W3C Proposed Edited Recommendation

RIF Core Dialect

W3C Proposed Edited Recommendation 11 December 2012

This version:
http://www.w3.org/TR/2012/PER-rif-core-20121211/
Latest version:
http://www.w3.org/TR/rif-core/
Previous version:
http://www.w3.org/TR/2010/REC-rif-core-20100622/
Editors:
Harold Boley, National Research Council Canada
Gary Hallmark, Oracle Corporation
Michael Kifer, State University of New York at Stony Brook, USA
Adrian Paschke, Freie Universitaet Berlin
Axel Polleres, DERI
Dave Reynolds, Hewlett-Packard Laboratories, Bristol UK

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

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

Abstract

This document, developed by the Rule Interchange Format (RIF) Working Group, specifies RIF-Core, a common subset of RIF-BLD and RIF-PRD based on RIF-DTB 1.0. The RIF-Core presentation syntax and semantics are specified by restriction in two different ways. First, RIF-Core is specified by restricting the syntax and semantics of RIF-BLD, and second, by restricting RIF-PRD. The XML serialization syntax of RIF-Core is specified by a mapping from the presentation syntax. A normative XML schema is also provided.

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May Be Superseded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 (this document)
3. RIF Basic Logic Dialect
4. RIF Production Rule Dialect
5. RIF Framework for Logic Dialects
6. RIF Datatypes and Built-Ins 1.0
7. RIF RDF and OWL Compatibility
8. OWL 2 RL in RIF
9. RIF Combination with XML data
10. RIF In RDF
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 Proposed Edited Recommendation

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.

Table of Contents

1 Overview

This specification describes ***RIF-Core*** (the Core dialect of the Rule Interchange Format). From a theoretical perspective, RIF-Core corresponds to the language of definite Horn rules without function symbols (often called 'Datalog') with a standard first-order semantics. RIF-Core thus is a subset of RIF-BLD [RIF-BLD]. At the same time, RIF-Core is a language of production rules where conclusions are interpreted as assert actions. RIF-Core thus also is a subset of RIF-PRD [RIF-PRD]. Moreover, RIF-Core is based on built-in functions and predicates over selected XML Schema datatypes, as specified in RIF-DTB 1.0 [RIF-DTB]. The common subset of RIF-BLD and RIF-PRD is specified based on RIF-DTB 1.0.

Syntactically, RIF-Core has a number of Datalog extensions to support features such as objects and frames as in F-logic [KLW95], internationalized resource identifiers (or IRIs, defined by [RFC-3987]) as identifiers for concepts, and XML Schema datatypes [XML-SCHEMA2]. In addition, RIF RDF and OWL Compatibility [RIF-RDF+OWL] defines the syntax and semantics of integrated RIF-Core/RDF and RIF-Core/OWL languages. These features make RIF-Core a Web-aware language. However, it should be kept in mind that RIF is designed to enable interoperability among rule languages in general, and its uses are not limited to the Web.

RIF-Core is defined as a specialization of RIF-BLD (hence of [RIF-FLD], making it a starting point of the RIF extensibility framework). It is a syntactic subset of RIF-BLD, so that a well-formed RIF-Core formula (including document and condition formulas) is also a well-formed RIF-BLD formula.

RIF-Core is also a syntactic subset of [RIF-PRD]. It is intended that a RIF-PRD consumer can treat a RIF-Core document as if it was a RIF-PRD rule set while it also conforms to the normative RIF-Core first order semantics. However, due to the presence of builtin functions and predicates there are rule sets in the syntactic intersection of RIF-PRD and RIF-BLD which would not terminate under RIF-PRD semantics. We therefore define a notion of safe RIF-Core rules, which is a subset of RIF-Core rules that can be executed using a forward chaining strategy, and we define conformance in terms of such safe rules. These notions of safeness and conformance are defined formally in section 5 Conformance and Safeness.

RIF-Core is not the *maximal* common subset of RIF-BLD and RIF-PRD. It omits some features from the intersection which do not significantly add to the expressiveness of the language and are judged to be not widely supported by rule languages.

To give a preview, here is a simple complete RIF-Core example deriving a ternary relation from its inverse.

Example 1 (An introductory RIF-Core example).

A rule can be written in English to derive buy relationships from the `sell` relationships that are stored as facts (e.g., as exemplified by the English statements below):

- *A buyer buys an item from a seller if the seller sells the item to the buyer.*
- *John sells LeRif to Mary.*

The fact *Mary buys LeRif from John* can be logically derived by a *modus ponens* argument. Assuming Web IRIs for the predicates buy and `sell`, as well as for the individuals `John`, `Mary`, and `LeRif`, the above English phrase can be represented in RIF-Core Presentation Syntax as follows.

```
Document(
  Prefix(cpt <http://example.com/concepts#>)
  Prefix(ppl <http://example.com/people#>)
  Prefix(bks <http://example.com/books#>)

  Group
  (
    Forall ?Buyer ?Item ?Seller (
        cpt:buy(?Buyer ?Item ?Seller) :- cpt:sell(?Seller ?Item ?Buyer)
    )

    cpt:sell(ppl:John bks:LeRif ppl:Mary)
  )
)
```

For the interchange of documents containing such rules (and facts), an equivalent RIF-Core XML syntax is provided in this specification. To formalize their meaning, a RIF-Core Semantics is specified.

This document assumes familiarity with [RIF-BLD] or [RIF-PRD], as RIF-Core is derived from these documents via syntactic restrictions.

2 RIF-Core Presentation Syntax

Like RIF-BLD and RIF-PRD, RIF-Core has both a ***presentation syntax*** and an ***XML syntax***. It is defined in "mathematical English," a special form of English for communicating mathematical definitions, examples, etc. and by an EBNF syntax. The mathematical English is normative, the EBNF is not normative; both instances of the presentation syntax are not intended to be a concrete syntax for RIF-Core. The English presentation syntax deliberately leaves out details such as the delimiters of the various syntactic components, escape symbols, parenthesizing, precedence of operators, and the like. Since RIF is an interchange format, it uses XML, and only XML, as its concrete syntax. RIF-Core conformance is described in terms of semantics-preserving mappings.

Since RIF-Core is a syntactic subset of RIF-BLD, this section defines the presentation syntax of RIF-Core as a restriction on the presentation syntax of RIF-BLD.

2.1 Alphabet of RIF-Core

The ***alphabet*** of the presentation language of RIF-Core is the alphabet of the RIF-BLD presentation language with the exclusion of the symbol ## (subclass) and the set of symbols `ArgNames` (used for named-argument uniterms).

2.2 Terms of RIF-Core

The ***Terms*** of RIF-Core are the terms of RIF-BLD with the exclusion of *subclass terms* and of *terms with named arguments*. In RIF-Core there are only *closed ground* lists.

Definition (List Term)

- A ***closed ground list*** has the form List(t_1 ... $_m$), where m≥0 and t_1, ..., t_m are ground terms (no tail and no variables are allowed).

