hasName \rightarrow \ y]) \\
\]

which says that whenever some \( x \) is the brother of some \( y \) and \( y \) is a parent of some \( z \), then \( x \) is an uncle of \( z \). This combination will satisfy the RIF rule:if \( x \) is the brother of \( y \) and \( y \) is the hasName of \( z \), then \( x \) is the uncle of \( z \).

Note that blank nodes cannot be referenced directly from RIF rules, since blank nodes are local to a specific RDF graph. Variables in RIF rules do, however, range over objects denoted by blank nodes. So, it is possible to "access" an object denoted by a blank node from a RIF rule using a variable in a rule body.

The following example illustrates the interaction between RDF and RIF in the face of blank nodes.

Consider a combination of an RDF graph that contains the triple

\[
_\text{i} \text{x hasName "John".}
\]

saying that there is something, denoted here by a blank node, which has the name "John", and a RIF document that contains the rules

\[
\text{forall } x \ y \ (x[rdf:type \rightarrow \ ex:named] \land x[ex:hasName \rightarrow \ y]) \\
\text{forall } x \ y \ (x[rdf:type \rightarrow \ http://a] \land x[ex:hasName \rightarrow \ y])
\]

which says that whenever there is some \( x \) that has some name \( y \), then \( x \) is of type ex:named and http://a has a property http://p with value \( y \).

From this combination the following RIF condition formulas can be derived:

\[
\text{exists } x \ [x[rdf:type \rightarrow \ ex:named] \\
\text{<http://a>[http://p] \text{ "John"}]
\]

as can the following RDF triples:

\[
_\text{y rdf:type ex:named.} \\
\text{<http://a> "John".}
\]

However, there is no RIF constant symbol \( t \) such that \( t[rdf:type \rightarrow \ ex:named] \) can be derived, because there is no constant that represents the named individual.

Note that, even when considering Simple entailment, not every combination is satisfiable. In fact, not every RIF document has a model. For example, the RIF BLD document consisting of the fact

\[
\text{"a"="b"}
\]

does not have a model, since the symbols "a" and "b" are mapped to (the distinct) character strings "a" and "b", respectively, in every semantic structure.
3.1 Syntax of RIF-RDF Combinations

This section first reviews the definitions of RDF Vocabularies and RDF graphs, after which RIF-RDF combinations are formally defined. The section concludes with a review of definitions related to datatypes and typed literals.

### 3.1.1 RDF Vocabularies and Graphs

An RDF Vocabulary \( V \) consists of the following sets of names:

- \( \text{URIs} V_\text{V} \), (corresponds to the Concepts and Abstract Syntax term "RDF URIs references"; see the End note on RDF URI references)
- \( \text{plain literals} V_\text{PL} \), (i.e., character strings with an optional language tag), and
- \( \text{typed literals} V_\text{TL} \), (i.e., pairs of character strings and datatype IRIs).

In addition, there is an infinite set of blank nodes, which is disjoint from the sets of names. See RDF Concepts and Abstract Syntax [RDF-Concepts] for precise definitions of these concepts.

**Definition.** Given an RDF Vocabulary \( V \), a **generalized RDF triple** of \( V \) is a statement of the form \( s p o \), where \( s, p \) and \( o \) are names in \( V \) or blank nodes.

**Definition.** Given an RDF Vocabulary \( V \), a **generalized RDF graph** is a set of generalized RDF triples of \( V \).

(See the [End note on generalized RDF graphs](#).)

### 3.1.2 RIF-RDF Combinations

A RIF-RDF combination consists of a RIF document and zero or more RDF graphs. Formally:

**Definition.** A **RIF-RDF combination** is a pair \( \langle R, S \rangle \), where \( R \) is a RIF document and \( S \) is a set of generalized RDF graphs of a Vocabulary \( V \).

When clear from the context, RIF-RDF combinations are referred to simply as combinations.

### 3.1.3 Datatypes and Typed Literals

Even though RDF allows the use of arbitrary datatype IRIs in typed literals, not all such datatype IRIs are recognized in the semantics. In fact, Simple entailment does not recognize any datatype and RDF and RDFS entailment recognize only the datatypes \( \text{xsd:string} \) and \( \text{xsd:integer} \). To facilitate discussing datatypes, and specifically datatypes supported in specific contexts (required for RIF-D-entailment), the notion of datatype maps [RDF-Semantics] is used.

A **datatype map** is a partial mapping from IRIs to datatypes.

RDFS, specifically RIF-D-entailment, allows the use of arbitrary datatype maps, as long as rdf:XMLLiteral is in the domain of the map. RIF BLD requires a number of additional datatypes to be included; these are the **RIF-required datatypes** [RIF-DB].

When checking consistency of a combination \( \langle R, S \rangle \) or entailment of a graph \( S \) or RIF formula \( \phi \) by a combination \( \langle R, S \rangle \), the set of **considered datatypes** is the union of the set of RIF-required datatypes and the sets of datatypes referenced in \( R \), the documents imported into \( R \), and \( \phi \) (when considering entailment of \( \phi \)).

**Definition.** Let \( DTS \) be a set of datatypes. A datatype map \( D \) is **conforming** with \( DTS \) if it satisfies the following conditions:

1. Every IRI identifying a datatype in \( DTS \) is in the domain of \( D \).
2. \( D \) maps each IRI in its domain to the datatype identified by that IRI in DTS.
Note that it follows from the definition that every datatype used in the RIF document in the combination or the entailed RIF formula (when considering entailment questions) is included in any datatype map conforming to the set of considered datatypes. There may be datatypes used in an RDF graph in the combination that are not included in such a datatype map.

**Definition.** Given a datatype map \( D \), a typed literal \((s, d)\) is a well-typed literal if

1. \( d \) is in the domain of \( D \) and \( s \) is in the lexical space of \( D(d) \)
2. \( d \) is the IRI of a symbol space required by RIF BLD and \( s \) is in the lexical space of the symbol space.

### 3.2. Semantics of RIF-RDF Combinations

The semantics of RIF-RDF combinations is defined through a combination of the RIF and RDF model theories, using a notion of common models. These models are then used to define satisfiability and entailment in the usual way. Combined entailment entails both entailment in RIF and entailment in RDF.

The RDF Semantics document [RDF-Semantics] defines four normative kinds of interpretations, as well as corresponding notions of satisfiability and entailment:

- **Simple interpretations**, which do not impose any conditions on the RDF and RDFS Vocabularies,
- **RDFS interpretations**, which impose additional conditions on the interpretation of the RDFS Vocabulary,
- **RDF interpretations**, which impose additional conditions on the interpretation of the RDF and RDFS Vocabularies, and
- **D-interpreta
tions**, which impose additional conditions on the treatment of datatypes, relative to a datatype map \( D \).

Those four types of interpretations are reflected in the definitions of satisfaction and entailment in this section.

### 3.2.1. Interpre
tations

This section defines the notion of common-RIF-RDF-interpretation, which is an interpretation of a RIF-RDF combination. This common-RIF-RDF-interpretation is the basis for the definitions of satisfaction and entailment in the following sections.

The correspondence between RDF semantic structures (interpretations) and RDF interpretations is defined through a number of conditions that ensure the correspondence in the interpretation of names (i.e., IRIs and literals) and formulas, i.e., the correspondence between RDF triples of the form \( s \ p \ o \) and RIF frames of the form \( s' [p' -> o'] \), where \( s', p', \) and \( o' \) are RIF symbols corresponding to the RDF symbols \( s, p, \) and \( o, \) respectively (cf. the Section Symbols in RIF Versus RDFS).

#### 3.2.1.1 RDF and RDF Interpretations

The notions of RDF interpretation and RIF semantic structure (interpretation) are briefly reviewed below.

As defined in [RDF-Semantics], a Simple interpretation \( I \) of a Vocabulary \( V \) is a tuple \( I = \langle IR, IP, IEXT, IS, IL, LV \rangle \), where

- \( IR \) is a non-empty set of resources (the domain),
- \( IP \) is a set of properties,
- \( IEXT \) is an extension function, which is a mapping from RDF into the power set of \( IR \times IR \)
- \( IS \) is a mapping from IRIs in \( V \) into \( IR \) (union \( IP \)),
- \( IL \) is a mapping from typed literals in \( V \) into \( IR \), and
- \( LV \) is the set of literal values, which is a subset of \( IR \), and includes all plain literals in \( V \).

RDF-, RDFS-, and D-interpreta
tions are Simple interpretations that satisfy certain conditions:

- A Simple interpretation \( I \) of a Vocabulary \( V \) is an RDF-interpretation if \( V \) includes the RDF Vocabulary and \( I \) satisfies the RDF axiomatic triples and the RDF semantic conditions.
- An RDF-interpretation \( I \) of a Vocabulary \( V \) is an RDFS-interpretation if \( V \) includes the RDFS Vocabulary and \( I \) satisfies the RDFS axiomatic triples and the RDFS semantic conditions.
- A Simple interpretation \( I \) of a Vocabulary \( V \) is a D-interpretation if \( V \) includes the IRIs in the domain of \( D \) and \( I \) satisfies the general semantic conditions for datatypes for every pair \( <d, D(d)> \) such that \( d \) is in the domain of \( D \).

As defined in [RIF-BLD], a semantic structure \( I \) is a tuple of the form \( \langle TV, DTS, D, DInd, DEq, IC, Ix, IR, IList, IIm, DTypes, IVar, IConst, IS', IStmt, ITruth \rangle \). The specification of RIF-RDF compatibility is only concerned with \( DTS, D, IC, Ix, IR, IList, IIm, DTypes, IVar, IConst \). The other mappings that are parts of a semantic structure are not used in the definition of combinations.

Recall that Const is the set of constant symbols and \( Var \) is the set of variable symbols in RIF.

- \( DTS \) is the set of datatypes, which have associated datatype identifiers,
- \( D \) is a set (the domain),
- \( DInd \) is a non-empty subset of \( D \),
- \( DEq \) is a non-empty subset of \( D \),
- \( IC \) is a mapping from constants to \( D \) such that constants in individual position are mapped to \( DInd \) and constants in function positions are mapped to \( DEq \),
- \( Ix \) is a mapping from \( Var \) to \( DEq \),
- \( IList \) is an injective mapping from \( DEq \) to \( DInd \),
- \( IR \) is a mapping from \( DTS \) \( \times D \) to \( DInd \),
- \( IIm \) is a mapping from constants to \( D \) such that \( IIm(\text{iri}) \) is a RIF symbol corresponding to \( \text{iri} \),
- \( DTypes \) is a mapping from \( D \) to \( TV \),
- \( IVar \) is a mapping from \( D \) to \( TV \),
- \( IConst \) is a mapping from \( D \) to \( TV \),
- \( IIm \) is a mapping from \( D \) to \( TV \),
- \( IStmt \) is a mapping from \( D \) to \( TV \),
- \( ITruth \) is a mapping from \( D \) to \( TV \).

For the purpose of the interpretation of imported documents, RIF BLD defines the notion of semantic multi-structures, which are nonempty sets of semantic structures of the form \( \{ \langle J_1, F_1, I_1 \rangle, \ldots, \langle J_n, F_n, I_n \rangle \} \) that differ only in interpretation of local constants. The structure \( I \) in the above is used to interpret document formulas, and will be used to specify RIF combinations.

#### 3.2.2 RDF Lists

Syntactically speaking, an RDF list is a set of triples of the form

\[
\begin{align*}
i_1 &\ rdf:first d_1, \\
i_1 &\ rdf:rest i_2, \\
\vdots &\\
i_n &\ rdf:rest \text{nil}.
\end{align*}
\]

Here, \( i_1 \ldots i_n \) provide the structure of the linked list and \( d_1 \ldots d_n \) are the items. The above list would be written in RIF syntax as \( \text{List}(d_1 \ldots d_n) \).

Given an RDF interpretation \( I = \langle IR, IP, IEXT, IS, IL, LV \rangle \), we say that an element \( i_1 \in IR \) refers to an RDF list \( \langle y_1, \ldots, y_n \rangle \) if \( i_1 = \text{IS}(\text{nil}) \), in case \( n=0 \); otherwise, \( i_1 y_1 \ldots i_1 y_n \) in such that \( i_1 y_1, \ldots, i_1 y_n \in IEXT(\text{IS}(\text{rest})) \), and \( \langle i_n, y_n \rangle \in IEXT(\text{IS}(\text{rest})) \), and \( \langle i_n, \text{IS}(\text{rest}) \rangle \in IEXT(\text{IS}(\text{rest})) \).

