The Rule Interchange Format and Its Dialects

Michael Kifer

Stony Brook University

Outline

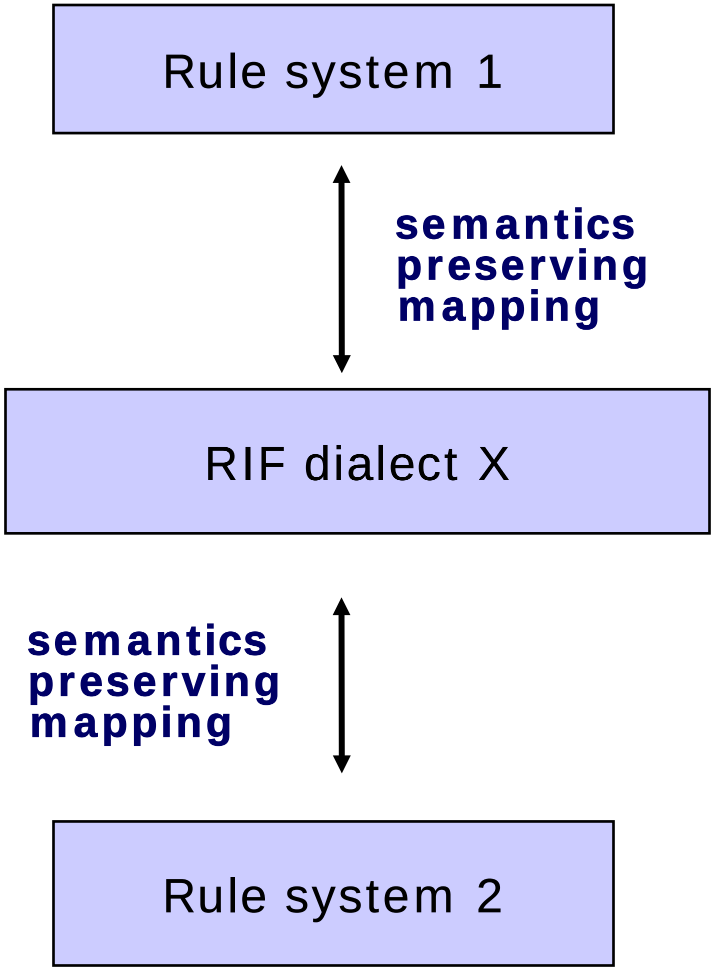
- What is Rule Interchange Format (RIF)?
- RIF Framework
- Current Logic Dialects
- Status/Conclusion

What is RIF?

- A collection of *dialects* (rigorously defined rule languages)
- Intended to facilitate rule sharing and exchange
- Dialect consistency

  Sharing of RIF machinery:

  - XML syntax
  - Presentation syntax
  - Semantics



Rule system 1

semantics preserving mapping

RIF dialect X

semantics preserving mapping

Rule system 2

Why Rule *Exchange*?