A closed list of the form `List()` (i.e., a list in which `m=0`) is called the ***empty list***.

2.3 Formulas of RIF-Core

The ***Formulas*** of RIF-Core are the formulas of RIF-BLD with the following restrictions.

- Subterms that occur inside atomic formulas can be variables, constants, ground list, or external positional terms. This implies that RIF-Core only allows external function applications.
- Equality terms and class membership terms *cannot* occur in rule conclusions -- they are allowed only in rule premises.
- Terms with named arguments and subclass terms are excluded from RIF-Core.

2.4 Annotations and Documents

RIF-Core allows every term and formula to be optionally annotated in the same way as in RIF-BLD. The frame formulas that are allowed as part of an annotation must be syntactically correct for RIF-Core. In particular, no function symbols are allowed in such a formula.

2.5 Well-formed Formulas

A syntactically correct RIF-Core formula that passes the well-formedness test for RIF-BLD is also a well-formed RIF-Core formula.

Recall that RIF-Core does not allow uninterpreted (i.e., non-external) function symbols. Therefore no symbol in RIF-Core can occur in the context of an (uninterpreted) function symbol.

2.6 EBNF Grammar for the Presentation Syntax of RIF-Core

Until now, we have used mathematical English to specify the syntax of RIF-Core as a restriction on RIF-BLD. Tool developers, however, may prefer EBNF notation, which provides a more succinct view of the syntax. However, EBNF is unable to express all of the well-formedness conditions. For instance, the requirement that each symbol appear in only one context cannot be expressed in EBNF. As a result, the EBNF grammar defines a strict superset of RIF-Core. For that reason this section is *not normative*.

The EBNF for the RIF-Core presentation syntax is given as follows. For convenience of reading we show the entire EBNF divided into three parts (rules, conditions, and annotations); these are derived from the ENBF for RIF-BLD by applying the restrictions described above.

Rule Language:

```
Document       ::= IRIMETA? 'Document' '(' Base? Prefix* Import* Group? ')'
Base           ::= 'Base' '(' ANGLEBRACKIRI ')'
Prefix         ::= 'Prefix' '(' NCName ANGLEBRACKIRI ')'
Import         ::= IRIMETA? 'Import' '(' LOCATOR PROFILE? ')'
Group          ::= IRIMETA? 'Group' '(' (RULE | Group)* ')'
RULE           ::= (IRIMETA? 'Forall' Var+ '(' CLAUSE ')') | CLAUSE
CLAUSE         ::= Implies | ATOMIC
Implies        ::= IRIMETA? (ATOMIC | 'And' '(' ATOMIC* ')') ':-' FORMULA
LOCATOR        ::= ANGLEBRACKIRI
PROFILE        ::= ANGLEBRACKIRI
```

Condition Language:

```
FORMULA        ::= IRIMETA? 'And' '(' FORMULA* ')' |
                   IRIMETA? 'Or' '(' FORMULA* ')' |
                   IRIMETA? 'Exists' Var+ '(' FORMULA ')' |
                   ATOMIC |
                   IRIMETA? Equal |
                   IRIMETA? Member |
                   IRIMETA? 'External' '(' Atom ')'
ATOMIC         ::= IRIMETA? (Atom | Frame)
Atom           ::= UNITERM
UNITERM        ::= Const '(' (TERM* ')'
GROUNDUNITERM  ::= Const '(' GROUNDTERM* ')'
Equal          ::= TERM '=' TERM
Member         ::= TERM '#' TERM
Frame          ::= TERM '[' (TERM '->' TERM)* ']'
TERM           ::= IRIMETA? (Const | Var | List | 'External' '(' Expr ')')
GROUNDTERM     ::= IRIMETA? (Const | List | 'External' '(' GROUNDUNITERM ')')
Expr           ::= UNITERM
List           ::= 'List' '(' GROUNDTERM* ')'
Const          ::= '"' UNICODESTRING '"^^' SYMSPACE | CONSTSHORT
Var            ::= '?' Name
Name           ::= NCName | '"' UNICODESTRING '"'
SYMSPACE       ::= ANGLEBRACKIRI | CURIE
```

Annotations:

```
IRIMETA        ::= '(*' IRICONST? (Frame | 'And' '(' Frame* ')')? '*)'
```

ANGLEBRACKIRI, CURIE, CONSTSHORT, and UNICODESTRING are defined in Section Shortcuts for Constants in RIF's Presentation Syntax of [RIF-DTB].

The following subsections explain and exemplify the Condition Language, Rule Language, and Annotations parts.

2.6.1 EBNF for the RIF-Core Condition Language

The RIF-Core Condition Language represents formulas that can be used in the premises of RIF-Core rules (also called rule bodies). The EBNF grammar for a superset of the RIF-Core condition language is shown in the above conditions part.

This is a specialization of the EBNF for the RIF-BLD condition language specified in the RIF-BLD conditions part reflecting the syntax restrictions on RIF-Core described normatively in sections 2.1 through 2.5 above.

Example 3 from the RIF-BLD document, illustrates some RIF-BLD conditions. All the conditions, except for the terms with named arguments and the equalities with (non-ground) list terms, are also RIF-Core conditions.

2.6.2 EBNF for the RIF-Core Rule Language

The presentation syntax for RIF-Core rules is based on the syntax in Section EBNF for the RIF-Core Condition Language with the productions shown in the above rules part.

Again, this is a specialization of the EBNF for the RIF-BLD rule language specified in the RIF-BLD rules part reflecting the syntax restrictions on RIF-Core described normatively in sections 2.1 through 2.5 above.

Example 4 from the RIF-BLD document also illustrates a set of RIF-Core rules. In contrast, Example 7 from the RIF-BLD document shows a formula that is *not* in RIF-Core because it includes terms with named arguments, which are not allowed in this dialect.

2.6.3 EBNF for RIF-Core Annotations

The presentation syntax for RIF-Core annotations uses the production shown in the above annotations part.

This defines the specialization of the EBNF for the RIF-BLD annotation language specified through the RIF-BLD annotations part where annotation frames use the more restricted TERMs defined in the above conditions part of RIF-Core.

Example 5 from the RIF-BLD document also illustrates a RIF-Core document that contains an annotated group formula.

3 RIF-Core as a Specialization of RIF-PRD

RIF-Core is a syntactic subset of RIF-PRD, and this section defines the presentation syntax of RIF-Core as a restriction on the presentation syntax of RIF-PRD Conditions, Actions, and Rules.

3.1 Alphabet of RIF-Core

The ***alphabet*** of the presentation language of RIF-Core is the alphabet of the RIF-PRD presentation language (Conditions, Actions, and Rules) with the exclusion of the symbols `##`, `such that`, `Not`, `INeg`, `Do`, `Assert`, `Retract`, `Modify`, `Execute`, and `New`.