Note that, if \( n > 0 \), there may be several lists referred to by \( i_1 \), since there is no restriction, in general, on the \( \text{rdf:first} \) and \( \text{rdf:rest} \) successors.
3.2.1.3 Common RIF-RDF Interpretations

Definition. A common-RIF-RDF-interpretation \( I \) is a pair \((\mathcal{I},I)\), where \( \mathcal{I} \) is a semantic multi-structure of the form \((\mathcal{J},I^1,I^2,...)\), and \( I \) is a Simple interpretation of a Vocabulary \( \mathcal{V} \), such that the following conditions hold:

1. (IR union UP) = \( \mathcal{D} \mathcal{U} \mathcal{D} \).
2. \( \mathcal{P} \) is a superset of the set of all \( \mathcal{D} \mathcal{U} \mathcal{D} \) such that there exist some a, b in \( \mathcal{D} \mathcal{U} \mathcal{D} \) such that \( h_u[I\mathcal{U}\mathcal{D}](a,b) = t \).
3. Let \( \mathcal{D} \mathcal{U} \mathcal{D} \) be a superset of the union of the value spaces of all considered data types.
4. \( \text{EXT}(I)(k) \) is the set of all pairs \( (a,b) \) with \( a, b \) in \( \mathcal{D} \mathcal{U} \mathcal{D} \) such that \( h_u[I\mathcal{U}\mathcal{D}](a,b) = t \).
5. \( IS(\mathcal{I}) = \{c|c(\mathcal{I})\} \) for every IRIF in \( \mathcal{V}_\mathcal{I} \).
6. \( IL(\mathcal{I},d) = I_l[\mathcal{I}(\mathcal{I})] \) for every well-typed literal \( (d, I) \in \mathcal{V}_\mathcal{I} \).
7. \( IEXT(\mathcal{I}(\mathcal{I})(\mathcal{D} \mathcal{R} \mathcal{F}) \text{-type}) = \mathcal{E} \). const\( (\mathcal{I}) \) for every set \( \mathcal{E} \) of all considered data types.
8. \( IEXT(\mathcal{I}(\mathcal{I})(\mathcal{D} \mathcal{R} \mathcal{F}) \text{-type}) = \mathcal{E} \). const\( (\mathcal{I}) \) for every set \( \mathcal{E} \) of all considered data types.
9. \( IEXT(\mathcal{I}(\mathcal{I})(\mathcal{D} \mathcal{R} \mathcal{F}) \text{-type}) = \mathcal{E} \). const\( (\mathcal{I}) \) for every set \( \mathcal{E} \) of all considered data types.
10. \( IEXT(\mathcal{I}(\mathcal{I})(\mathcal{D} \mathcal{R} \mathcal{F}) \text{-type}) = \mathcal{E} \). const\( (\mathcal{I}) \) for every set \( \mathcal{E} \) of all considered data types.

Condition 1 ensures that the combination of resources and properties corresponds exactly to the RDF domain; note that if \( \mathcal{I} \) is an RDF-, RDFS-, or DInterpretation, \( \mathcal{P} \) is a subset of IR, and thus \( \mathcal{I} = \mathcal{D} \mathcal{U} \mathcal{D} \).
Condition 2 ensures that the set of RDF properties at least includes all elements that are used as properties in frames in the RDF domain.
Condition 3 ensures that all concrete values in \( \mathcal{D} \mathcal{U} \mathcal{D} \) are included in \( \mathcal{V}_\mathcal{I} \). (by definition, the value spaces of all considered data types are included in \( \mathcal{D} \mathcal{U} \mathcal{D} \). Condition 4 ensures that RDF triples are interpreted in the same way as frame formulas. Condition 5 ensures that IRIs are interpreted in the same way. Condition 6 ensures that typed literals are interpreted in the same way. Note that no correspondences are defined for the mapping of names in RDF that are not symbols of RIF, e.g., ill-typed literals and RDF URI references that are not absolute IRIs. Condition 7 ensures that typing in RDF and typing in RIF correspond, i.e., a \( \mathcal{RDF} \text{-type} \) is true if a \( \mathcal{I}_\mathcal{R} \text{-type} \) is true. Condition 8 ensures that whenever a RDF subclass statement holds, the corresponding RIF subclass statement holds as well, i.e., a \( \mathcal{RDF} \text{-subclass} \) is true if a \( \mathcal{I}_\mathcal{R} \text{-subclass} \) is true. Finally, condition 9 requires the existence of an RDF list for every RDF list and condition 10 in addition requires the existence of an RDF list for every RDF list.

3.2.2 Satisfaction and Models

The notion of satisfiability refers to the conditions under which a common-RIF-RDF-interpretation \((\mathcal{I},I)\) is a model of a combination \( C, S \). The notion of satisfiability is defined for all four entailment regimes of RDF (i.e., Simple, RDF, RDFS, and D). The definitions are all analogous. Intuitively, a common-RIF-RDF-interpretation \((\mathcal{I},I)\) satisfies a combination \( C, S \) if \( \mathcal{I} \) is a model of \( R \) and \( I \) satisfies \( S \). Formally:

Definition. A common-RIF-RDF-interpretation \((\mathcal{I},I)\) satisfies a RIF-RDF combination \( C=\langle R, S \rangle \) if \( \mathcal{I} \) is a model of \( R \) and \( I \) satisfies every RDF graph \( S \) in \( S \); in this case \((\mathcal{I},I)\) is called a RIF-Simple-model, or \( \mathcal{C} \), and \( I \) is a satisfiable model of \( C \). \( \mathcal{I} \) satisfies a generalized RDF graph \( S \) if \( I \) satisfies every RDF graph \( S \) in \( S \); in this case \( \mathcal{I} \) is called a RIF-D-model, or \( \mathcal{C} \), and \( I \) is a satisfiable model of \( C \).

Let \((\mathcal{I},I)\) be a model of a combination \( C \) and let \( D \) be a datatype map conforming with the set \( DTS \) of datatypes in \( \mathcal{I} \). \( (\mathcal{I},I) \) is a RIF-D-model of \( C \) if \( I \) is a \( \mathcal{D} \)-interpretation; in this case \( C \) is RIF-D-satisfiable.

The other notions of entailment are defined analogously:

Definition. A combination \( C \) is RIF-Simple-entails \( S \) (resp., \( \varphi \)) if every Simple model of \( C \) satisfies \( S \) (resp., \( \varphi \)).
A combination \( C \) is RIF-RDF-entails \( S \) (resp., \( \varphi \)) if every RIF-RDF-model of \( C \) satisfies \( S \) (resp., \( \varphi \)).
A combination \( C \) is RIF-RDFS-entails \( S \) (resp., \( \varphi \)) if every RIF-RDFS-model of \( C \) satisfies \( S \) (resp., \( \varphi \)).

Note that simple entailment in combination with an empty rule set is not the same as simple entailment in RDF, since certain entailments involving datatypes are enforced by the RDF semantics in combinations, cf. the example involving strings and plain literals above.

4 OWL Compatibility

This section specifies how a RDF document interacts with a set of OWL ontologies in an RIF-OWL combination. The semantics of a combination is defined for OWL 2 [OWL2-Syntax]. Since OWL 2 is an extension of OWL 1 [OWL-Reference], the specification in this section applies also to combinations of RIF documents with OWL 1 ontologies.

OWL 2 specifies two different variants of the language: OWL 2 DL [OWL2-OWL1] and OWL 2 Full [OWL2-OWL-Based-Semantics], where the latter are RDF graphs that use OWL Vocabulary: the RDF representation of an OWL 2 DL ontology is also an OWL 2 Full ontology. OWL 1 [OWL1-Reference] and OWL 1 DL [OWL1-Reference], which are sublanguages of OWL 1, can be seen as syntactical subsets of OWL 2 DL. OWL 2 ontologies may be interpreted under one of two semantics: the Direct Semantics [OWL2-Semantics], which is only defined for OWL 2 DL, and is based on standard Description Logic semantics, and the RDF-Based Semantics [OWL2-RDF-Based-Semantics], which is defined for arbitrary OWL 2 DL. The syntax of OWL 2 DL is defined in terms of a Structural Specification, and there is a mapping to an RDF representation for interchange. The RDF representation of OWL 2 DL [OWL2-RDF-Mapping] does not extend the RDF syntax, but rather restricts it: every OWL 2 DL ontology in RDF form is an RDF graph, but not every RDF graph is an OWL 2 DL ontology. OWL 2 Full RDF have the same syntax: every RDF graph is an OWL 2 Full ontology and vice versa. This syntactical difference is reflected in the definition of RIF-OWL compatibility: combinations of RIF with OWL 2 DL are based on the OWL 2 Structural Specification, whereas combinations with OWL 2 Full are based on the RDF syntax.

Since the OWL 2 Full syntax is the same as the RDF syntax and the OWL 2 RDF-Based Semantics is an extension of the RDF Semantics, the definition of RIF-OWL 2 Full compatibility is an extension of RIF-RDF compatibility. However, defining RIF-OWL DL compatibility in the same way would entail losing certain properties of the Direct Semantics. One of the main reasons for this is the difference in the way classes and properties are interpreted in the RDF-Based and Direct Semantics. In the RDF-Based Semantics, classes and properties are interpreted as objects in the domain of interpretation, which are then associated with subsets of, respectively binary relations over the domain of interpretation, using the rdf:type property and the extension function \( \text{EXT} \), as in RDF. In the Direct Semantics, classes and properties are directly interpreted as subsets of, respectively binary relations over the domain. This is a key property of the first-order logic nature of Description Logic semantics and enables the use of Description Logic reasoning techniques for processing OWL 2 DL descriptions. Defining RIF-OWL DL compatibility as an extension of RIF-RDF compatibility would define a correspondence between OWL 2 DL statements and RDF frame formulas. Since RDF frame formulas are interpreted using an extension function, as in RDF, defining the correspondence between them and OWL 2 DL statements would change the semantics of between statements, even if the RDF document were empty.

A RIF-OWL combination that is faithful to the first-order nature of the OWL 2 Direct Semantics requires interpreting classes and properties as sets and binary relations, respectively, suggesting that a correspondence could be defined with unary and binary predicates. It is, however, also desirable that there be uniform syntax for the RIF component of both RIF-OWL 2 DL and RIF-RDF/OWL 2 Full combinations, because one may not know at the time of constructing the rules which type of inference will be used. Consider, for example, an RDF graph \( S \) consisting of the following statements:

- \( \text{ix: rdf:type owl:ontology} \).
- \( \text{a rdf:type C} \).

and a RIF document with the rule...
For all x (\(x[rdf:type \rightarrow D]\) :: \(x[rdf:type \rightarrow C]\))

The combination of the two, according to the specification of RDF Compatibility, allows deriving the triple

\[ a[rdf:type D]. \]

Now, the RDF graph S is also an OWL 2 DL ontology. Therefore, one would expect the triple to be implied according to the semantics of RIF-OWL DL combinations as well.

To ensure that the RIF-OWL DL combination is faithful to the OWL 2 Direct Semantics and to enable using the same, or similar, RIF rules in combinations with both OWL 2 DL and RDF/OWL 2 Full, the interpretation of frame formulas \(s[p \rightarrow o]\) in RIF-OWL DL combinations is slightly different from their interpretation in RIF and syntactical restrictions are imposed on the use of variables and function terms in frame formulas.

The remainder of this section formally defines combinations of RIF rules with OWL 2 DL and OWL 2 Full ontologies and the semantics of such combinations. A combination consists of a RIF document and a set of OWL ontologies. The semantics of combinations is defined in terms of combined models, which are pairs of RIF semantic multi-structures and OWL 2 Direct, respectively OWL 2 RDF-Based interpretations. The interaction between the structures and interpretations is defined through a number of concepts. Entailment is defined as model inclusion, as usual.

4.1 Syntax of RIF-OWL Combinations

Since RDF graphs and OWL 2 Full ontologies cannot be distinguished, the syntax of RIF-OWL 2 Full combinations is the same as the syntax of RIF-RDF combinations.

The syntax of OWL ontologies in RIF-OWL DL combinations is given by the Structural Specification of OWL 2 and the restrictions on OWL 2 DL ontologies [OWL2-Syntax]. Certain restrictions are imposed on the syntax of the RIF rules in combinations with OWL 2 DL. Specifically, the only terms allowed in class and property positions in class membership frame formulas are constant symbols. A DL-frame formula is a frame formula \(b_i \rightarrow c_i\), such that \(b_i = rdfs:Literal\) and for every \(b_i\), with \(1 \leq i \leq n\), it holds that \(b_i\) is a constant symbol and if \(b_i = rdfs:Literal\), then \(c_i\) is a constant symbol. A DL-class membership formula is a class membership formula \(\text{rdf:type} b\#c\) such that \(b\) is a constant symbol.

We do not allow subclass formulas in rule conditions in OWL 2 DL combinations, since at the time of writing there are no known effective and efficient ways of dealing with such subclass formulas in conditions in reasoners.

**Definition.** A rule condition \(\varphi\) is a DL-condition if every frame formula in \(\varphi\) is a DL-frame formula, and \(\varphi\) does not contain subclass formulas.

A RIF-BLD document formula \(R\) is a RIF-BLD DL-document formula if every frame formula in \(R\) is a DL-frame formula, every class membership formula in \(R\) is a DL-class membership formula, every subclass formula in \(R\) is a DL-subclass formula, and \(R\) does not contain any rules with subclass formulas.

**Definition.** A RIF-OWL DL-combination \(\vee D, O\) is DL-safe if every rule in \(R\) is DL-safe, given \(O\), a RIF-OWL DL-document formula and \(O\) is a set of OWL 2 DL ontologies of a Vocabulary \(V\) over an OWL 2 datatype map \(D\).

When clear from the context, RIF-OWL DL-combinations are referred to simply as combinations.

4.1.1 Safeness Restrictions

In the literature, several restrictions on the use of variables in combinations of rules and Description Logics have been identified [Motik05, Rosati06] for the purpose of decidable reasoning. This section specifies such safeness restrictions for RIF-OWL DL combinations.