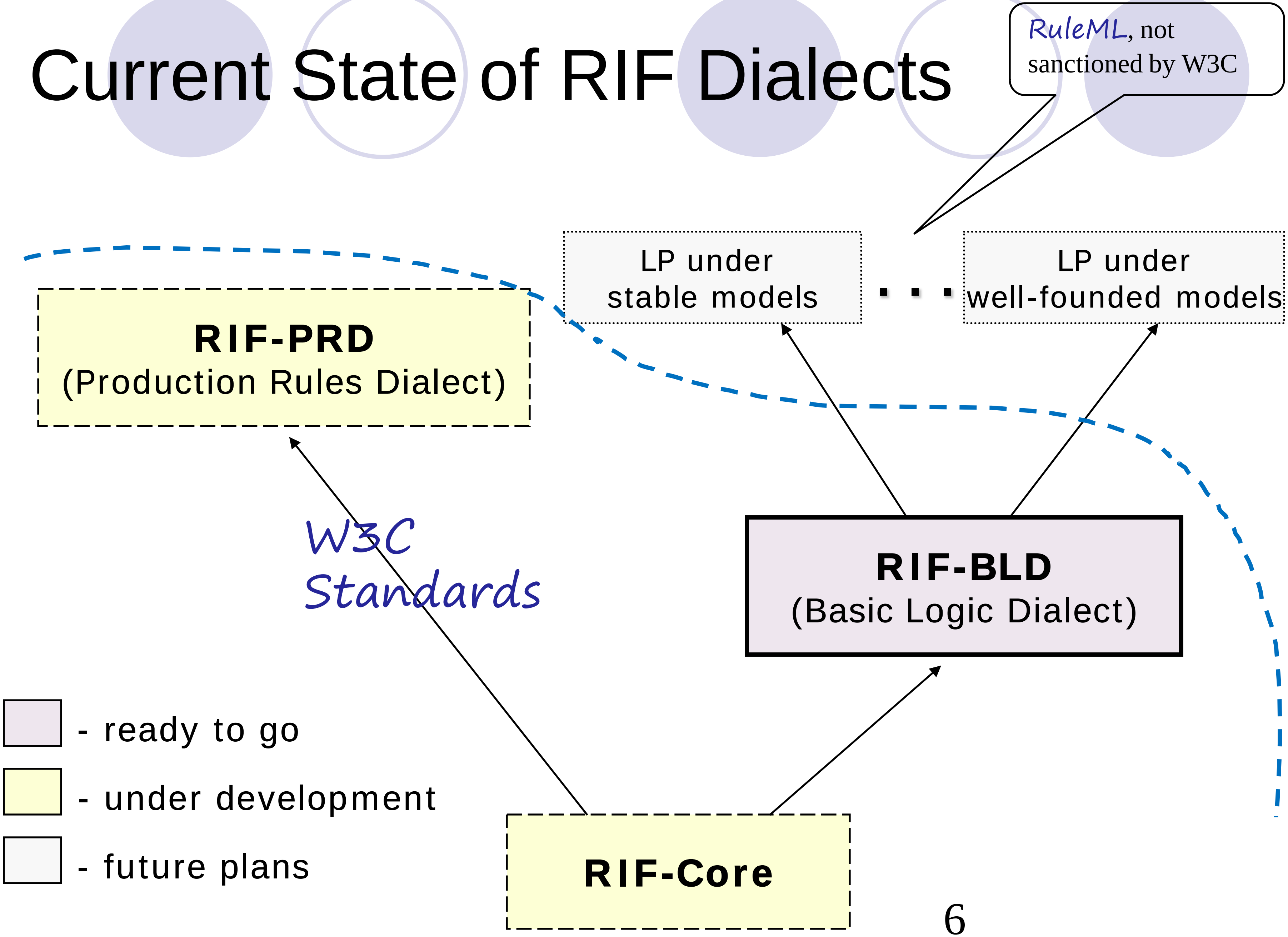
(and not The One True Rule Language)

- Many different paradigms for rule languages
  - Pure first-order
  - Logic programming/deductive databases
  - Production rules
  - Reactive rules
- Many different features and syntaxes
- Different commercial interests
- Different preferences, aesthetics

Why RIF *Dialects*?
(and not just *one* dialect)

- Again: many paradigms for rule languages
  - First-order rules
  - Logic programming/deductive databases
  - Reactive rules
  - Production rules
- Many different semantics
  - Classical first-order
  - Stable-model semantics for negation
  - Well-founded semantics for negation
  - ... ...
- A carefully chosen set of interrelated dialects can serve the purpose of sharing and exchanging rules over the Web

Current State of RIF Dialects



RuleML, not sanctioned by W3C

LP under stable models

. . .

LP under well-founded models

RIF-PRD (Production Rules Dialect)

W3C Standards

RIF-BLD (Basic Logic Dialect)

RIF-Core

🟪 - ready to go

🟨 - under development

⬜ - future plans

Why Is RIF Important?

- A strong chance to bring rule languages into mainstream
- Could make Web programming truly cool!
- For academic types:
  - A treasure-trove of interesting problems
- For industrial types:
  - A vast field for entrepreneurship
  - A great potential for new products

Technical Part

- W3C didn’t allow the development of useful logic dialects beyond the basics
- But it did allow to develop RIF-FLD, a framework for future such dialects
- RIF-FLD: The RIF Framework
  - What?
  - Why?
  - How?

What Is The RIF Framework?

- Formal guidelines for constructing RIF dialects in a consistent manner
- Includes:
  - Syntactic framework
  - Semantic framework
  - XML framework

Why Create a RIF Framework?

- Too hard to define a dialect from scratch
  - RIF-BLD is just a tad more complex than Horn rules, but requires more than 30 pages of dense text
- Instead: define dialects by *specializing* from RIF-FLD
  - RIF-BLD can be specified in < 3 pages in this way
- RIF-FLD is a “*super-dialect*” that ensures that all dialects use the same set of concepts and constructs

RIF-FLD (cont’d)

- RIF-FLD is not a fully specified dialect ...

  ... but a *framework* for dialects
- Very general syntax, but several parameters are not specified – left to the actual dialects
- Very general semantics, but several aspects are under-specified – left to the actual dialects
- General XML syntax – the actual dialects can specialize

RIF-FLD's Syntactic Framework

- Presentation syntax
  - Human-oriented
  - Designed for
    - Precise specification of syntax and semantics
    - Examples
    - Perhaps even for rule authoring
  - Maps to XML syntax
- XML syntax
  - For exchange through the wire
  - Machine consumption

RIF-FLD Syntactic Framework (cont'd)

- General (and extensible) so other dialects' syntaxes can be expressed by *specializing* the syntax of FLD
- Interpretable in model-theoretic terms
  - because FLD is intended as a framework for *logic-based* dialects with model-theoretic semantics