3.2 Terms of RIF-Core

The ***Terms*** of RIF-Core are the terms of RIF-PRD with the exclusion of *subclass terms*. In Core there are only *closed ground* lists.

3.3 Formulas of RIF-Core

The ***Formulas*** of RIF-Core are the formulas of RIF-PRD with the exclusion of *negation formulas*.

3.4 Annotations and Documents

RIF-Core allows every term and formula to be optionally annotated in the same way as in RIF-PRD. The frame formulas that are allowed as part of an annotation must be syntactically correct for RIF-Core.

3.5 Well-formed Formulas

A syntactically correct RIF-Core formula that passes the well-formedness test for RIF-PRD is also a well-formed RIF-Core formula.

3.6 Rules and Groups

A RIF-Core rule is a well-formed RIF-PRD rule rule with no nested forall, no binding pattern, and where the action block is a single atom, a single frame, or a conjunction of atoms and/or frames. A RIF-Core group is a RIF-PRD group without `strategy` and without `priority`.

4 RIF-Core Semantics

RIF-Core is a syntactic subset of RIF-BLD, and the semantics of RIF-Core is identical to the semantics of RIF-BLD for that subset. RIF-Core is also a syntactic subset of RIF-PRD, and the semantics of RIF-Core is also identical to the semantics of RIF-PRD for that subset.

5 XML Serialization Syntax for RIF-Core

The XML syntax of RIF-Core is a subset of the XML syntax of RIF-BLD. All XML tags of RIF-BLD (except `Subclass`, `sub` and `super`) are supported, but the XML schema of RIF-Core restricts their context with respect to what is allowed by the XML schema of RIF-BLD. The semantics of the XML syntax for RIF-Core is defined through the same RIF-BLD XML-to-presentation syntax mapping.

XML serialization of a complete RIF-Core document appears in the RIF-BLD specification as Example 8.

6 Safeness Criteria

RIF-Core is a syntactic subset of both RIF-BLD and RIF-PRD. The semantics of a RIF-Core formula is the same as the semantics given to it by RIF-BLD.

All RIF-Core documents are also syntactically valid RIF-PRD documents. However, some formulas may be *unsafe* and cannot be executed under the RIF-PRD operational semantics. Thus, in order to allow production rule systems and logic programming systems to interchange rules via RIF-Core, we restrict RIF-Core to *safe* rules so that the logical semantics of RIF-BLD and the operational fixed-point semantics of RIF-PRD coincide.

6.1 Safeness

Intuitively, safeness of rules guarantees that, when performing reasoning in a forward-chaining manner, it is possible to find bindings for all the variables in the rule so that the condition can be evaluated.

To define safeness, we need to define, first, the notion of *binding patterns* for externally defined terms, as well as under what conditions variables are considered *bound*.

Definition (Binding pattern). ***Binding patterns*** are lists of the form $(p_1, ..., p_n)$, such that p_i=b or p_i=u, for $1 \le i \le n$: b stands for a "bound" and u stands for an "unbound" argument. ☐

Each external function or predicate has an associated list of ***valid binding patterns***. We define here the binding patterns valid for the functions and predicates defined in [RIF-DTB].

Every function or predicate `f` defined in [RIF-DTB] has a valid binding pattern for each of its schemas with only the symbol b such that its length is the number of arguments in the schema. In addition,

- the external predicate `pred:iri-string` has the valid binding patterns (b, u) and (u, b) and
- the external predicate `pred:list-contains` has the valid binding pattern (b, u).

The functions and predicates defined in [RIF-DTB] have no other valid binding patterns.

To keep the definitions concise and intuitive, *boundedness* and *safeness* are defined, below, for condition formulas in disjunctive normal form, that can be existentially quantified themselves, but that contain, otherwise, no existential sub-formula. The definitions apply to any valid RIF-Core condition formula, because they can always, in principle, be put in that form, by applying the following syntactic transforms, in sequence:

1. if *f* contains existential sub-formulas, all the quantified variables are renamed, if necessary, and given a name that is unique in *f*, and the scope of the quantifiers is extended to *f*. Assume, for instance, that *f* has an existential sub-formula, *sf* = *Exists* $v_1...v_n$ *(sf')*, $n \ge 1$, such that the names $v_1...v_n$ do not occur in *f* outside of *sf*. After the transform, *f* becomes *Exists* $v_1...v_n$ *(f')*, where *f'* is *f* with *sf* replaced by *sf'* . The transform is applied iteratively to all the existential sub-formulas in *f*;
2. the (possibly existentially quantified) resulting formula is rewritten in disjunctive normal form ([Mendelson97], p. 30).

Definition (Boundedness). An external function term $External(f(t_1,...,t_n))$ is ***bound*** in a condition formula, if and only if `f` has a valid binding pattern $(p_1, ..., p_n)$ and, for all j, $1 \le j \le n$, such that p_j=b, t_j is bound in the formula.

A variable, *v*, is ***bound*** in an atomic formula, *a*, if and only if

- *a* is neither an equality nor an external predicate, and *v* occurs as an argument in *a*;
- or *v* is bound in the conjunction formula *f* = *And(a)*.

A variable, *v*, is ***bound*** in a conjunction formula, *f* = $And(c_1...c_n)$, $n \ge 1$, if and only if, either

- *v* is bound in at least one of the conjuncts;
- or *v* occurs as the *j-th* argument in a conjunct, c_i, that is an externally defined predicate, and the *j-th* position in a binding pattern that is associated with c_i is u, and all the arguments that occur, in c_i, in positions with value *b* in the same binding pattern are bound in *f'* = $And(c_1...c_{i-1}\ c_{i+1}...c_n)$;
- or *v* occurs in a conjunct, c_i, that is an equality formula, and *v* occurs as the term on one side of the equality, and the term on the other side of the equality is bound in *f'* = $And(c_1...c_{i-1}\ c_{i+1}...c_n)$.

A variable, *v*, is ***bound*** in a disjunction formula, if and only if *v* is bound in every disjunct where it occurs;

A variable, *v*, is ***bound*** in an existential formula, $Exists\ v_1,...,v_n\ (f')$, $n \ge 1$, if and only if *v* is bound in *f'*. ☐

Notice that the variables, $v_1,...,v_n$, that are existentially quantified in an existential formula *f* = $Exists\ v_1,...,v_n\ (f')$, are bound in any formula, *F*, that contains *f* as a sub-formula, if and only if they are bound in *f*, since they do not exist outside of *f*.

Definition (Safeness). A variable, *v*, is ***safe*** in a condition formula, *f*, if and only if

- *f* is an atomic formula and *f* is not an equality formula in which both terms are variables, and *v* occurs in *f*;
- or *f* is a conjunction, *f* = $And(c_1...c_n)$, $n \ge 1$, and *v* is safe in at least one conjunct in *f*, or *v* occurs in a conjunct, c_i, that is an equality formula in which both terms are variables, and *v* occurs as the term on one side of the equality, and the variable on the other side of the equality is safe in *f'* = $And(c_1...c_{i-1}\ c_{i+1}...c_n)$;
- or *f* is a disjunction, and *v* is safe in every disjunct;
- or *f* is an existential formula, *f* = $Exists\ v_1,...,v_n\ (f')$, $n \ge 1$, and *v* is safe in *f'* .