Given a set of OWL 2 DL ontologies \(O\), a variable \(x\) in a RIF rule \(Q\), \(H \rightarrow B\) is DL-safe if it occurs in an atomic formula in \(B\) that is not of the form \(s[p \rightarrow o]\) or \(s[rdf:type \rightarrow A]\), where \(s, p, o,\) and \(A\) are terms (possibly including \(x\)) and \(p\) or \(o\) occurs in one of the ontologies in \(O\). A disjunction-free RIF rule \(Q(H \rightarrow B)\) is DL-safe, given \(O\), if every variable that occurs in \(H\) or \(B\) is DL-safe. A disjunction-free RIF rule \(Q(H \rightarrow B)\) is weakly DL-safe, given \(O\), if every variable that occurs in \(H\) is DL-safe.

**Definition.** A RIF-OWL DL-combination \(\vee D, O\) is DL-safe if every rule in \(R\) is DL-safe, given \(O\), a RIF-OWL DL-combination \(\vee D, O\) is weakly DL-safe if every rule in \(R\) is weakly DL-safe, given \(O\).

4.1.2 Datatypes in OWL 2

Compared with RDF and the RIF, OWL 2 uses a slightly extended notion of datatype.

In the remainder of this section, a datatype \(D\), contains, in addition to the lexical space, value space, and lexical-to-value mapping, a facet space, which is a set of pairs \((\text{facet}, \text{value})\), where \(\text{facet}\) is an IRI and \(\text{value}\) is a data value, and a facet-to-value mapping, which is a mapping from facets to subsets of the value space of \(d\).

An OWL 2 datatype map \(D\) is a datatype map that maps the IRIs of the datatypes specified in Section 4 of [OWL2-Syntax] to the corresponding datatypes such that the domain of \(D\) does not include rdfs:Literal.

We note here that the definitions of datatypes and datatypes map in the OWL 2 Direct Semantics specification [OWL2-Semantics] are somewhat different. There, a datatype is some entity with some associated IRIs, and the datatype map assigns lexical value, and facet spaces, as well as lexical-to-value and facet-to-value mappings. The definitions of datatype and datatype map we use are isomorphic, and, indeed, the same as in the OWL 2 RDF-Based Semantics specification [OWL2-RDF-Based-Semantics]. The latter does not preclude the use of rdfs:Literal in datatype maps. Note that we do not restrict the use of rdfs:Literal in OWL 2 ontologies or RDF graphs.

4.2 Semantics of RIF-OWL Combinations

The semantics of RIF-OWL 2 Full combinations is a straightforward extension of the Semantics of RIF-RDF Combinations.

The semantics of RIF-OWL 2 DL combinations cannot straightforwardly extend the semantics of RIF-RDF combinations, because the OWL 2 Direct Semantics does not extend the RDF Semantics. In order to keep the syntax of the rules uniform between RIF-OWL 2 Full and RIF-OWL DL combinations, the semantics of RIF frame formulas is slightly altered in RIF-OWL DL combinations.

4.2.1 OWL RDF-Based Semantics

Given an OWL 2 datatype map \(D\) and a Vocabulary \(V\) that includes the domain of \(D\) and the OWL 2 RDF-Based Vocabulary Vocabulary, a RIF interpretation \(I\) is an OWL 2 RDF-Based Interpretation of \(V\) with respect to \(D\) if it satisfies the semantic conditions in Section 5 of [OWL2-RDF-Based-Semantics].

The semantics of RIF-OWL 2 Full combinations is a straightforward extension of the semantics of RIF-RDF combinations. It is based on the same notion of common interpretations, but defines additional notions of satisfiability and entailment.

**Definition.** Let \(I, J\) be a common-RIF-RDF-interpretation that is a model of a RIF-RDF combination \(C = \vee R, S\) and let \(D\) be an OWL 2 datatype map conforming with the set of datatypes in \(I\), \((J, I)\) is an RIF-OWL RDF-Based-model of \(C\) if \(I\) is an OWL 2 RDF-Based Interpretation with respect to \(D\); in this case \(C\) is RIF-OWL RDF-Based-satisfiable with respect to \(D\) if every RIF-OWL RDF-Based-model of \(C\) satisfies \(q\).

Let \(C\) be a RIF-RDF combination, let \(S\) be a generalized RDF graph, let \(q\) be a condition formula, and let \(D\) be an OWL 2 datatype map conforming with the set of considered datatypes. C RIF-OWL RDF-Based-entails \(q\) with respect to \(D\) if every RIF-OWL RDF-Based-model of \(C\) satisfies \(q\). Likewise, C RIF-OWL RDF-Based-entails \(q\) with respect to \(D\) if every RIF-OWL RDF-Based-model of \(C\) satisfies \(q\).
4.2.2.2 Semantics of RIF-OWL DL Combinations

As defined in [OWL2-Semantics], an interpretation for a Vocabulary V over a datatype map D is a tuple \( I = ( \text{IR}, \text{LV}, \text{OP}, \text{DP}, \text{I}, \text{D}, \text{DT}, \text{FA}) \), where

- \( \text{IR} \) is a non-empty set, called the object domain,
- \( \text{LV} \) is a non-empty set, called the data domain, which includes all value spaces of the datatypes in the range of \( D \),
- \( \text{OP} \) is a mapping from object properties to subsets of \( \text{IR} \times \text{IR} \),
- \( \text{DP} \) is a mapping from object properties to subsets of \( \text{IR} \times \text{IR} \),
- \( \text{I} \) is a mapping from individuals into \( \text{IR} \),
- \( \text{D} \) is a mapping from datatypes to subsets of \( \text{LV} \),
- \( \text{DT} \) is a mapping from datatypes to subsets of \( \text{LV} \),
- \( \text{FA} \) is a mapping from \( \text{IR} \) to subsets of value spaces in \( D \).

The OWL semantics imposes a number of further restrictions on the mapping functions to ensure the interpretation of datatypes, literals, and facets conforms with the given datatype map D and to define the semantics of built-in classes and properties (e.g., owl:Thing). The mappings DT, FA, and D are essentially given by the datatype map.

Definition. Given a Vocabulary \( V \) over an OWL 2 datatype map \( D \), a common-RIF-OWL Direct-interpretation \( I \) for \( V \) over \( D \) is a pair \((I,\bar{I})\), where \( I \) is a di-semantic multi-structure of the form \((J,I)^3,...,I_i,...)\), and \( \bar{I} \) is an interpretation for \( V \) over \( D \), such that the following conditions hold.

1. \( D \) is conforming with the datatypes in \( I \).
2. (IR union LV) is Dind.
3. \( \bar{C}(c) \) is the set of all objects \( k \) such that \( h_{\text{ datatype}}(h_{\text{subclass}}(k)(\text{rdf:type}))(\text{IC}(c)(k)) \) is true, for every \( c \) identifying a class in \( V \).
4. \( \text{DT}(c) \) is the set of all objects \( k \) such that \( h_{\text{ datatype}}(h_{\text{subclass}}(k)(\text{rdf:type}))(\text{IC}(c)(k)) \) is true, for every \( c \) identifying a datatype in \( V \).
5. \( \text{OP}(p) \) is the set of all pairs \( (k,\bar{l}) \) such that \( h_{\text{op}}(h_{\text{prop}}(\text{IC}(p)(\text{IC}(c)(k))))(\text{IC}(l)) \) is true, for every \( p \) identifying an object property in \( V \).
6. \( \text{DP}(p) \) is the set of all pairs \( (k,\bar{l}) \) such that \( h_{\text{dp}}(h_{\text{prop}}(\text{IC}(p)(\text{IC}(c)(k))))(\text{IC}(l)) \) is true, for every \( p \) identifying a datatype property in \( V \).
7. \( l(\bar{k}) \) is \( k(\bar{x}) \) for every \( \bar{k} \) identifying an individual in \( V \).
8. \( \text{OP}(p) \) is the set of all objects \( k \) such that \( h_{\text{op}}(h_{\text{prop}}(\text{IC}(p)(\text{IC}(c)(k))))(\text{IC}(l)) \) is true, for every \( p \) identifying a class in \( V \).
9. \( \text{DP}(p) \) is the set of all objects \( k \) such that \( h_{\text{dp}}(h_{\text{prop}}(\text{IC}(p)(\text{IC}(c)(k))))(\text{IC}(l)) \) is true, for every \( p \) identifying a datatype in \( V \).
10. \( \text{DT}(c) \) is a subset of \( \text{DT}(\text{C}(d)) \) for every \( \text{C} \) identifying a datatype in \( V \).

Condition 2 ensures that the relevant parts of the domains of interpretation are the same. Conditions 3 and 4 ensure that the interpretation (extension) of an OWL class or datatype identified by an IRI corresponds to the interpretation of frames of the form \( \text{IR} = \text{IR} \times \text{IR} \times \text{IR} \times \text{IR} \), Conditions 5 and 6 ensure that the interpretation (extension) of an OWL object or data property identified by an IRI corresponds to the interpretation of frames of the form \( \text{IR} \times \text{IR} \times \text{IR} \times \text{IR} \times \text{IR} \times \text{IR} \). Condition 7 ensures that individual identifiers in the OWL ontologies and the RIF documents are interpreted in the same way. Conditions 8 and 9 ensure that typing in OWL and typing in RIF correspond, i.e.,

\[ \text{ClassAssertion} (a) \text{ is true iff a } \# \text{ is true.} \]

Finally, 10 ensures that whenever a RIF subclass statement holds, the corresponding OWL subclass statement holds as well, i.e., \( \text{SubClassOf}(a, b) \text{ is true iff a } \# b \text{ is true.} \]

Using the definition of common-RIF-OWL Direct-interpretation, satisfaction, models, and entailment are defined in the usual way:

Definition. A common-RIF-OWL Direct-interpretation \((I,\bar{I})\) for a Vocabulary \( V \) over an OWL 2 datatype map \( D \) is an RIF-OWL Direct-model of a RIF-OWL DL-combination if \( \bar{I} \) is a model of \( R \) and \((I,\bar{I})\) is a model of every ontology \( O \) in \( D \); in this case \( C \) is RIF-OWL Direct-satisfiable for \( V \) over \( D \).

Let \( C \) be a RIF-OWL DL-combination, let \( D \) be an OWL 2 DL ontology, let \( \text{D-condition} \) be a DL-condition formula, and let \( D \) be an OWL 2 datatype map conforming with the set of considered datatypes, and let \( V \) be a Vocabulary over \( D \) for every ontology in \( C \) and for \( D \). C RIF-OWL Direct-entails \( D \) with respect to \( D \) if every common-RIF-OWL Direct-
interpretation for $V$ over $D$ that is an RIF-OWL Direct-model of $C$ is an RIF-OWL Direct-model of $O$. Likewise, $C$ RIF-OWL Direct-entails $\varphi$ with respect to $D$ if every common-RIF-OWL Direct-interpretation for $V$ over $D$ that is an RIF-OWL Direct-model of $C$ is an RIF-OWL Direct-model of $\varphi$. □

Example. In the OWL 2 Direct Semantics, the domains for interpreting individuals respectively, literals (data values), are disjoint. The disjointness entails that data values cannot be members of a class and individuals cannot be members of a datatype.

RIF does not make such distinctions; variable quantification ranges over the entire domain. So, the same variable may be assigned to an abstract individual or a concrete data value. Additionally, RIF constants (e.g., IRIs) denoting individuals can be written in place of a data value, such as the value of a data-valued property or in datatyped membership statements; similarly for constants denoting data values. Such statements cannot be satisfied in any common-RIF-OWL Direct-interpretation. The following example illustrates several such statements.

Consider the datatype $xs:string$ and a RIF-OWL DL combination consisting of the set containing only an OWL 2 DL ontology that contains

\[
\text{ex:myiri rdf:type ex:A}.
\]

and a RIF document containing the following fact

\[
\text{ex:myiri[rdf:type -> xs:string]}
\]

This combination is not RIF-OWL Direct-satisfiable, because $\text{ex:myiri}$ is an individual identifier and $S$ maps individual identifiers to elements in $O$, which is disjoint from the elements in the datatype $xs:string$.

Consider a RIF-OWL DL combination consisting of the set containing only the OWL 2 DL ontology

\[
\text{ex:hasChild rdf:type owl:ObjectProperty}.
\]

and a RIF document containing the following fact

\[
\text{ex:myiri[ex:hasChild -> "John"]}
\]

This combination is not RIF-OWL Direct-satisfiable, because $\text{ex:hasChild}$ is an object property, and values of object properties may not be concrete data values.

Consider a RIF-OWL DL combination consisting of the OWL DL ontology

\[
\text{SubClassof}(\text{ex:A ex:B})
\]

and a RIF document containing the following rule

\[
\text{Forall}\ x\ (\text{?x[rdf:type -> ex:A]})
\]

This combination is not RIF-OWL Direct-satisfiable, because the rule requires every element, including every concrete data value, to be a member of the class $\text{ex:A}$. However, since every OWL interpretation requires every member of $\text{ex:A}$ to be an element of the object domain, concrete data values cannot be members of the object domain. □

5 Importing RDF and OWL in RIF

In the preceding sections, RIF-RDF Combinations and RIF-OWL combinations were defined in an abstract way, as pairs consisting of a RIF document and a set of RDF graphs/OWL ontologies. In addition, different semantics were specified based on the various RDF and OWL entailment regimes. RIF provides a mechanism for explicitly referring to (importing) RDF graphs from documents and specifying the intended profile (entailment regime) through the use of Import statements.