Examples of Syntactic Forms Supported in RIF-FLD

- Function/predicate application

  Point(?X abc)

  ?X(Amount(20) ?Y(cde fgh))

- Functions/predicates with named arguments

  ?F(name->Bob age->15)

HiLog-y variables are allowed

Examples of Syntactic Forms (cont'd)

- Frame (object-oriented F-logic notation)

  Obj[Prop$_1$->Val$_1$ ... Prop$_n$->Val$_n$]

- Member/Subclass (: and :: in F-logic)

  Member#Class

  SubCl##SupCl

- Higher-order functions

  ?F(a)(b c)

  f(?X(a b)(c)(d ?E) ?X ?Y(ab)(?Z))

Examples of Syntactic Forms (cont'd)

- Equality
  - Including in rule conclusions
- Negation
  - Symmetric (classical, explicit): Neg
  - Default (various– stable/ASP, well-founded): Naf
- Connectives, quantifiers

```
Or (And(?X And p(?X ?Y)) ?Z(p))
Forall ?X ?Y (Exists ?Z
                    (f(?X(a b)(c)(d ?E) ?X ?Y(ab)(?Z))))
```

  - New connectives/quantifiers can be added

Syntactic Forms (Cont'd)

- Some dialects may allow/disallow some syntactic forms
  - For instance, no frames
- Some may restrict certain symbols to only certain contexts
  - For instance, no variables over functions, no higher-order functions
- A syntactic form can occur
  - as a *term* (i.e., in an object position)
  - or as a *formula*, or both (*reification*)
- How can all this be specified without repeating the definitions?

Signatures

- Every symbol is given a *signature*
  - Specifies the contexts where the symbol is allowed to occur
  - Symbols can be *polymorphic* (can take different kinds of arguments)
  - And *polyadic* (can occur with different numbers of arguments)
- Each dialect defines:
  - Which signatures are to be given to which symbols
  - How this assignment is specified

Is the syntactic framework too fancy?

- Cannot be rich enough!
- Cf. languages like
  - Flora-2
  - Rulelog

RIF-FLD Semantic Framework

- Defines *semantic structures* (a.k.a. *interpretations*)
  - Structures that determine if a formula is true
  - Very general. Gives semantics to:
    - Frame syntax, predicate syntax, predicates with named arguments
    - Higher-order features
    - Reification
  - Supports multivalued logics
    - For uncertainty, inconsistency

Semantic Framework (cont'd)

- Logical entailment
  - Central to any logic
  - Determines which formulas entail which other formulas
- Unlikely to find one notion of entailment for all logic dialects because

Semantic Framework (cont'd)

- Thus, RIF-FLD under-specifies the semantics
  - Defines entailment parametrically, leaves parameters to the actual dialects
  - Parameters: *intended models*, sets of truth values, etc.
  - Entailment between sets of formulas:
    - *P* |= *Q* iff

      every *intended* model ***I*** of *P* is also a model of *Q*

Other Issues: Link to the Web World

- Symbol spaces
  - Partitions all constants into subsets; each subset have different semantics
    - rif:iri – these constants denote objects that are universally known on the Web (as in RDF)
    - rif:local – constants that denote objects local to specific documents
    - Data types: symbol spaces with fixed interpretation (includes most of the XML data types + more)
- Document formulas, meta-annotations, ...

Logic Dialects

- RIF-BLD, the basic logic dialect (a W3C recommendation)
  - Horn rules, no negation
  - Frames, predicates/functions with named arguments
  - Equality both in rule premises and conclusions
- Also a subset called RIF-CORE
- RIF dialects defined under the RuleML umbrella
  - RIF-CASPD, the core answer set programming dialect
    - Extends BLD with negation based on stable models
  - RIF-CLPWD, the core logic programming dialect based on the well-founded semantics
    - Extends BLD with negation based on the well-founded models
  - RIF-URD, the uncertainty rules dialect
    - Extends BLD with uncertain rules

Current Status

- RIF is good for academia and industry, but
  - Few tools
  - Slow uptake
  - Partly because W3C made it hard to develop something useful for rule systems other than production rules
  - The only thing we could push through was the RIF-FLD framework for defining future RIF dialects.
    - Some useful RIF dialects were defined under RuleML

Implementations

- http://www.w3.org/2005/rules/wiki/Implementations
- Ontobroker
- SILK
- RIF4J
- RIFTR
- … … …

RIF Links

- FLD: http://www.w3.org/TR/rif-bld/
- BLD: http://www.w3.org/TR/rif-bld/

- CASPD: http://ruleml.org/rif/RIF-CASPD.html
- CLPWD: http://ruleml.org/rif/RIF-CLPWD.html
- URD: http://ruleml.org/rif/URSW2008_F9_ZhaoBoley.pdf

Thank You!

Questions?