A RIF-Core rule, *r* is ***safe*** if and only if

- *r* is a variable free atomic formula,
- or *r* is a universal fact, $Forall\ v_1,...,v_n\ (f)$, $n \ge 1$, and *f* is variable free,

- or *r* is a rule implication, φ :- ψ, and all the variables that occur in *φ* are safe in *ψ*, and all the variables that occur in *ψ* are bound in *ψ*;
- or *r* is a universal rule, Forall $v_1,...,v_n$ (r'), $n \geq 1$, and *r'* is safe.

A group, Group ($s_1...s_n$), $n \geq 0$, is ***safe*** if and only if

- it is empty, that is, $n = 0$;
- or s_1 and ... and s_n are safe.

A document is ***safe*** if and only if

- it contains a safe group, or no group at all,
- and all the documents that it imports are safe. ☐

Example. Consider the following formula:

```
Forall ?x ?y ?z ?u
  (ex:p(?x) :- Or( And( ex:q(?z)
                        External(pred:iri-string(?x ?z))))
                   And( ?x=?y ?y=?u ex:q(?u)))
```

One can verify that this formula is safe, in the following way: the only variable appearing in the conclusion of the rule is ?x; ?x is safe in the first component of the disjunction, because it occurs in the atomic formula pred:iri-string(?x,?z). It is also safe in the second disjunct, because it occurs as the left term in an equality formula where the right term is ?y, which is safe because it occurs as the left term in an equality formula where the right term is ?u, which is safe because it occurs in the atomic formula ex:q(?u). Being safe in both disjuncts, ?x is safe in the disjunction.

Moreover, ?x, ?y, ?z and ?u are all bound in the body of the rule:

- ?z is bound in the first disjunct because it occurs as an argument in the atom ex:q(?z). Therefore, it is bound in the disjunction because it does not occur in the other disjunct;
- ?u is bound in the second disjunct because it occurs as an argument in the atom ex:q(?u). Therefore, it is bound in the disjunction because it does not occur in the other disjunct;
- ?y is bound in the second disjunct because it occurs as the left term in an equality formula where the right term is ?u, which is bound in the conjunction without that equality formula. Therefore, it is bound in the disjunction because it does not occur in the other disjunct;
- ?x is bound in the first disjunct because (u,b) is a valid binding pattern for pred:iri-string, where ?x occurs as the first argument, and ?z, which occurs as the second argument, is bound in the conjunction without the external predicate. ?x is also bound in the second disjunct, because it occurs as the left term in an equality formula where the right term is ?y, which is bound in the conjunction without that equality formula. Therefore, ?x is bound in the disjunction.

6.2 Strong Safeness (Informative)

While safeness guarantees the possibility to do forward chaining with the rules, it does not guarantee that it is possible to construct a finite grounding. For this purpose we define strong safeness.

The conformance clauses for RIF-Core only require conformance over safe rule sets as defined above. However, some rule engines, such as some Datalog engines, are only able to process rule sets which can be finitely grounded. For maximum interoperability with such systems it is recommended that RIF-Core producers restrict themselves to strongly safe rule sets where possible.

Let R be a set of safe rule implications φ :- ψ and let P be the set of pairs (p,n), where p is a predicate symbol and n is a nonnegative integer (an arity). For the purposes of the definitions in this section we view frames a[b -> c] and membership formulas a#b, respectively, as ternary and binary predicate symbols, and so (->,3), (#,2) ∈ P. Note that equality = does not appear in P.

We define the *graph of variable dependencies* of a set of atomic formulas A as a labeled directed graph G_R=(V, E, L), where the labeling function L maps edges to sets of external function and predicate symbols, V is the set of variables appearing in A, and E is the smallest set and L' is the smallest function such that for every variable ?V

- for every atomic formula ?V=t or t=?V in A and every variable ?V'≠?V appearing in t such that $f_1, ..., f_n$, $0 \leq n$, are the function symbols of the terms in t (including t itself) in which ?V' appears, (?V,?V') ∈ E and {$f_1, ..., f_n$} ∈ L'((?V,?V')), and
- for every external atomic formula External(f($t_1,...,t_n$)) in A, every i ∈ {1,...,n} such that t_i=?V, every valid binding pattern (p_1, ..., p_n) of f such that p_i=u, and every variable ?V' appearing in some t_j such that p_i=b and f_1, ..., f_m, $0 \leq m$, are the function symbols of the terms in t_j in which ?V' appears, (?V,?V') ∈ E and {$f_1, ..., f_m$} ∈ L'((?V,?V')).

Finally, L is defined as: for every (e,e') ∈ E, L((e,e')) is the union of the minimal sets in L'((e,e')).

For every rule implication, φ :- ψ, we define the collection, $B_ψ$, of the sets of the atomic formulas in each of the conjunctions that are the components of ψ', where ψ' is ψ rewritten as a condition formula in disjunctive normal form, possibly existentially quantified itself, but otherwise containing no existential sub-formula (see description of the transform in the section Safeness, above).

The *dependency graph* of a set of implications R is a labelled directed graph G_R=(V, E), where edges are triples (v,v',l) such that v, v' ∈ V and l is a set of external function and predicate symbols. V is defined as: for every `(p,n)` ∈ P and every integer `i` such that `1 ≤ i ≤ n`, `(p,n)/i` ∈ V. E is the smallest set such that for every `(p,n)/i` ∈ V and every `φ :- ψ` in R such that there is an atomic subformula $p(t_1,...,t_i,...,t_n)$ of φ, then for every variable ?V appearing in t_i:

- for every non-external and non-equality atomic formula with predicate symbol `p'` and `m` arguments in any A ∈ $B_ψ$ and every `j` ∈ `{1,...,m}` such that a variable `?V'` is the `j`th argument and there is a path from ?V to `?V'` in the graph of variable dependencies of A and F is the union of the labels of the shortest path, $((p,n)/i,(p',m)/j, F \cup \{f_1,...,f_l\})$ ∈ E, where $f_1, ..., f_l$, $0 \le l$, are the function symbols of the terms in t_i in which ?V appears.

Definition (Strong safeness). A set of rule implications *R* is ***strongly safe*** if its dependency graph does not contain cycles involving edges labelled with sets involving a function defined in [RIF-DTB] that is not a casting function. A RIF document *D* is strongly safe if the set *R* of rule implications that are subformulas of *D* is strongly safe.

7 Conformance Clauses

RIF-Core conformance is described in terms of semantics-preserving transformations.