This section specifies how RIF documents with such import statements must be interpreted.

A RIF document contains a number of Import statements. Unary Import statements are used for importing RIF documents, and the interpretation of these statements is defined in Section 3.5 of RIF-BLD. This section defines the interpretation of binary Import statements:

\[
\text{Import(t1} <p1> ... Import(tn} <pn>)
\]

Here, $t_i$ is an absolute IRI referring to an RDF graph to be imported and $p_i$ is an absolute IRI denoting the profile to be used for the import.

The profile determines which notions of model, satisfiability and entailment must be used. For example, if a RIF document $R$ imports an RDF graph $S$ with the profile RDF, the notions of RIF-RDFS-model, RIF-RDFS-satisfiability, and RIF-RDFS-entailment must be used for the combination $\langle R, \{S\} \rangle$.

Profiles are ordered as specified in Section 5.1.1. If several graphs are imported in a document, and these imports specify different profiles, the highest of these profiles is used. For example, if a RIF document $R$ imports an RDF graph $S_1$ with the profile RDF and an RDF graph $S_2$ with the profile OWL RDF-Based, the notions of RIF-OWL RDF-Based-model, RIF-OWL RDF-Based-satisfiability, and RIF-OWL RDF-Based-entailment must be used with the combination $\langle R, \{S_1, S_2\} \rangle$.

Finally, if a RIF document $R$ imports an RDF graph $S$ with the profile OWL Direct, $R$ must be a RIF-OWL DL-document formula. $S$ must be the RDF representation of an OWL 2 DL ontology $O$, and the notions of RIF-OWL Direct-model, RIF-OWL Direct-satisfiability, and RIF-OWL Direct-entailment must be used with the combination $\langle R, \{O\} \rangle$.

5.1 Profiles of Imports

RIF defines specific profiles for the different notions of model, satisfiability and entailment of combinations, as well as one generic profile. The use of a specific profile specifies how a combination should be interpreted. If a specific profile cannot be handled by a receiver, the combination should be rejected. The use of a generic profile implies that a receiver may interpret the combination to the best of its ability.

The use of profiles is not restricted to the profiles specified in this document. Any specific profile that is used with RIF must specify an IRI that identifies it, as well as associated notions of model, satisfiability, and entailment for combinations.

5.1.1 Specific Profiles

The following table lists the specific profiles defined by RIF, the IRIs of these profiles, and the notions of model, satisfiability, and entailment that must be used with the profile.

<table>
<thead>
<tr>
<th>Profile</th>
<th>IRI of the Profile</th>
<th>Specifics in RIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td><a href="http://www.w3.org/ns/entailment/Simple">http://www.w3.org/ns/entailment/Simple</a></td>
<td>RIF-Simple-model</td>
</tr>
<tr>
<td>RDF</td>
<td><a href="http://www.w3.org/ns/entailment/RDF">http://www.w3.org/ns/entailment/RDF</a></td>
<td>RIF-RDFS-model</td>
</tr>
<tr>
<td>RDFS</td>
<td><a href="http://www.w3.org/ns/entailment/RDFS">http://www.w3.org/ns/entailment/RDFS</a></td>
<td>RIF-RDFS-model</td>
</tr>
<tr>
<td>D</td>
<td><a href="http://www.w3.org/ns/entailment/D">http://www.w3.org/ns/entailment/D</a></td>
<td>RIF-D-model</td>
</tr>
<tr>
<td>OWL Direct</td>
<td><a href="http://www.w3.org/ns/entailment/OWL-Direct">http://www.w3.org/ns/entailment/OWL-Direct</a></td>
<td>RIF-OWL Direct-model</td>
</tr>
</tbody>
</table>
5.1.2 Generic Profile

RIF specifies one generic profile. The use of the generic profile does not imply the use of a specific notion of model, satisfiability, and entailment.

6 Conformance Clauses

We define notions of conformance for RIF-RDF and RIF-OWL combinations. We define these notions both for the RIF Core [RIF-Core] and RIF BLD [RIF-BLD] dialects.

Conformance is described in terms of semantics-preserving transformations between the native syntax of a compliant processor and the XML syntax of RIF Core and BLD.

We say that an RDF graph $G$ is a standard RDF graph if for every triple $s p o$ in $G$, $s$ is an IRI or blank node, $p$ is an IRI, and $o$ is an IRI, literal, or blank node. A combination $<R, S>$ is standard if every graph in $S$ is standard.

Each RIF processor has a set of supported datatypes and symbol spaces that include the symbol spaces listed in [RIF-TTD], and of supported external terms that include the built-ins listed in [RIF-TTD]. The datatype map of a RIF processor is the smallest datatype map conforming with the set of datatypes in $T$.

Now, let $P$ (Simple, RDF, RDFS, D, OWL Direct, OWL RDF-Based) be a specific RDF profile. A RIF-RDF combination $C = < R, S >$ is a BLDT$_P$ combination if $R$ is a BLDT$_P$ formula and $C$ is a Core$_E$P combination if $R$ is a Core$_E$ formula.

A RIF-OWL DL-combination $C = < R, O >$ is a BLDT$_O$:OWL Direct combination if $R$ is a BLDT$_O$ formula and $C$ is a Core$_E$:OWL Direct combination if $R$ is a Core$_E$ formula.

A RIF processor is a compliant BLDT$_P$ consumer, for $P$ (Simple, RDF, RDFS, D, OWL Direct, OWL RDF-Based), if it implements a semantics-preserving mapping, $\mu$, from the set of standard BLDT$_P$ combinations, standard RDF graphs, OWL 2 ontologies, and BLDT$_E$ formulas to the language $L$ of the processor ($\mu$ does not need to be an "onto" mapping) and, in case $P$ (OWL Direct, OWL RDF-Based), its datatype map is an OWL 2 datatype map.

We say that a RIF document $D$ is list-safe if it contains no occurrences of rdf:first, rdf:rest, or rdf:nil in rule consequents. An RDF graph $G$ is list-safe if it contains no occurrences of rdf:first or rdf:rest outside of the property positions, it contains no occurrences of rdf:nil outside of triples of the form $\ldots$ rdf:rest o2 o1 $\ldots$, and there are no two triples $s$ rdf:rest o1 o2 and $s$ rdf:first o2 o1 $\ldots$. A combination $< R, S >$ is list-safe if $R$ is list-safe and the merged of the graphs in $S$ is list-safe.

A RIF processor is a compliant Core$_E$P consumer, for $P$ (Simple, RDF, RDFS, D, OWL Direct, OWL RDF-Based), if it implements a semantics-preserving mapping, $\mu$, from the set of standard list-safe Core$_E$:P combinations, standard RDF graphs, OWL 2 ontologies, and Core$_E$:E formulas to the language $L$ of the processor ($\mu$ does not need to be an "onto" mapping) and, in case $P$ (OWL Direct, OWL RDF-Based), its datatype map is an OWL 2 datatype map.

Formally, this means that for any pair $(q, q')$, where $q$ is a BLDT$_E$P combination and $q'$ is an RDF graph, OWL 2 ontology, or BLDT$_E$ formula such that $q \models q'$ is defined, $q \models q'$ iff $\mu(q) \models \mu(q')$. Here $\models$ denotes P-entailment and $\models$ denotes the logical entailment in the language $L$ of the RIF processor.

A RIF processor is a compliant BLDT$_E$P producer if it implements a semantics-preserving mapping, $\nu$, from the language $L$ of the processor to the set of all BLDT$_E$ formulas, RDF graphs, OWL 2 ontologies, and BLDT$_E$:P combinations ($\nu$ does not need to be an "onto" mapping).

Formally, this means that for any pair $(q, q')$ of formulas in $L$ such that $q \models q'$ is defined, $q \models q'$ iff $\nu(q') \models \nu(q')$. Here $\models$ denotes P-entailment and $\models$ denotes the logical entailment in the language $L$ of the RIF processor.

7 Acknowledgements

This document is the product of the Rules Interchange Format (RIF) Working Group (see below), the members of which deserve recognition for their time and commitment to RIF. The editors extend special thanks to: Mike Dean, Michael Kifer, Stella Mitchell, AXel Polleres, and Dave Reynolds, for their thorough reviews and insightful discussions; the working group chairs, Chris Welty and Christian de Sainte-Marie, for their invaluable technical help and inspirational leadership; and W3C staff contact Sandra Hawke, a constant source of ideas, help, and feedback.
The following members of the joint RIF-OWL task force have contributed to the OWL Compatibility section in this document: Mike Dean, Peter F. Patel-Schneider, and Ulrike Sattler.

The regular attendees at meetings of the Rule Interchange Format (RIF) Working Group at the time of the publication were: Adrian Paschke (Freie Universität 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).

8 References

8.1 Normative References

[RIF-OWL-BASED-SEMASTICS]

[RIF-OWL-SEMASTICS]

[RIF-OWL-SYNTAX]

[RDF-DESCRIPTION]

[RDF-SEMATICS]

[RIF-CORE]

[RIF-BLD]

[RIF-DTB]

8.2 Informational References

[CURIE]

[DLP]

[Motik05]

[OWL-REFERENCE]

[OWL-SEMATICS]

[OWL2-PROFILES]

[OWL2-RDF-MAPPING]

[RDF-SHEMA]

[RDF-SYNTAX]

[RFC-3066]

[RIF-PRD]

[RIF-UCRA]

[Rosati06]

[SWRL]

[FLWRL]

[XML SCHEMA DATATYPES]

9 Appendix: Embeddings (Informative)

RIF-RDF combinations can be embedded into RIF documents in a fairly straightforward way, thereby demonstrating how a RIF-compliant translator without native support for RDF can process RIF-RDF combinations.

RIF-OWL combinations cannot be embedded in RIF, in the general case. However, there is a subset of OWL 2 DL, namely the OWL 2 RL profile (OWL2-Profiles), for which RIF-OWL combinations that can be embedded.

Simple, RDF, RDFS and OWL 2 RL entailment for RIF-RDF combinations are embedded in RIF BLD.
Note that Simple, RDF and RDFS entailments are superficially embeddable within RIF Core. However, condition 7 of the semantics of RIF-RDF combinations cannot be axiomatized in RIF Core due to restrictions on the use isa (#) in rule heads. OWL 2 RL is not embeddable in RIF Core due to the need for equality reasoning.

The embeddings are defined using an embedding function tr that maps symbols, triples, and RDF graphs/OWL ontologies to RIF symbols, statements, and documents, respectively.

To embed consistency checking in RDF(S) and OWL, we use a special 0-ary predicate symbol rif:error, which is assumed not to be used in the RIF documents in the combination.

Besides the namespace prefixes defined in the Overview, the following namespace prefix is used in this appendix: pred refers to the RDF namespace for built-in predicates http://www.w3.org/1999/02/22-rdf/owl

To facilitate the definition of the embeddings we define the notion of a merge of RIF formulas.

Definition. Let \( R = (R_1, \ldots, R_i) \) be a set of document, group, and rule formulas, such that there are no prefix or base directives, or relative IRIs in \( R \) and \( \text{directives}_{\text{tr}} \) are all the import directives occurring in document formulas in \( R \). The merge of \( R \), denoted \( \text{merge}(R) \), is defined as document\( \langle \text{directives}_1 \ldots | \text{documents}_i \rangle \), where \( \text{documents}_i \) is obtained from \( R_i \) in the following way:

- if \( R_i \) is a document formula of the form document\( \langle \text{directives}_i \ldots | \text{directives}_i \rangle \), then \( \text{R}_i \rightarrow \text{R}_i \) and
- if \( R_i \) is a non-document formula (i.e., fact, rule, or group), then \( \text{R}_i \rightarrow \text{R}_i \).

Note that the requirement that no prefix or based directives, or relative IRIs are included in any of the formulas to be merged is not a limitation, since compact IRIs can be rewritten to absolutes IRIs, as can relative IRIs, by exploiting prefix and base directives, and the location of the document.

9.1 Embedding RIF-RDF Combinations

RIF-RDF combinations are embedded by combining the RIF rules with embeddings of the RDF graphs and an axiomatization of Simple, RDF, and RDFS entailment.

The embedding is not defined for combinations that include infinite RDF graphs and for combinations that include RDF graphs with RDF URI references that are not absolute IRIs (see the End note on RDF URI references) or plain literals without language tags that are not in the lexical space of the xs:string datatype [XML-Schema]. Also, the embedding is not defined for RDF lists.

We define a list-free combination as a combination that does not contain any mention of the symbols rdf:first, rdf:rest, or rdf:nil.

In the remainder of this section we first define the embedding of symbols, triples, and graphs, after which we define the axiomatization of Simple, RDF, and RDFS entailment of combinations and, finally, demonstrate faithfulness of the embeddings.

9.1.1 Embedding Symbols

Given a combination \( C = \langle R, S \rangle \), the function \( \text{tr} \) maps RDF symbols of a Vocabulary \( V \) and a set of blank nodes \( B \) to RIF symbols, as defined in the following table. It is assumed that the Vocabulary \( V \) includes all the IRIs and literals used in the RDF documents and condition formulas under consideration.

In the table, the mapping \( \text{tr} \) is an injective function that maps typed literals to new constants in the rif:local symbol space, where a new constant is a constant that is not used in the document or its vicinity (i.e., imported or entailed formula, or entailing combination). It "generates" a new constant from a typed literal.

<table>
<thead>
<tr>
<th>RDF Symbol</th>
<th>RIF Symbol</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRI i in ( V )</td>
<td>Constant with symbol space rif:iri</td>
<td>( \text{tr}(i) = \langle i \rangle )</td>
</tr>
<tr>
<td>Blank node :x :* in ( B )</td>
<td>Variable with symbol space ( x )</td>
<td>( \text{tr}(. :x ) = ?x )</td>
</tr>
<tr>
<td>Plain literal without a language tag xxx in ( V )</td>
<td>Constant with the symbol space xs:string</td>
<td>( \text{tr}(\text{xxx}) = \text{&quot;xxx&quot;} )</td>
</tr>
<tr>
<td>Plain literal with a language tag &quot;xxx&quot;@lang in ( V )</td>
<td>Constant with the symbol space rdf:PlainLiteral</td>
<td>( \text{tr}(\text{&quot;xxx&quot;@lang}) = \text{&quot;xxx@lang&quot;#rdfs:PlainLiteral} )</td>
</tr>
<tr>
<td>Well-typed literal &quot;s&quot;^^&quot;u&quot; in ( V )</td>
<td>Constant with the symbol space u</td>
<td>( \text{tr}(\text{&quot;s&quot;^^&quot;u&quot;}) = \text{&quot;s&quot;^^&quot;u&quot;} )</td>
</tr>
<tr>
<td>Non-well-typed literal &quot;s&quot;^^&quot;u&quot; in ( V )</td>
<td>Local constant ( s \cdot u ) that is not used in ( C ) and is obtained from ( &quot;s&quot;^^&quot;u&quot; )</td>
<td>( \text{tr}(\text{&quot;s&quot;^^&quot;u&quot;}) = \text{&quot;s&quot;^^&quot;u&quot;} )</td>
</tr>
</tbody>
</table>

9.1.2 Embedding Triples and Graphs

This section extends the mapping function \( \text{tr} \) to triples and defines two embedding functions for RDF graphs. In one embedding (trG), graphs are embedded as RIF documents and variables (originating from blank nodes) are skolemized, i.e., replaced with new constant symbols. In the other embedding (trS), graphs are embedded as condition formulas and variables (originating from blank nodes) are existentially quantified. The following sections show how these embeddings can be used for reasoning with combinations.

For skolemization we assume a function \( \text{sk} \) that takes as argument a formula \( \Phi \) and returns a formula \( \Phi' \) that is obtained from \( \Phi \) by replacing every variable symbol \( ?x \) with \( \langle \text{new-iri} \rangle \), where \( \text{new-iri} \) is a new globally unique IRI (i.e., it does not occur in the graph or its vicinity (i.e., entailing combination or entailed graph/formula).

<table>
<thead>
<tr>
<th>RDF Construct</th>
<th>RIF Construct</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triple s p o</td>
<td>Frame formula ( \text{tr(s)}[\text{tr(p)} \rightarrow \text{tr(o)}] )</td>
<td>( \text{tr}(s \ p \ o) = \text{tr(s)}[\text{tr(p)} \rightarrow \text{tr(o)}] )</td>
</tr>
<tr>
<td>Graph S</td>
<td>Group formula ( \text{tr(G)}(S) = \text{sk} \langle \text{document then} \text{document} \ldots \rangle ), where ( t_1, \ldots, t_n ) are the triples in ( S )</td>
<td>( \text{tr}(S) = \text{sk} \langle \text{document then} \text{document} \ldots \rangle ), where ( t_1, \ldots, t_n ) are the triples in ( S )</td>
</tr>
<tr>
<td>Graph S</td>
<td>Condition formula ( \text{tr} \langle S \rangle )</td>
<td>( \text{tr}(S) = \text{Exists } \text{tr(\text{x}) then } \text{tr(\text{y})} \text{ and } \text{tr(\text{t})} \text{ and } \text{tr(\text{t})}, \text{where } \text{x}, \ldots, \text{x}_k \text{ are the blank nodes occurring in } S \text{ and } t_1, \ldots, t_n \text{ are the triples in } S )</td>
</tr>
</tbody>
</table>

9.1.3 Embedding Simple Entailment

The semantics of the RDF Vocabulary does not need to be axiomatized for Simple entailment. Nonetheless, the connection between RIF class membership and subclass statements and the RDF type and subclass statements needs axiomatization. We define:

\[
\begin{align*}
\text{RSimple} & = \text{Document} \langle \text{Group} \rangle \\
& = \forall ?x ?y \langle \text{rdfs:subClassOf} \rightarrow ?y \rangle : \langle ?x \# ?y \rangle \\
& \lor \forall ?x ?y \langle \text{rdf:type} \rightarrow ?y \rangle : \langle ?x \# ?y \rangle \\
& \lor \forall ?x ?y \langle \text{rdfs:subClassOf} \rightarrow ?y \rangle : \langle ?x \# ?y \rangle \\
& \lor \forall ?x ?y \langle \text{rdf:type} \rightarrow ?y \rangle : \langle ?x \# ?y \rangle 
\end{align*}
\]
Theorem: A list-free RIF-RDF combination $C = \langle R, \Sigma \rangle$ is RIF-Satisfiable, $(\Sigma, S_1, ..., S_n)$ RIF-RDF entails a generalized RDF graph $T$ if and only if merge($(R^{\text{RDF}}, R, \Sigma)$, $..., \Sigma, S_n))$ entails $\text{tr}(T)$; C RIF-RDF entails $\phi$ condition formula $\phi$ if and only if merge($(R^{\text{RDF}}, R)$, $..., \Sigma, S_n)))$ entails $\phi$.

Proof. Consider the following parts of the theorem.

We proceed by contradiction. Assume I satisfies $\phi$. Consider the common-RIF-RDF-interpretation $\langle I, \Sigma \rangle$ that satisfies $\phi$. (See Section 1.5 of [RDF-Semantics].) Let $S$ be a semantic multi-structure $S$ in the same way as above, and so it satisfies $\phi$.

Proof. We prove both directions through contraposition. We first consider condition formulas (the first part of the theorem), after which we consider graphs/formulas considered for entailment checking). The set of

To finitely embed RDF entailment, we need to consider a subset of the graphs/formulas considered for entailment checking. We axiomatize the semantics of the RDF Vocabulary using the following RIF rules.

We proceed by contradiction. Assume I satisfies $\phi$. Consider the common-RIF-RDF-interpretation $\langle I, \Sigma \rangle$ that satisfies $\phi$. By conditions 7, 8, and 4 in the definition of common-RIF-RDF-interpretation, the fact that I satisfies $\Sigma$ and by construction of $\phi$ it follows that $\phi$ is a model of $\Sigma$. So $\phi$ is a model of $\Sigma$. Since I is not a model of $\phi$ and $\phi$ does not contain any of the new IRIs, $\phi$ is not the model of $\phi$. Therefore, $\phi$ does not entail $\phi$.

Assume now that $\phi$ does not entail $\phi$. We have that $\phi$ is a model of $\phi$, by assumption. So, $\phi$ satisfies $\phi$. Again, by assumption, $\phi$ is not a model of $\phi$. Therefore, $\phi$ does not entail $\phi$.

9.1.4 Embedding RDF Entailment

We axiomatize the semantics of the RDF Vocabulary using the following RIF rules.

To finitely embed RDF entailment, we need to consider a subset of the RDF axiomatic triples.

We assume that none of unary predicate symbols $\text{ex:wellxml}$ and $\text{ex:illxml}$ and no datatypes beyond those found in [RIF-DTB] are used in the context of the given combination and $\text{pred:is-literal-\text{anyURI}} \ldots \text{pred:is-literal-XMLLiteral}$ are the positive guard predicates defined in [RIF-DTB].

Here, inconsistencies may occur if non-well-typed XML literals, axiomatized using the $\text{ex:illxml}$ predicate, are in the class extension of rdf:XMLLiteral. If this situation occurs, rdf:is-derivation, which signifies an inconsistency in the combination.

Theorem: An RIF-RDF-satisfiable list-free RIF-RDF combination $C = \langle R, \Sigma \rangle$ is RIF-Satisfiable, $(\Sigma, S_1, ..., S_n)$ RIF-RDF entails $T$ if and only if merge($(R^{\text{RDF}}, R, \Sigma)$, $..., \Sigma, S_n))$ entails $\text{tr}(T)$. C RIF-RDF entails $\phi$ condition formula $\phi$ if and only if merge($(R^{\text{RDF}}, R)$, $..., \Sigma, S_n)))$ entails $\phi$.

Proof. In the proof we abbreviate merge($(R^{\text{RDF}}, R, \Sigma)$, $..., \Sigma, S_n)))$ with $\phi$. Proof. We prove both directions through contraposition. We first consider condition formulas (the first part of the theorem), after which we consider graphs/formulas considered for entailment checking). The set of

To finitely embed RDF entailment, we need to consider a subset of the graphs/formulas considered for entailment checking. We axiomatize the semantics of the RDF Vocabulary using the following RIF rules.

We proceed by contradiction. Assume I satisfies $\phi$. Consider the common-RIF-RDF-interpretation $\langle I, \Sigma \rangle$ that satisfies $\phi$. By conditions 7, 8, and 4 in the definition of common-RIF-RDF-interpretation, the fact that I satisfies $\Sigma$ and by construction of $\phi$ it follows that $\phi$ is a model of $\Sigma$. So $\phi$ is a model of $\phi$. Since I is not a model of $\phi$ and $\phi$ does not contain any of the new IRIs, $\phi$ is not the model of $\phi$. Therefore, $\phi$ does not entail $\phi$.

Assume now that $\phi$ does not entail $\phi$. We have that $\phi$ is a model of $\phi$, by assumption. So, $\phi$ satisfies $\phi$. Again, by assumption, $\phi$ is not a model of $\phi$. Therefore, $\phi$ does not entail $\phi$.
We assume that, for every non-well-typed literal of the form \([s, rdf:XMLLiteral] in V_{\text{in}}, k([s, **] \text{if } rdf:XMLLiteral]) is not in the value space of any of the considered datatypes and \(tr([s, **] \text{if } rdf:XMLLiteral])[\text{rdf:type} \rightarrow \text{rdf:XMLLiteral})] is not satisfied in \(I\). Since \(C\) is RIF-RDF-satisfiable, one can verify that this does not compromise satisfaction of \(R\). Finally, we may assume, without loss of generality, that \(I\) does not satisfy \(\text{rif:error}\). See also the proof of the following theorem.

For any positive integer \(j\) such that \(\text{rdf}: j\) does not occur in the context of \(C\), \(I\) and \(I'\) are extended such that \(I([\text{rdf}: j])=I([\text{rdf}: m])\) (see the definition of finite-axiomatic triples above for the definition of \(m\)). Clearly, this does not compromise satisfaction of \(R\) or non-satisfaction of \(\phi\) and \(\text{rif:}y\).

To show that \(I\) is an RDF-interpretation, we need to show that \(I\) satisfies the RDF axiomatic triples and the RDF semantic conditions.

Satisfaction of the axiomatic triples follows immediately from the inclusion of \(tr([l])\) in \(R_{\text{RDF}}\) for every RDF finite-axiomatic triple \(l\), the fact that \(I\) is a model of \(R_{\text{RDF}}\), and construction of \(I\). Consider the three RDF semantic conditions:

1. Satisfaction of condition 1 follows from satisfaction of the first rule in \(R_{\text{RDF}}\) in \(I\) and construction of \(I\); specifically, the second bullet in the definition.
2. Consider a well-typed XML literal \("xxx\" \text{if } rdf:XMLLiteral}\. By the definition of satisfaction in RIF BLD, \(I(["xxx\" \text{if } rdf:XMLLiteral})\) is the value of \(xxx\) (condition 2a), and is clearly in \(L_{\text{V}}\) (condition 2b), by definition of \(I\). Condition 2c is satisfied by satisfaction of the second rule in \(R_{\text{RDF}}\) in \(I\).
3. Satisfaction of 3a and 3b follows straightforwardly from our assumptions on \(l\). This establishes the fact that \(I\) is an RDF-interpretation.

\(\text{(*)}\) Recall that, by assumption, \(ex:wellxml\) and \(ex:illxml\) are not used in \(R\). Therefore, changing satisfaction of atomic formulas involving \(ex:wellxml\) and \(ex:illxml\) does not affect satisfaction of \(R\). We assume that \(F_1(ex:wellxml)=l\) and \(F_1(ex:illxml)=l\) are distinct unique elements, i.e., no other constants is mapped to \(l\) and \(I\).

We define \(F_2(k)\) and \(F_1(l)\) as follows: For every typed literal of the form \([s, rdf:XMLLiteral]\) such that \(F_1([s, **] \text{if } rdf:XMLLiteral)=l\), if \([s, rdf:XMLLiteral]\) is well-typed, \(F_2(k)(l)(u)=t\) and \(F_1(l)(u)(t)=t\), otherwise \(F_2(k)(l)(u)=f\) and \(F_1(l)(u)(t)=f\) for every other object \(v\) in \(D_{\text{RDF}}\).

Consider \(R_{\text{RDF}}\). Satisfaction of \(R_{\text{RDF}}\) was established in the proof in the previous section. Satisfaction of the facts corresponding to the RDF axiomatic triples in \(I\) follows immediately from the definition of \(\text{common-RIF-RDF-interpretation}\) and the fact that \(I\) is an RDF-interpretation, and thus satisfies all RDF axiomatic triples.