Let T be a set of datatypes and symbol spaces that includes the datatypes specified in [RIF-DTB] and the symbol spaces `rif:iri` and `rif:local`. Suppose also that E is a set of external predicates and functions that includes the built-ins listed in [RIF-DTB]. We say that a formula φ is a $Core_{T,E}$ formula iff

- φ is a well-formed Core formula,
- all the datatypes and symbol spaces used in φ are in T, and
- all the externally defined functions and predicates used in φ are in E.

A RIF processor is a ***conformant*** $Core_{T,E}$ ***consumer*** iff it implements a *semantics-preserving mapping* from the set of all *safe* $Core_{T,E}$ formulas to the language *L* of the processor.

A RIF processor is a ***conformant*** $Core_{T,E}$ ***producer*** iff it implements a *semantics-preserving mapping* from the language *L* of the processor to a set of *safe* $Core_{T,E}$ formulas.

An ***admissible document*** is an XML document that conforms to all the syntactic constraints of RIF-Core, including ones that cannot be checked by an XML Schema validator. Note that the concrete presentation syntax given in Section 2.6 is purely informative (to help implementers see the set of language structures supported by RIF-Core); the only normative concrete syntax for RIF-Core is the XML syntax.

In addition:

- Conformant Core producers and consumers are required to support only the entailments of the form $φ \models_{CORE} ψ$, where ψ is a *closed RIF-Core condition formulas*, that is a RIF-Core condition formula which also meets the criteria for closed condition formula defined in RIF-BLD.
- A ***conformant Core consumer*** must reject any document containing features it does not support.
- A ***conformant Core producer*** is a conformant $Core_{T,E}$ producer which produces documents that include only the symbol spaces, datatypes, and externals that are required by Core.

8 Acknowledgements

This document is the product of the Rules Interchange Format (RIF) Working Group (see below) whose members deserve recognition for their time and commitment. The editors extend special thanks to Jos de Bruijn for his safeness definition and to: Jos de Bruijn, Leora Morgenstern, Christian de Sainte-Marie, Stella Mitchell and Changhai Ke 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 Sandro Hawke, a constant source of ideas, help, and feedback.

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

W3C Proposed Edited Recommendation

9 References

9.1 Normative References

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

[RFC-3066]
RFC 3066 **-** *Tags for the Identification of Languages*, H. Alvestrand, IETF, January 2001. This document is http://www.ietf.org/rfc/rfc3066 .

[RFC-3987]
RFC 3987 **-** *Internationalized Resource Identifiers (IRIs)*, M. Duerst and M. Suignard, IETF, January 2005. This document is http://www.ietf.org/rfc/rfc3987 .

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

[RIF-DTB]
RIF Datatypes and Built-Ins 1.0 Axel Polleres, Harold Boley, Michael Kifer, eds. W3C Proposed Edited Recommendation, 11 December 2012, http://www.w3.org/TR/2012/PER-rif-dtb-20121211/. Latest version available at http://www.w3.org/TR/rif-dtb/.

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

[RIF-RDF+OWL]
RIF RDF and OWL Compatibility Jos de Bruijn, Chris Welty, eds. W3C Proposed Edited Recommendation, 11 December 2012, http://www.w3.org/TR/2012/PER-rif-rdf-owl-20121211/. Latest version available at http://www.w3.org/TR/rif-rdf-owl/.

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

[XML1.0]
Extensible Markup Language (XML) 1.0 (Fourth Edition), W3C Recommendation, World Wide Web Consortium, 16 August 2006, edited in place 29 September 2006. This version is http://www.w3.org/TR/2006/REC-xml-20060816/.

[XML-Base]
XML Base, W3C Recommendation, World Wide Web Consortium, 27 June 2001. This version is http://www.w3.org/TR/2001/REC-xmlbase-20010627/. The latest version is available at http://www.w3.org/TR/xmlbase/.

[XML-SCHEMA2]
XML Schema Part 2: Datatypes, W3C Recommendation, World Wide Web Consortium, 2 May 2001. This version is http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/. The latest version is available at http://www.w3.org/TR/xmlschema-2/.

9.2 Informational References

[ANF01]
Normal Form Conventions for XML Representations of Structured Data, Henry S. Thompson. October 2001. Available at http://www.ltg.ed.ac.uk/~ht/normalForms.html.

[CL73]
Symbolic Logic and Mechanical Theorem Proving, C.L. Chang and R.C.T. Lee. Academic Press, 1973.

[CURIE]
CURIE Syntax 1.0, S. McCarron, M. Birbeck, Editors, W3C Working Group Note, 16 December 2010, http://www.w3.org/TR/2010/NOTE-curie-20101216 . Latest version available at http://www.w3.org/TR/curie.

[Enderton01]
A Mathematical Introduction to Logic, Second Edition, H. B. Enderton. Academic Press, 2001.

[KLW95]
Logical foundations of object-oriented and frame-based languages, M. Kifer, G. Lausen, J. Wu. Journal of ACM, July 1995, pp. 741--843.

[Mendelson97]
Introduction to Mathematical Logic, Fourth Edition, E. Mendelson. Chapman & Hall, 1997.

[OWL-Reference]
OWL Web Ontology Language Reference, M. Dean, G. Schreiber, Editors, W3C Recommendation, 10 February 2004. Latest version available at http://www.w3.org/TR/owl-ref/.

[RDFSYN04]
RDF/XML Syntax Specification (Revised), Dave Beckett, Editor, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/. Latest version available at http://www.w3.org/TR/rdf-syntax-grammar/.

[RIF-UCR]
RIF Use Cases and Requirements Adrian Paschke, David Hirtle, Allen Ginsberg, Paula-Lavinia Patranjan, Frank McCabe, eds. W3C Working Draft, 18 December 2008, http://www.w3.org/TR/2008/WD-rif-ucr-20081218/. Latest version available at http://www.w3.org/TR/rif-ucr/.

[TRT03]
Object-Oriented RuleML: User-Level Roles, URI-Grounded Clauses, and Order-Sorted Terms, H. Boley. Springer LNCS 2876, Oct. 2003, pp. 1-16. Available at http://nparc.cisti-icist.nrc-cnrc.gc.ca/npsi/ctrl?action=rtdoc&an=5764336&article=19&fd=pdf.

[vEK76]
The semantics of predicate logic as a programming language, M. van Emden and R. Kowalski. Journal of the ACM 23 (1976), pp. 733-742.

10 Appendix: XML Schema for RIF-Core

The **namespace** of RIF is "http://www.w3.org/2007/rif#".

XML schemas for the RIF-Core sublanguages are defined below and are also available at http://www.w3.org/2010/rif-schema/core/ with additional examples.

10.1 Condition Language

```
<?xml version="1.0" encoding="UTF-8"?>

 <xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xml="http://www.w3.org/XML/1998/namespace"
  xmlns="http://www.w3.org/2007/rif#"
  targetNamespace="http://www.w3.org/2007/rif#"
  elementFormDefault="qualified"
  version="Id: CoreCond.xsd, v. 1.4, 2010-05-08, hboley/apaschke">

 <xs:import namespace='http://www.w3.org/XML/1998/namespace'
            schemaLocation='http://www.w3.org/2001/xml.xsd'/>

  <xs:annotation>
    <xs:documentation>
    This is the XML schema for the Condition Language as defined by
    the RIF-Core dialect.

    The schema is based on the following EBNF for the RIF-Core Condition Language
    (prepared for generalization to the RIF-BLD and RIF-PRD Condition Languages):

  FORMULA        ::= IRIMETA? 'And' '(' FORMULA* ')' |
                     IRIMETA? 'Or' '(' FORMULA* ')' |
                     IRIMETA? 'Exists' Var+ '(' FORMULA ')' |
                     ATOMIC |
                     IRIMETA? Equal |
                     IRIMETA? Member |
                     IRIMETA? 'External' '(' Atom ')'
  ATOMIC         ::= IRIMETA? (Atom | Frame)
  Atom           ::= UNITERM
  UNITERM        ::= Const '(' (TERM* ')'
  GROUNDUNITERM  ::= Const '(' GROUNDTERM* ')'
```

```
Equal          ::= TERM '=' TERM
Member         ::= TERM '#' TERM
Frame          ::= TERM '[' (TERM '->' TERM)* ']'
TERM           ::= IRIMETA? (Const | Var | List | 'External' '(' Expr ')')
GROUNDTERM     ::= IRIMETA? (Const | List | 'External' '(' GROUNDUNITERM ')')
Expr           ::= UNITERM
List           ::= 'List' '(' GROUNDTERM* ')'
Const          ::= '"' UNICODESTRING '"^^' SYMSPACE | CONSTSHORT
Var            ::= '?' Name
Name           ::= NCName | '"' UNICODESTRING '"'
SYMSPACE       ::= ANGLEBRACKIRI | CURIE

IRIMETA        ::= '(*' IRICONST? (Frame | 'And' '(' Frame* ')')? '*)'

  </xs:documentation>
</xs:annotation>

<xs:group name="FORMULA">
  <!--
FORMULA        ::= IRIMETA? 'And' '(' FORMULA* ')' |
                   IRIMETA? 'Or' '(' FORMULA* ')' |
                   IRIMETA? 'Exists' Var+ '(' FORMULA ')' |
                   ATOMIC |
                   IRIMETA? Equal |
                   IRIMETA? Member |
                   IRIMETA? 'External' '(' Atom ')'
  -->
  <xs:choice>
    <xs:element ref="And"/>
    <xs:element ref="Or"/>
    <xs:element ref="Exists"/>
    <xs:group ref="ATOMIC"/>
    <xs:element ref="Equal"/>
    <xs:element ref="Member"/>
    <xs:element name="External" type="External-FORMULA.type"/>
  </xs:choice>
</xs:group>

<xs:complexType name="External-FORMULA.type">
  <!-- sensitive to FORMULA (Atom) context-->
  <xs:sequence>
    <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
    <xs:element name="content" type="content-FORMULA.type"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="content-FORMULA.type">
  <!-- sensitive to FORMULA (Atom) context-->
  <xs:sequence>
    <xs:element ref="Atom"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="And">
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="formula" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Or">
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="formula" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Exists">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="declare" minOccurs="1" maxOccurs="unbounded"/>
        <xs:element ref="formula"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="formula">
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="FORMULA"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="declare">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Var"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:group name="ATOMIC">
    <!--
  ATOMIC         ::= IRIMETA? (Atom | Frame)
    -->
    <xs:choice>
      <xs:element ref="Atom"/>
      <xs:element ref="Frame"/>
    </xs:choice>
  </xs:group>

  <xs:element name="Atom">
    <!--
  Atom           ::= UNITERM
    -->
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="UNITERM"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:group name="UNITERM">
    <!--
  UNITERM        ::= Const '(' (TERM* ')'
    -->
    <xs:sequence>
      <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="op"/>
      <xs:element name="args" type="args-UNITERM.type" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:group>

  <xs:group name="GROUNDUNITERM">
    <!-- sensitive to ground terms
  GROUNDUNITERM        ::= Const '(' (GROUNDTERM* ')'
    -->
    <xs:sequence>
      <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="op"/>
      <xs:element name="args" type="args-GROUNDUNITERM.type" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:group>

  <xs:element name="op">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Const"/>
```

```
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:complexType name="args-UNITERM.type">
  <!-- sensitive to UNITERM (TERM) context-->
  <xs:sequence>
    <xs:group ref="TERM" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="ordered" type="xs:string" fixed="yes"/>
</xs:complexType>

<xs:complexType name="args-GROUNDUNITERM.type">
  <!-- sensitive to GROUNDUNITERM (TERM) context-->
  <xs:sequence>
    <xs:group ref="GROUNDTERM" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="ordered" type="xs:string" fixed="yes"/>
</xs:complexType>

<xs:element name="Equal">
  <!--
Equal          ::= TERM '=' TERM
  -->
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="left"/>
      <xs:element ref="right"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="left">
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="TERM"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="right">
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="TERM"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Member">
  <!--
Member         ::= TERM '#' TERM
  -->
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="instance"/>
      <xs:element ref="class"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="instance">
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="TERM"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="class">
  <xs:complexType>
```

```
      <xs:sequence>
        <xs:group ref="TERM"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Frame">
    <!--
  Frame          ::= TERM '[' (TERM '->' TERM)* ']'
    -->
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="object"/>
        <xs:element name="slot" type="slot-Frame.type" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="object">
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="TERM"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="slot-Frame.type">
    <!-- sensitive to Frame (TERM) context-->
    <xs:sequence>
      <xs:group ref="TERM"/>
      <xs:group ref="TERM"/>
    </xs:sequence>
    <xs:attribute name="ordered" type="xs:string" fixed="yes"/>
  </xs:complexType>

  <xs:group name="TERM">
    <!--
  TERM           ::= IRIMETA? (Const | Var | List | 'External' '(' Expr ')')
    -->
      <xs:choice>
         <xs:element ref="Const"/>
         <xs:element ref="Var"/>
         <xs:element ref="List"/>
         <xs:element name="External" type="External-TERM.type"/>
      </xs:choice>
  </xs:group>

  <xs:group name="GROUNDTERM">
    <!--
  GROUNDTERM     ::= IRIMETA? (Const | List | 'External' '(' GROUNDUNITERM ')')
    -->
      <xs:choice>
         <xs:element ref="Const"/>
         <xs:element ref="List"/>
         <xs:element name="External" type="External-GROUNDUNITERM.type"/>
      </xs:choice>
  </xs:group>

 <xs:element name="List">
    <!--
      List           ::= 'List' '(' GROUNDTERM* ')'
                 rewritten as
      List           ::= 'List' '(' LISTELEMENTS? ')'
    -->
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
        <xs:group ref="LISTELEMENTS" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
<xs:group name="LISTELEMENTS">
  <!--
LISTELEMENTS   ::= GROUNDTERM+
  -->
  <xs:sequence>
    <xs:element ref="items"/>
  </xs:sequence>
</xs:group>

<xs:element name="items">
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="GROUNDTERM" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ordered" type="xs:string" fixed="yes"/>
  </xs:complexType>
</xs:element>

<xs:complexType name="External-TERM.type">
  <!-- sensitive to TERM (Expr) context-->
  <xs:sequence>
    <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
    <xs:element name="content" type="content-TERM.type"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="External-GROUNDUNITERM.type">
  <!-- sensitive to GROUNDTERM (Expr) context-->
  <xs:sequence>
    <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
    <xs:element name="content" type="content-GROUNDUNITERM.type"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="content-TERM.type">
  <!-- sensitive to TERM (Expr) context-->
  <xs:sequence>
    <xs:element ref="Expr"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="content-GROUNDUNITERM.type">
  <!-- sensitive to GROUNDTERM (Expr) context-->
  <xs:sequence>
    <xs:element name="Expr" type="content-GROUNDEXPR.type"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="content-GROUNDEXPR.type">
  <!-- sensitive to GROUNDEXPR context-->
  <xs:sequence>
    <xs:element name="GROUNDUNITERM"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="Expr">
  <!--
Expr           ::= UNITERM
  -->
  <xs:complexType>
    <xs:sequence>
      <xs:group ref="UNITERM"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Const">
  <!--
Const          ::= '"' UNICODESTRING '"^^' SYMSPACE | CONSTSHORT
  -->
  <xs:complexType mixed="true">
```

```
    <xs:sequence>
      <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="type" type="xs:anyURI" use="required"/>
    <xs:attribute ref="xml:lang"/>
  </xs:complexType>
</xs:element>

<xs:element name="Name" type="xs:string">
  <!--
Name           ::= UNICODESTRING
  -->
</xs:element>

<xs:element name="Var">
  <!--
Var            ::= '?' Name
  -->
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:group name="IRIMETA">
  <!--
IRIMETA   ::= '(*' IRICONST? (Frame | 'And' '(' Frame* ')')? '*)'
  -->
  <xs:sequence>
    <xs:element ref="id" minOccurs="0" maxOccurs="1"/>
    <xs:element ref="meta" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:group>

<xs:element name="id">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Const" type="IRICONST.type"/>   <!-- type="&rif;iri" -->
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="meta">
  <xs:complexType>
   <xs:choice>
     <xs:element ref="Frame"/>
     <xs:element name="And" type="And-meta.type"/>
   </xs:choice>
  </xs:complexType>
</xs:element>

<xs:complexType name="And-meta.type">
<!-- sensitive to meta (Frame) context-->
  <xs:sequence>
    <xs:element name="formula" type="formula-meta.type" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="formula-meta.type">
  <!-- sensitive to meta (Frame) context-->
  <xs:sequence>
    <xs:element ref="Frame"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="IRICONST.type" mixed="true">
  <!-- sensitive to location/id context-->
  <xs:sequence/>
  <xs:attribute name="type" type="xs:anyURI" use="required" fixed="http://www.w3.org/2007/rif#iri"/>
</xs:complexType>
```