Satisfaction of the axiomatic triples in \(R_{\text{RDF}}\) follows immediately from the definition of \(I\). Finally, satisfaction of the rules in \(R_{\text{RDF}}\) follows straightforwardly from the RDF semantic conditions 1, 2, and 3. This establishes the fact that \(I\) is a model of \(R_{\text{RDF}}\).

\(\text{Theorem}\) A list-free RDF-RDF combination \(\langle R, \{S_i, \ldots, S_n\} \rangle\) is \(R\)-\text{RDF-satisfiable} iff merge(\(\langle R_{\text{RDF}}, R, \text{tr}(S_j), \ldots, \text{tr}(S_n)\rangle\)) does not entail \(\text{rif:}y\).

\(\text{Proof}\) Recall that we assume \(\text{rif:}y\) does not occur in \(R\). If \(\langle R, \{S_i, \ldots, S_n\} \rangle\) is not \(R\)-RDF-satisfiable, then either merge(\(R, \text{tr}(S_j), \ldots, \text{tr}(S_n)\)) is not consistent, or condition 3a or 3b (see previous proof) is violated. In either case, \(\text{rif:}y\) is entailed. If \(\text{rif:}y\) is entailed, either merge(\(R_{\text{RDF}}, R, \text{tr}(S_j), \ldots, \text{tr}(S_n)\)) is inconsistent, which means merge(\(R, \text{tr}(S_j), \ldots, \text{tr}(S_n)\)) is not consistent and thus \(\langle R, \{S_i, \ldots, S_n\} \rangle\) is not RDF-satisfiable, or the body of the second or third rule in \(R_{\text{RDF}}\) is satisfied in every model, which means either condition 3a or 3b is violated, and so \(\langle R, \{S_i, \ldots, S_n\} \rangle\) is not RDF-satisfiable.

\(\text{9.1.5 Embedding RDFS Entailment}\)

We axiomatize the semantics of the RDFS(V) Vocabulary using the following RDF rules.

Similar to the RDF case, the set of RDFS finite-axiomatic triples is the smallest set such that:

- every RDFS axiomatic triple not of the form \(\text{ rdf}: \ldots \) \ldots \ldots where \(j\) is a positive integer, is an RDFS finite-axiomatic triple.
- the triples \(\text{rdf:} \ldots \text{rdf}:\text{domain} \text{rdf:} \ldots \text{rdf:} \text{range}\) are RDFS:Resource, for some positive integer \(k\) such that \(\text{rdf}: \ldots \text{rdf}:\text{domain} \text{rdf:} \ldots \text{rdf:} \text{range}\) do not occur in the context of \(C\), are RDFS finite-axiomatic triples, and if \(\text{rdf}: \ldots \text{rdf}:\text{domain} \text{rdf:} \ldots \text{rdf:} \text{range}\) occurs in the context of the combination \(C\), for some positive integer \(j\), then \(\text{rdf}: \ldots \text{rdf}:\text{domain} \text{rdf:} \ldots \text{rdf:} \text{range}\) are RDFS:Resource.

We assume that the unary predicate symbol \(\text{ex:}\text{well}\) is not used in the context of the given combination.
In the following theorems it is assumed that, in combinations $C < \langle S_1, \ldots, S_n \rangle$, $R$ does not have mentions of rdfs:Resource, $S_1, \ldots, S_n$ do not have mentions of rdfs:Resource beyond triples of the form $x : rdfs:type -> \text{rdfs:Literal}$, and entailed graphs $T$ and formulas $\phi$ do not have mentions of rdfs:Resource.

**Theorem** A RIF-RDFS-satisfiable list-free RIF-RDF combination $C < \langle S_1, \ldots, S_n \rangle$ is RIF-RDFS-entails and $R$ has no mentions of rdfs:Resource if and only if $\text{merge}(R, R^\text{RDFS}, \text{tr}(S_1), \ldots, \text{tr}(S_n))$ entails $\phi$. 

**Proof.** Consider any value $a$ in $I$ such that $a$ is in ICEXT($I$($C$)) and $a$ is not in LV. By definition, either $a$ is in LV, or $a$ is a model of $R^\text{RDFS}$.

Clearly, this change does not affect satisfaction of the RDF axiomatic triples and the semantic conditions 1 and 2. To see that condition 3 is still satisfied, consider

(*) In addition to the earlier assumptions about $I$, we assume that $\text{tr}(s'':\langle \text{rdf:Literal} \rangle) \text{ is not satisfied in } I$, for any typed literal of the form $(s, \text{rdfs:Literal})$ in $V_L$. We amend the definition of $I$ by changing the definitions of $LV$ and $IEXT$ to the following:

- $LV$ is union of the value spaces of all considered datatypes)
- $IEXT$ is the empty set.

Clearly, this change does not affect satisfaction of the RDF axiomatic triples and semantic conditions 1 and 2. To see that condition 3 is still satisfied, consider

(**) Consider $R^\text{RDFS}$. Satisfaction of $R^\text{RDFS}$ was established in the proof of the previous section. Satisfaction of the facts corresponding to the RDF axiomatic triples in $I$ follows immediately from the definition of common-RIF-RDF interpretation and the fact that $I$ is an RDFS-interpretation, and thus satisfies all RDF axiomatic triples.

Satisfaction of the 1st through the 2nd rule in $R^\text{RDFS}$ follow straightforwardly from the RDF semantic conditions 1 through 10. Satisfaction of the 13th rule follows from the fact that, given an ill-typed XML literal $x$, $\text{IL}(x)$ is in LV, by definition.

Theorem A list-free RIF-RDF combination $C < \langle S_1, \ldots, S_n \rangle$ is RIF-RDFS-satisfiable if and only if $\text{merge}(R, R^\text{RDFS}, \text{tr}(S_1), \ldots, \text{tr}(S_n))$ does not entail $\text{rdf:_literal}$. 

**Proof.** The theorem follows immediately from the previous theorem and the observations in the proof of the second theorem in the previous section. □

### 9.2 Embedding RIF-OWL 2 RL Combinations

It is known that expressive Description Logic languages such as OWL 2 DL cannot be straightforwardly embedded into typical rules languages such as RIF BLD [RIF-BLD] because of features such as disjunction and negation.

In this section we consider a subset of OWL 2 DL in RIF-OWL DL combinations, namely, the OWL 2 RL profile [OWL2-Profiles], and show how reasoning with RIF-OWL 2 RL combinations can be reduced to reasoning with RIF.

The embedding of RIF-OWL 2 RL combinations is not defined for combinations that include infinite OWL ontologies.

Since OWL 2 RL includes equality through ObjectMaxCardinality and DataMaxCardinality restrictions, as well as FunctionalObjectProperty and DataProperty restrictions, and there is no non-trivial interaction between such equality and the predicates in the RDF rules in the combination, embedding RIF-OWL 2 RL combinations into RIF requires equality. Therefore, the embedding presented in this appendix is not in RIF Core, even if the RIF
document in the combination is, if the ontologies in the combination do not contain any of the mentioned constructs, the embedding is in Core. Also, it is well-known that adding equality to a rules language does not increase its expressiveness in the absence of function symbols: one can replace equality = with a new binary predicate symbol, and add rules for reflexivity and the principle of substitutivity (also called the replacement property).

9.2.1 Embedding RIF DL-document formulas into RIF BLD

Recall that the semantics of frame formulas in DL-document formulas is different from the semantics of frame formulas in RIF documents. Nonetheless, DL-document formulas can be embedded into RIF documents, by translating frame formulas to predicate formulas. The mapping tr is the identity mapping on all RIF formulas, with the exception of frame formulas, as defined in the following table.

In the table, the mapping tr∗ is an injective function that maps constants to new constants, i.e., constants that are not used in the original document or its vicinity (i.e., imported, entailed or entailing formula). It “generates” a new constant from an existing one.

### Table: Mapping RIF DL-document formulas to RIF documents

<table>
<thead>
<tr>
<th>RIF Construct</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term t</td>
<td>tr(t)=t</td>
</tr>
<tr>
<td>Atomic formula φ that is not a frame formula</td>
<td>tr(φ)=φ</td>
</tr>
<tr>
<td>a[b₁→c₁ ... bₙ→cₙ], with na₂</td>
<td>tr[a(b₁→c₁ ... bₙ→cₙ)]=And(tr[a(b₁→c₁)] ... tr[a(bₙ→cₙ)])</td>
</tr>
<tr>
<td>a[b→c], where a and c are terms and b = rdf:type is a constant</td>
<td>tr[a(b→c)]=tr(b)(a,c)</td>
</tr>
<tr>
<td>a[rdf:type → c], where a is a term and c is a constant</td>
<td>tr[a(rdf:type → c)]=tr(c)(a)</td>
</tr>
<tr>
<td>a[∀cᵢ, where a is a term and c is a constant</td>
<td>tr[a[∀cᵢ]]=tr(∀cᵢ)(a)</td>
</tr>
<tr>
<td>b#c, where a, b are constants</td>
<td>tr[b#c]=Forall ?x (tr(b)(?x) : tr(b)(?x))</td>
</tr>
<tr>
<td>Exists ?y₁ ... ?yₙ(φ)</td>
<td>tr[Exists ?y₁ ... ?yₙ(φ)]=Exists ?y₁ ... ?yₙ(tr(φ))</td>
</tr>
<tr>
<td>And(φ₁ ... φₙ)</td>
<td>tr[And(φ₁ ... φₙ)]=And(tr(φ₁) ... tr(φₙ))</td>
</tr>
<tr>
<td>Or(φ₁ ... φₙ)</td>
<td>tr[Or(φ₁ ... φₙ)]=Or(tr(φ₁) ... tr(φₙ))</td>
</tr>
<tr>
<td>?q₁ : ?q₂</td>
<td>tr[?q₁ : ?q₂]=tr(?q₁) :: tr(?q₂)</td>
</tr>
<tr>
<td>Forall ?y₁ ... ?yₙ(φ)</td>
<td>tr[Forall ?y₁ ... ?yₙ(φ)]=Forall ?y₁ ... ?yₙ(tr(φ))</td>
</tr>
<tr>
<td>Group(?q₁ ... ?qₙ)</td>
<td>tr[Group(?q₁ ... ?qₙ)]=Group(tr(?q₁) ... tr(?qₙ))</td>
</tr>
<tr>
<td>Document(directory₁ ... directoryₙ)</td>
<td>tr[Document(directory₁ ... directoryₙ)]=Document(directory₁ ... directoryₙ)</td>
</tr>
</tbody>
</table>

For the purpose of making statements about this embedding, we define a notion of entailment for DL-document formulas.

**Definition.** A RIF-BLD DL-document formula $R$ *d-entails* a DL-condition $q$ if for every di-semantic multi-structure $I$ that is a model of $R$ it holds that $\text{Val}(q)=t$.

The following lemma establishes faithfulness with respect to entailment of the embedding.

### RIF-BLD DL-document formula Lemma 2: RIF-BLD DL-document formula $R$ d-entails a DL-condition $q$ if and only if tr(R) d-entails tr(q).

**Proof.** We prove both directions by contraposition.

$\implies$ Assume tr(R) does not entail tr(q). This means there is some semantic multi-structure $I$ that is a model of tr(R), but $I$ is not a model of tr(φ). By definition of the embedding, for every $X \subseteq D$ such that tr(φ)(t) is true and $I$ is a model of $\text{Val}(q)=t$, we define $\text{Val}(q)=t$.

For every constant $c$, with $c'=tr(c)$, $\text{Val}(c)=t$ if $\text{Val}(c')=t$ for some constant $c'$. This is obtained from the argument for $X \subseteq D$ such that tr(φ)(t) is true and $I$ is a model of $\text{Val}(q)=t$, we define $\text{Val}(q)=t$.

Consider the semantic multi-structure $I$, which is obtained from $F$ as follows: $F' = \langle TV, D, D_{ind}, D_{class}, I_k, I_v, I_f, I_{isa}, I_{trans}, I_{htr}, I_{htf}, I_{hub}, I_{lsa}, I_{external}, I_{htrn} \rangle$ is not a model of tr(R).

We define $PC, P_{isa}, P_{hub}$, and $P_{sub}$ as follows:

- For every constant $c$, with $c'=tr(c)$, $\text{Val}(c)=t$ if $\text{Val}(c')=t$ for some constant $c'$.

Consider the semantic multi-structure $I$, which is obtained from $F$ as follows: $F' = \langle TV, D, D_{ind}, D_{class}, I_k, I_v, I_f, I_{isa}, I_{trans}, I_{htr}, I_{htf}, I_{hub}, I_{lsa}, I_{external}, I_{htrn} \rangle$ is not a model of tr(R).

We define $PC, P_{isa}, P_{hub}$, and $P_{sub}$ as follows:

- For every constant $c$, with $c'=tr(c)$, $\text{Val}(c)=t$ if $\text{Val}(c')=t$ for some constant $c'$.

Consider the semantic multi-structure $I$, which is obtained from $F$ as follows: $F' = \langle TV, D, D_{ind}, D_{class}, I_k, I_v, I_f, I_{isa}, I_{trans}, I_{htr}, I_{htf}, I_{hub}, I_{lsa}, I_{external}, I_{htrn} \rangle$ is not a model of tr(R).