W3C Proposed Edited Recommendation

```
</xs:schema>
```

10.2 Rule Language

```
<?xml version="1.0" encoding="UTF-8"?>

 <xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xml="http://www.w3.org/XML/1998/namespace"
  xmlns="http://www.w3.org/2007/rif#"
  targetNamespace="http://www.w3.org/2007/rif#"
  elementFormDefault="qualified"
  version="Id: CoreRule.xsd, v. 1.5, 2010-05-08, hboley/apaschke">

  <xs:annotation>
    <xs:documentation>
    This is the XML schema for the Rule Language as defined by
    the RIF-Core dialect.

    The schema is based on the following EBNF for the RIF-Core Rule Language
    (prepared for generalization to the RIF-BLD and RIF-PRD Rule Languages):

  Document       ::= IRIMETA? 'Document' '(' Base? Prefix* Import* Group? ')'
  Base           ::= 'Base' '(' ANGLEBRACKIRI ')'
  Prefix         ::= 'Prefix' '(' NCName ANGLEBRACKIRI ')'
  Import         ::= IRIMETA? 'Import' '(' LOCATOR PROFILE? ')'
  Group          ::= IRIMETA? 'Group' '(' (RULE | Group)* ')'
  RULE           ::= (IRIMETA? 'Forall' Var+ '(' CLAUSE ')') | CLAUSE
  CLAUSE         ::= Implies | ATOMIC
  Implies        ::= IRIMETA? (ATOMIC | 'And' '(' ATOMIC* ')') ':-' FORMULA
  LOCATOR        ::= ANGLEBRACKIRI
  PROFILE        ::= ANGLEBRACKIRI

    Note that this is an extension of the syntax for the RIF-Core Condition Language (CoreCond.xsd).
    </xs:documentation>
  </xs:annotation>

  <!-- The Rule Language includes the Condition Language from the same directory -->
  <xs:include schemaLocation="CoreCond.xsd"/>

  <xs:element name="Document">
    <!--
  Document  ::= IRIMETA? 'Document' '(' Base? Prefix* Import* Group? ')'
    -->
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="directive" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="payload" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="directive">
    <!--
  Base and Prefix represented directly in XML
    -->
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Import"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="payload">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Group"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

W3C Proposed Edited Recommendation

```
  <xs:element name="Import">
    <!--
  Import    ::= IRIMETA? 'Import' '(' LOCATOR PROFILE? ')'
  LOCATOR   ::= ANGLEBRACKIRI
  PROFILE   ::= ANGLEBRACKIRI
    -->
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="location"/>
        <xs:element ref="profile" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="location" type="xs:anyURI"/>

  <xs:element name="profile" type="xs:anyURI"/>

  <xs:element name="Group">
    <!--
  Group     ::= IRIMETA? 'Group' '(' (RULE | Group)* ')'
    -->
    <xs:complexType>
     <xs:sequence>
      <xs:group ref="Group.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

 <xs:group name="Group.content">
     <xs:sequence>
        <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="sentence" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
 </xs:group>

 <xs:element name="sentence">
   <xs:complexType>
     <xs:choice>
       <xs:group ref="RULE"/>
       <xs:element ref="Group"/>
     </xs:choice>
   </xs:complexType>
 </xs:element>

  <xs:group name="RULE">
    <!--
  RULE      ::= (IRIMETA? 'Forall' Var+ '(' CLAUSE ')') | CLAUSE
    -->
    <xs:choice>
      <xs:element ref="Forall"/>
      <xs:group ref="CLAUSE"/>
    </xs:choice>
  </xs:group>

  <xs:element name="Forall">
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="Forall.content"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:group name="Forall.content">
      <xs:sequence>
        <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="declare" minOccurs="1" maxOccurs="unbounded"/>
        <!-- different from formula in And, Or and Exists -->
        <xs:element name="formula">
          <xs:complexType>
```

```
          <xs:group ref="CLAUSE"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:group>

  <xs:group name="CLAUSE">
    <!--
  CLAUSE   ::= Implies | ATOMIC
    -->
    <xs:choice>
      <xs:element ref="Implies"/>
      <xs:group ref="ATOMIC"/>
    </xs:choice>
  </xs:group>

  <xs:element name="Implies">
    <!--
  Implies   ::= IRIMETA? (ATOMIC | 'And' '(' ATOMIC* ')') ':-' FORMULA
    -->
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="IRIMETA" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="if"/>
        <xs:element ref="then"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="if">
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="FORMULA"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="then">
    <xs:complexType>
     <xs:sequence>
       <xs:group ref="then.content"/>
     </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:group name="then.content">
     <xs:choice>
       <xs:group ref="ATOMIC"/>
       <xs:element name="And" type="And-then.type"/>
     </xs:choice>
  </xs:group>


  <xs:complexType name="And-then.type">
    <!-- sensitive to then (ATOMIC) context-->
    <xs:sequence>
      <xs:element name="formula" type="formula-then.type" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="formula-then.type">
    <!-- sensitive to then (ATOMIC) context-->
    <xs:sequence>
      <xs:group ref="ATOMIC"/>
    </xs:sequence>
  </xs:complexType>

 </xs:schema>
```

11 Appendix: RIF Media Type Registration

The anticipated RIF media type is "application/rif+xml". The registration for this media type (pending IETF discussion and approval by the IESG) follows.

```
Type name: application

Subtype name: rif+xml

Required parameters: none

Optional parameters: charset, as per RFC 3023 (XML Media Types)

Encoding considerations: same as RFC 3023 (XML Media Types)

Security considerations:

    Systems which consume RIF documents are potentially vulnerable
    to attack by malicious producers of RIF documents.  The
    vulnerabilities and forms of attack are similar to those of
    other Web-based formats with programming or scripting
    capabilities, such as HTML with embedded Javascript.

    Excessive Resource Use / Denial of Service Attacks

       Complete processing of a RIF document, even a conformant
       RIF Core document, may require arbitrarily great CPU and
       memory resources.  Through the use of "import", processing
       may also require arbitrary URI dereferencing, which may
       consume all available network resources on the consuming
       system or other systems.  RIF consuming systems SHOULD
       implement reasonable defenses against these attacks.

    Exploiting Implementation Flaws

       RIF is a relatively complex format, and rule engines can be
       extremely sophisticated, so it is likely that some RIF
       consuming systems will have bugs which allow specially
       constructed RIF documents to perform inappropriate
       operations. We urge RIF implementors to make systems which
       carefully anticipate and handle all possible inputs,
       including those which present syntactic or semantic errors.

    External (Application) Functions

       Because RIF may be extended with local, application defined
       datatypes and functions, new vulnerabilities may be
       introduced.  Before being installed on systems which consume
       untrusted RIF documents, these external functions should be
       closely reviewed for their own vulnerabilities and for the
       vulnerabilities that may occur when they are used in
       unexpected combinations, like "cross-site scripting"
       attacks.

    In addition, as this media type uses the "+xml" convention, it
    shares the same security considerations as other XML formats;
    see RFC 3023 (XML Media Types).


Interoperability considerations:

    This media type is intended to be shared with other RIF
    dialects, to be specified in the future.  Interoperation
    between the dialects is governed by the RIF specifications.

Published specifications:

    RIF Core Dialect
    W3C Working Draft (Recommendation Track)
    http://www.w3.org/TR/rif-core/
```

```
    RIF Datatypes and Builtins
    W3C Working Draft (Recommendation Track)
    http://www.w3.org/TR/rif-dtb/

    RIF Basic Logic Dialect
    W3C Working Draft (Recommendation Track)
    http://www.w3.org/TR/rif-bld/

    RIF Production Rule Dialect
    W3C Working Draft (Recommendation Track)
    http://www.w3.org/TR/rif-prd/

    RIF Framework for Logic Dialects
    W3C Working Draft (Recommendation Track)
    http://www.w3.org/TR/rif-fld/

    This media type is intended for use by all RIF dialects,
    including those to be specified in the future.  Identification
    of the RIF dialect in use by a document is done by examining
    the use of specific XML elements within the document.

Applications that use this media type:

    See: http://www.w3.org/2005/rules/wiki/Implementations

Additional information:

  Magic number(s):

        As with XML in general (See RFC 3023 (XML Media Types)),
        there is no magic number for this format.

        However, the XML namespace "http://www.w3.org/2007/rif#" will
        normally be present in the document.  It may theoretically
        be missing if the document uses XML entities in an
        obfuscatory manner, and may also be present in documents with
        ther media types, so use of the namespace is not conclusive.

        The hex form of that namespace will depend on the charset.
        For utf-8, the hex is: 68 74 74 70 3a 2f 2f 77 77 77 2e 77
        33 2e 6f 72.

  File extension(s):

        .rif (or .xml)

  Macintosh file type code(s):

        "TEXT" (like other XML)

Person & email address to contact for further information:

    Sandro Hawke, sandro@w3.org.  Please send technical comments
    and questions about RIF to public-rif-comments@w3.org, a
    mailing list with a public archive at
    http://lists.w3.org/Archives/Public/public-rif-comments/

Intended usage:

    COMMON

Restrictions on usage:

    None

Author:

    The editor and contact for this media type registration is
    Sandro Hawke, sandro@w3.org.

Change controller:
```

W3C Proposed Edited Recommendation

```
RIF is a product of the Rule Interchange Format (RIF) Working
Group of the World Wide Web Consortium (W3C).  See
http://www.w3.org/2005/rules/wg for information on the group.
The W3C (currently acting through this working group) has
change control over the RIF specification.
```

12 Appendix: Change Log (Informative)

This appendix summarizes the main changes to this document.

Changes since the draft of July 3, 2009.

- IRI was replaced with ANGLEBRACKIRI in CoreRule.xsd, v. 1.3.
- The complexType ANYURICONST was introduced in CoreRule.xsd, v. 1.3.
- A number of typos were found and fixed.

Changes since the Candidate Recommendation of October 1, 2009.

- Import's anyURIs were moved directly into location and profile.
- Simplified notion of conformant Core consumer.
- Fixed List by permitting IRIMETA and aligning syntax to Expr and Atom.
- Accommodated DTB-triggered UNICODESTRING/NCName changes of BLD in EBNF.

Changes since the Recommendation of 22 June, 2010.

- Definition (Strong safeness): introduced document metavariable D; omitted Editor's Note.