We define $PC, P_{isa}, P_{hub}$, and $P_{sub}$ as follows:

- For every constant $c$, with $c'=tr(c)$, $\text{Val}(c)=t$ if $\text{Val}(c')=t$ for some constant $c'$.

Consider the semantic multi-structure $I$, which is obtained from $F$ as follows: $F' = \langle TV, D, D_{ind}, D_{class}, I_k, I_v, I_f, I_{isa}, I_{trans}, I_{htr}, I_{htf}, I_{hub}, I_{lsa}, I_{external}, I_{htrn} \rangle$ is not a model of tr(R).

We define $PC, P_{isa}, P_{hub}$, and $P_{sub}$ as follows:

- For every constant $c$, with $c'=tr(c)$, $\text{Val}(c)=t$ if $\text{Val}(c')=t$ for some constant $c'$.
Observe that $R$ and $\phi$ do not include predicate formulas involving derived constant symbols $tr'(c)$. The remainder of the proof is analogous to the ($\Rightarrow$) direction. □

### 9.2.2 Embedding OWL 2 RL into RIF BLD

The embedding of OWL 2 RL into RIF BLD has two stages: normalization and embedding.

The OWL 2 syntax is given in terms of a Structural Specification, and there is a functional-style syntax that is a serialization of this Structural Specification. For convenience, normalization and embedding in this section are done in terms of the functional-style syntax. That is, the normalization mapping takes as input a functional-style syntax ontology document and produces a normalized ontology document. The embedding mapping takes as input a normalized ontology document and produces an RIF document. We refer to Section 4.2 of [OWL2-Profiles](http://www.w3.org/TR/2012/PER-owl2-profiles/) for the specification of the OWL 2 RL syntax.

#### 9.2.2.1 Normalization of OWL 2 RL

Normalization splits the OWL axioms so that the later mapping to RIF of the individual axioms results in rules. Additionally, it simplifies the axioms and removes annotations.

It is assumed that the normalization process is preceded by a simplification process that removes all namespace prefixes, turns all CURIEs and relative IRIs into absolute IRIs, and removes all annotations, import statements, entity declarations, and annotation axioms.

We note here that, strictly speaking, simplified OWL 2 RL ontologies are not OWL 2 RL ontologies in the general case, because certain entity declarations are required (e.g., those distinguishing data from object properties). It is assumed that such entity declarations are present implicitly, i.e., they do not appear explicitly in the simplified ontology, but they are known. We also note that removing import statements in the simplification does not prohibit importing ontologies in practice; since combinations contain sets of ontologies, all imported ontologies may be added to these sets. The normalization mapping $tr$ takes as input a simplified ontology $O$ and produces an equivalent normalized ontology $O'$.

The names of variables used in the mapping generally correspond to the names of productions in the OWL 2 RL grammar.

<table>
<thead>
<tr>
<th>#</th>
<th>Statement</th>
<th>Normalized Statement</th>
<th>Condition on translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$tr$: Ontology{ [ ontologyIRI [ versionIRI ] ] axiom; ... axiom; }</td>
<td>$tr$: Ontology{ $tr$(axiom1) ... $tr$(axiomn) }</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$tr$: SubClassOf(subClassExpression ... ObjectIntersectionOf{ superClassExpression1 ... superClassExpressionm } ... superClassExpressionn) )</td>
<td>$tr$(SubClassOf(subClassExpression ... superClassExpression1 ... superClassExpressionm ... superClassExpressionn))</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$tr$: SubClassOf(subClassExpression1 ObjectComplementOf{subClassExpression1})</td>
<td>$tr$(SubClassOf(ObjectIntersectionof{subClassExpression1 subClassExpression2 owl:Nothing}))</td>
<td>$X$ is a superClassExpression that does not contain ObjectIntersectionOf or ObjectComplementOf</td>
</tr>
<tr>
<td>4</td>
<td>$tr$(SubClassOf{subClassExpression X})</td>
<td>SubClassOf{subClassExpression X}</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$tr$: EquivalentClasses( equivClassExpression1 ... equivClassExpressionm )</td>
<td>$tr$(SubClassOf(equivClassExpression1 equivClassExpression2)) ... $tr$(SubClassOf(equivClassExpressionm-1 equivClassExpressionm)) $tr$(SubClassOf(equivClassExpression1 equivClassExpression2 ... equivClassExpressionm))</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$tr$: DisjointClasses( subClassExpression1 ... subClassExpressionm )</td>
<td>$tr$(SubClassOf(ObjectIntersectionOf{subClassExpression1 subClassExpression2 owl:Nothing})) ... $tr$(SubClassOf(ObjectIntersectionOf{subClassExpression1 subClassExpressionm-1 subClassExpressionm owl:Nothing})) $tr$(SubClassOf(ObjectIntersectionOf{subClassExpression1 subClassExpressionm owl:Nothing}))</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$tr$: SubObjectPropertyOf( subPropertyExpression superPropertyExpression )</td>
<td>SubObjectPropertyOf( subPropertyExpression superPropertyExpression )</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>$tr$: SubDataPropertyOf( subPropertyExpression superPropertyExpression )</td>
<td>SubDataPropertyOf( subPropertyExpression superPropertyExpression )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 9 | `\( \text{trN} (\text{EquivalentObjectProperties(} \text{ObjectPropertyExpression}_1, \ldots, \text{ObjectPropertyExpression}_m \text{)}) \)`  
  `\( \text{trN} (\text{SubObjectPropertyOf(} \text{ObjectPropertyExpression}_1, \ldots, \text{ObjectPropertyExpression}_m \text{))} \)` |
| 10 | `\( \text{trN} (\text{EquivalentDataProperties(} \text{DataPropertyExpression}_1, \ldots, \text{DataPropertyExpression}_m \text{)}) \)`  
  `\( \text{trN} (\text{SubDataPropertyOf(} \text{PropertyExpression}_1, \ldots, \text{PropertyExpression}_m \text{))} \)` |
| 11 | `\( \text{trN} (\text{DisjointObjectProperties(} \text{ObjectPropertyExpression}_1, \ldots, \text{ObjectPropertyExpression}_m \text{)}) \)`  
  `\( \text{DisjointDataProperties(} \text{ObjectPropertyExpression}_1, \ldots, \text{ObjectPropertyExpression}_m \text{)}) \)` |
| 12 | `\( \text{trN} (\text{DisjointDataProperties(} \text{DataPropertyExpression}_1, \ldots, \text{DataPropertyExpression}_m \text{)}) \)`  
  `\( \text{DisjointObjectProperties(} \text{ObjectPropertyExpression}_1, \ldots, \text{ObjectPropertyExpression}_m \text{)}) \)` |
| 13 | `\( \text{trN} (\text{InverseObjectProperties(} \text{PropertyExpression}_1, \text{PropertyExpression}_2 \text{)}) \)`  
  `\( \text{InverseObjectProperties(} \text{PropertyExpression}_1, \text{PropertyExpression}_2 \text{)}) \)` |
| 14 | `\( \text{trN} (\text{ObjectPropertyDomain(} \text{PropertyExpression}, \text{superClassExpression} \text{)}) \)`  
  `\( \text{trN} (\text{SubClassOf(} \text{ObjectSomeValuesFrom(} \text{PropertyExpression \owl:Thing}, \text{superClassExpression} \text{)}) \)` |
| 15 | `\( \text{trN} (\text{DataPropertyDomain(} \text{DataProperty}, \text{superClassExpression} \text{)}) \)`  
  `\( \text{trN} (\text{SubClassOf(} \text{ObjectSomeValuesFrom(} \text{DataProperty \owl:Thing}, \text{superClassExpression} \text{)}) \)` |
| 16 | `\( \text{trN} (\text{ObjectPropertyRange(} \text{ObjectInverseOf(} \text{Property}, \text{superClassExpression} \text{)}) \)`  
  `\( \text{trN} (\text{SubClassOf(} \text{ObjectSomeValuesFrom(} \text{Property \owl:Thing}, \text{superClassExpression} \text{)}) \)` |
| 17 | `\( \text{trN} (\text{ObjectPropertyRange(} \text{Property}, \text{superClassExpression} \text{)}) \)`  
  `\( \text{trN} (\text{SubClassOf(} \text{ObjectSomeValuesFrom(} \text{ObjectInverseOf(} \text{Property} \owl:Thing}, \text{superClassExpression} \text{)}) \)` |
| 18 | `\( \text{trN} (\text{DataPropertyRange(} \text{DataProperty}, \text{superClassExpression} \text{)}) \)`  
  `\( \text{trN} (\text{SubClassOf(} \text{owl:Thing}, \text{DataAllValuesFrom(} \text{DataProperty \superClassExpression} \text{)}) \)` |
| 19 | `\( \text{trN} (\text{FunctionalObjectProperty(} \text{PropertyExpression} \text{)}) \)`  
  `\( \text{FunctionalObjectProperty(} \text{PropertyExpression} \text{)}) \)` |
| 20 | `\( \text{trN} (\text{FunctionalDataProperty(} \text{DataProperty} \text{)}) \)`  
  `\( \text{FunctionalDataProperty(} \text{DataProperty} \text{)}) \)` |
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>InverseFunctionalObjectProperty( PropertyExpression )</td>
</tr>
<tr>
<td>22</td>
<td>IrreflexiveObjectProperty( PropertyExpression )</td>
</tr>
<tr>
<td>23</td>
<td>SymmetricObjectProperty( PropertyExpression )</td>
</tr>
<tr>
<td>24</td>
<td>AsymmetricObjectProperty( PropertyExpression )</td>
</tr>
<tr>
<td>25</td>
<td>TransitiveObjectProperty( PropertyExpression )</td>
</tr>
<tr>
<td>26</td>
<td>DatatypeDefinition( ... )</td>
</tr>
<tr>
<td>27</td>
<td>HasKey( ... )</td>
</tr>
<tr>
<td>28</td>
<td>SameIndividual( Individual1, Individual2, ..., Individualn )</td>
</tr>
<tr>
<td>29</td>
<td>DifferentIndividuals( Individual1, ..., Individualn )</td>
</tr>
<tr>
<td>30</td>
<td>ClassAssertion( superClassExpression Individual )</td>
</tr>
<tr>
<td>31</td>
<td>ObjectPropertyAssertion( ObjectPropertyExpression source target )</td>
</tr>
<tr>
<td>32</td>
<td>NegativeObjectPropertyAssertion( ObjectPropertyExpression source target )</td>
</tr>
<tr>
<td>33</td>
<td>DataPropertyAssertion( DataProperty source target )</td>
</tr>
<tr>
<td>34</td>
<td>SubClassOf( ObjectOneOf( source ) DataHasValue(DataProperty target) )</td>
</tr>
</tbody>
</table>
We note that normalized OWL 2 RL ontologies are not necessarily OWL 2 RL ontologies, since owl:Thing may appear in subclass expressions, as a result of the transformation of DataPropertyRange axioms.

The following lemma establishes the fact that, for the purpose of entailment, the ontologies in a combination may be replaced with their normalization.

**Normal Form Lemma** Given a combination C = (C₁,...,Cₙ), where C₁,...,Cₙ are simplified OWL 2 RL ontologies that do not import ontologies, C RIF-OWL Direct-entails φ iff C = (C₁(tr₁),...,Cₙ(trₙ)) > RIF-OWL Direct-entails φ.

**Proof.** We prove both directions by contradiction: if the entailment does not hold on the one side, we show that it also does not hold on the other.

(⇒) Assume C does not RIF-OWL Direct-entail φ. This means there is a common-RIF-OWL Direct-interpretation (I, []) that is a model of C, but I is not a model of φ. By the definition of satisfaction of axioms and assertions in Section 2.3 and the interpretation of object property, data range, and class expressions in Section 2.4 in [OWL2-Semantics] it is easy to verify that, if for every axiom d appearing in {C₁,...,Cₙ}, I satisfies tr₁(d), then I satisfies C₁,..., Cₙ, and thus (I, []) satisfies C. Since I is not a model of φ, C does not RIF-OWL Direct-entail φ.

(⇐) Assume C does not RIF-OWL Direct-entail φ. This means there is a common-RIF-OWL Direct-interpretation (I, []) that is a model of C, but I is not a model of φ. It is easy to verify, by the definition of satisfaction of axioms and assertions in Section 2.3 and the interpretation of object property, data range, and class expressions in Section 2.2 in [OWL2-Semantics], that I satisfies tr₁(C₁),..., and trₙ(Cₙ). So, (I, []) is a model of C, and thus C does not RIF-OWL Direct-entail φ. □

### 9.2.2.2 Embedding Normalized OWL 2 RL

We now proceed with the embedding of normalized OWL 2 RL ontologies into RIF DL-document formulas. The embedding function tr₁ takes as input a normalized OWL 2 RL ontology and returns a RIF-BLD DL-document formula. The embeddings of IRIs and literals is as defined in Section 9.1.1 Embedding Symbols. It is assumed that the Vocabulary V of the ontologies includes all the constants used in the RIF documents and condition formulas under consideration.

<table>
<thead>
<tr>
<th>#</th>
<th>Normalized OWL</th>
<th>RIF-BLD DL-document formula</th>
<th>Condition on translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>tr₁ Ontology( axiom₁ )... axiomₙ )</td>
<td>Document(Group( tr₁(axiom₁) ... tr₁(axiomₙ) ))</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>tr₁ SubClassOf(subClassExpression superClassExpression)</td>
<td>tr₁(subClassExpression,superClassExpression)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>tr₁(subClassExpression, [Object</td>
<td>Data]AllValuesFrom(property₁...[Object</td>
<td>Data]AllValuesFrom(propertyₙ X ...) ))</td>
</tr>
<tr>
<td>3a</td>
<td>tr₁(subClassExpression, X)</td>
<td>Forall 7x (tr₁(X, ?y₁) :- And( tr₁(subClassExpression, ?x) tr₁(X, ?y₁) ) )</td>
<td>X is not an [Object</td>
</tr>
<tr>
<td>4</td>
<td>tr₁(subClassExpression, [Object</td>
<td>Data]MaxCardinality(property₁...[Object</td>
<td>Data]MaxCardinality(propertyₙ PropertyExpression ClassExpression) ... ))</td>
</tr>
<tr>
<td>4a</td>
<td>tr₁(subClassExpression, [Object</td>
<td>Data]MaxCardinality(PropertyExpression ClassExpression))</td>
<td>Forall 7x ?y (rif:error :- And( tr₁(subClassExpression, ?x) tr₁(PropertyExpression, ?x, ?y) tr₁(ClassExpression, ?y) ) )</td>
</tr>
<tr>
<td>5</td>
<td>tr₁(subClassExpression, [Object</td>
<td>Data]AllValuesFrom(property₁...[Object</td>
<td>Data]AllValuesFrom(propertyₙ PropertyExpression ClassExpression) ... ))</td>
</tr>
<tr>
<td>Rule</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5a</td>
<td>For all ( x ), ( y_1 ), ( y_2 ) ( (y_1 = y_2) \rightarrow (\text{tr}(\text{subClassExpression}, x) \land \text{tr}(\text{propertyExpression}, y_1) \land \text{tr}(\text{propertyExpression}, y_2)) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>( \text{tr}(A, x) \rightarrow \neg x \text{[rdf:type]} \rightarrow \text{tr}(A) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>( \text{And}(\text{tr}(\text{classExpression}_1, x) \ldots \text{tr}(\text{classExpression}_n, x)) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>( \text{Or}(\text{tr}(\text{classExpression}_1, x) \ldots \text{tr}(\text{classExpression}_n, x)) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>( \text{Forall}(x = \text{tr}(\text{individual}_1) \ldots \text{tr}(\text{individual}_n)) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>( \text{Exists}(y \text{[And}(\text{tr}(\text{propertyExpression}, x), \text{tr}(\text{classExpression}, y))]) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>( \text{Forall}(x, y) \rightarrow \text{tr}(x) \rightarrow \text{tr}(y) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>( \text{Forall}(x, y) \rightarrow \neg \text{tr}(x) \rightarrow \text{tr}(y) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>( \text{tr}(\text{propertyExpression}, x, \text{tr}(\text{value})) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>( \text{Forall}(x, y_1 \ldots y_n) \text{[And}(\text{tr}(\text{subObjectPropertyOf}(\text{objectPropertyChain}(\text{propertyExpression}_1 \ldots \text{propertyExpression}_n))) \land \text{tr}(\text{propertyExpression}_1)) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>( \text{Forall}(x, y) \text{[And}(\text{tr}(\text{propertyExpression}_1, y) \land \text{tr}(\text{propertyExpression}_2, y))) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>( \text{Forall}(x, y) \text{[And}(\text{tr}(\text{disjoint}(\text{objectData}) \text{properties}(\text{propertyExpression}_1, \text{propertyExpression}_2))) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>( \text{Forall}(x, y) \text{[And}(\text{tr}(\text{inverses}(\text{objectData}) \text{properties}(\text{propertyExpression}_1, \text{propertyExpression}_2))) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>( \text{Forall}(x, y) \text{[And}(\text{tr}(\text{functional}(\text{objectData}) \text{properties}(\text{propertyExpression}_1))) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>( \text{Forall}(x, y_1 \ldots y_n) \text{[And}(\text{tr}(\text{inverse-func}(\text{objectData}) \text{properties}(\text{propertyExpression}_1))) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>( \text{Forall}(x, y) \text{[And}(\text{tr}(\text{irreflexive}(\text{objectData}) \text{properties}(\text{propertyExpression}_1))) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>( \text{Forall}(x, y) \text{[And}(\text{tr}(\text{symmetric}(\text{objectData}) \text{properties}(\text{propertyExpression}_1))) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>( \text{Forall}(x, y) \text{[And}(\text{tr}(\text{asymmetric}(\text{objectData}) \text{properties}(\text{propertyExpression}_1))) )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Besides the embedding in the previous table, we also need an axiomatization of some of the aspects of the OWL 2 Direct Semantics, e.g., separation between individual and datatype domains. This axiomatization is defined relative to an OWL Vocabulary \( V \), which includes all well-typed literals used in the rules, and a datatype map \( D \), which includes all considered datatypes. In the table, for a given datatype \( d \), \( L^v(d) \) is the lexical-to-value mapping of \( d \).

### Normalized Combination Embedding Lemma

Given a datatype map \( D \) conforming to \( T \), a RIF-OWL DL-combination \( C = \langle R(D_1), \ldots, D_n \rangle \), where \( \langle D_2, \ldots, D_n \rangle \) is an imports-closed set of normalized OWL 2 DL ontologies with vocabulary \( V \), RIF-OWL Direct-entails a DL-condition \( \varphi \) with respect to \( D \) iff merge(\( \langle R' \rangle, R^\text{DL-direct}(\langle V \rangle), \text{tr}(D_1), \ldots, \text{tr}(D_n) \rangle \)) \( \text{dl-entails} \) \( \varphi \), where \( R' \) is like \( R \), except that every well-typed \( C \) in the form \( \text{ar} \) has been replaced with \( \{ \text{ar} \rightarrow \text{b} \} \).

#### Proof

We prove both directions by contraposition.

1. **If** \( R^\text{DL-direct}(\langle V \rangle) \) \( \text{dl-entails} \) \( \varphi \), then \( \text{merge}(\langle R', \text{R}^\text{DL-direct}(\langle V \rangle), \text{tr}(D_1), \ldots, \text{tr}(D_n) \rangle) \) \( \text{dl-entails} \) \( \varphi \).

   - **Proof.**
     - For each \( R \) in the form \( \{ \text{ar} \rightarrow \text{b} \} \), we define
       
       \[
       D^\text{RIF-OWLDL-comb}(D_1, D_2, \ldots, D_n) = \text{merge}(\langle R', \text{R}^\text{DL-direct}(\langle V \rangle), \text{tr}(D_1), \ldots, \text{tr}(D_n) \rangle)
       \]

     - **Case A.** If \( R \) is a RIF-OWLDL-combination, then
       
       \[
       D^\text{RIF-OWLDL-comb}(D_1, D_2, \ldots, D_n) = \text{merge}(\langle R', \text{R}^\text{DL-direct}(\langle V \rangle), \text{tr}(D_1), \ldots, \text{tr}(D_n) \rangle)
       \]

     - **Case B.** If \( R \) is a RIF-OWLDL-combination, then
       
       \[
       D^\text{RIF-OWLDL-comb}(D_1, D_2, \ldots, D_n) = \text{merge}(\langle R', \text{R}^\text{DL-direct}(\langle V \rangle), \text{tr}(D_1), \ldots, \text{tr}(D_n) \rangle)
       \]

     - **Case C.** If \( R \) is a RIF-OWLDL-combination, then
       
       \[
       D^\text{RIF-OWLDL-comb}(D_1, D_2, \ldots, D_n) = \text{merge}(\langle R', \text{R}^\text{DL-direct}(\langle V \rangle), \text{tr}(D_1), \ldots, \text{tr}(D_n) \rangle)
       \]

     - **Case D.** If \( R \) is a RIF-OWLDL-combination, then
       
       \[
       D^\text{RIF-OWLDL-comb}(D_1, D_2, \ldots, D_n) = \text{merge}(\langle R', \text{R}^\text{DL-direct}(\langle V \rangle), \text{tr}(D_1), \ldots, \text{tr}(D_n) \rangle)
       \]

     - **Case E.** If \( R \) is a RIF-OWLDL-combination, then
       
       \[
       D^\text{RIF-OWLDL-comb}(D_1, D_2, \ldots, D_n) = \text{merge}(\langle R', \text{R}^\text{DL-direct}(\langle V \rangle), \text{tr}(D_1), \ldots, \text{tr}(D_n) \rangle)
       \]

   - **Lemma.** Given a set of datatypes \( D \), the lexical-to-value mapping \( L^v(D) \) is defined as follows:
     - \( D^\text{RIF-OWLDL-comb}(D_1, D_2, \ldots, D_n) = \text{merge}(\langle R', \text{R}^\text{DL-direct}(\langle V \rangle), \text{tr}(D_1), \ldots, \text{tr}(D_n) \rangle)
   
   - **Theorem.** If \( R \) is a RIF-OWLDL-combination, then
     
     \[
     D^\text{RIF-OWLDL-comb}(D_1, D_2, \ldots, D_n) = \text{merge}(\langle R', \text{R}^\text{DL-direct}(\langle V \rangle), \text{tr}(D_1), \ldots, \text{tr}(D_n) \rangle)
     \]

   - **Conclusion.** If \( R \) is a RIF-OWLDL-combination, then
     
     \[
     D^\text{RIF-OWLDL-comb}(D_1, D_2, \ldots, D_n) = \text{merge}(\langle R', \text{R}^\text{DL-direct}(\langle V \rangle), \text{tr}(D_1), \ldots, \text{tr}(D_n) \rangle)
     \]

   - **End of Proof.**
RIF RDF and OWL Compatibility

Theorem

Vocabulary

Observe that the mapping tr() does not distinguish between frame formulas of the form \( \text{rdfs:type} \rightarrow b \) and membership formulas \( a \in b \). We may thus safely
assume that $R$ has no occurrences of the latter. Then, by the Normalized Combination Embedding Lemma,
merge($\langle R, R_{\text{OWL-Direct}}(V), \text{tr}(\text{tr}(O_1)), ..., \text{tr}(\text{tr}(O_n)) \rangle$) $\triangleright$-entails $\phi$ iff $\langle R_{\text{OWL-Direct}}(V), \text{tr}(\text{tr}(O_1)), ..., \text{tr}(\text{tr}(O_n)) \rangle$ $\triangleright$-entails $\phi$.

Finally, by the Normalization Lemma,
$\langle R, \text{tr}(O_1), ..., \text{tr}(O_n) \rangle$ $\triangleright$-entails $\phi$ iff $\text{C} = \langle R, \{O_1, ..., O_n\} \rangle$ $\triangleright$-entails $\phi$.

This chain of equivalences establishes the theorem. \hfill $\Box$

**Theorem** Given a datatype map $D$ conforming with $T$, a RIF-OWL DL-combination $\langle R, \{O_1, ..., O_n\} \rangle$, where $\{O_1, ..., O_n\}$ is an imports-closed set of OWL 2 RL ontologies with Vocabulary $V$, is RIF-OWL Direct-satisfiable with respect to $D$ iff $\text{tr}(\text{merge}(\langle R, R_{\text{OWL-Direct}}(V), \text{tr}(\text{tr}(O_1)), ..., \text{tr}(\text{tr}(O_n)) \rangle))$ does not entail $\text{rif:}\text{error}$.

**Proof.** The theorem follows immediately from the previous theorem and the observation that a combination (respectively, document) is RIF-OWL Direct-satisfiable (respectively, has a model) if and only if it does not entail the condition formula "a"="b". \hfill $\Box$

**10 Appendix: Change log (Informative)**

Changes since the 11 May 2010 Proposed Recommendation.

In the table in Section 9.2.2.2: The expression $\text{tr}(X, ?y)$ has been added to the third row, second column; omitting this expression had been an oversight. Rows 3–5 did not account for inverse properties; this had been rectified. For the purpose of understandability, rows 3a, 4a, 5a have been added to make the case $n=0$ of rows 3, 4, 5 explicit.

Changes since the 22 June 2010 Recommendation.

Added a clarification to Section 9 on the restriction for subclass preventing embedding.

**11 End Notes**

**RDF URI References:** There are certain RDF URI references that are not IRIs (e.g., those containing spaces). It is possible to use such RDF URI references in RDF graphs that are combined with RIF rules. However, such URI references cannot be represented in RIF rules and their use in RDF is discouraged.

**Generalized RDF graphs:** Standard RDF graphs, as defined in [RDF-Concepts], do not allow the use of literals in subject and predicate positions and blank nodes in predicate positions. The RDF Core working group has listed two issues questioning the restrictions that literals may not occur in subject and blank nodes may not occur in predicate positions in triples. Anticipating lifting of these restrictions in a possible future version of RDF, we use the more liberal notion of generalized RDF graph. We note that the definitions of interpretations, models, and entailment in the RDF Semantics document [RDF-Semantics] also apply to such generalized RDF graphs.

We note that every standard RDF graph is a generalized RDF graph. Therefore, our definition of combinations applies to standard RDF graphs as well.

We note also that the notion of generalized RDF graphs is more liberal than the notion of RDF graphs used by SPARQL: generalized RDF graphs additionally allow blank nodes and literals in predicate positions.