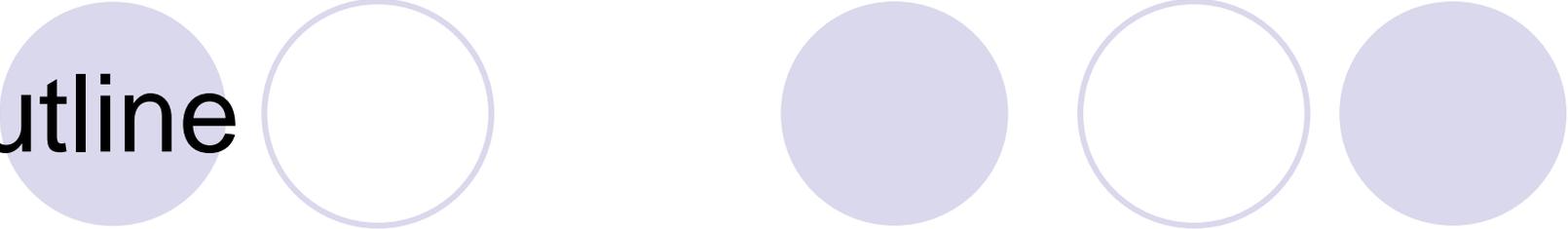


# Rule Interchange Format: The Framework

Michael Kifer  
State University of New York  
at Stony Brook

# Outline



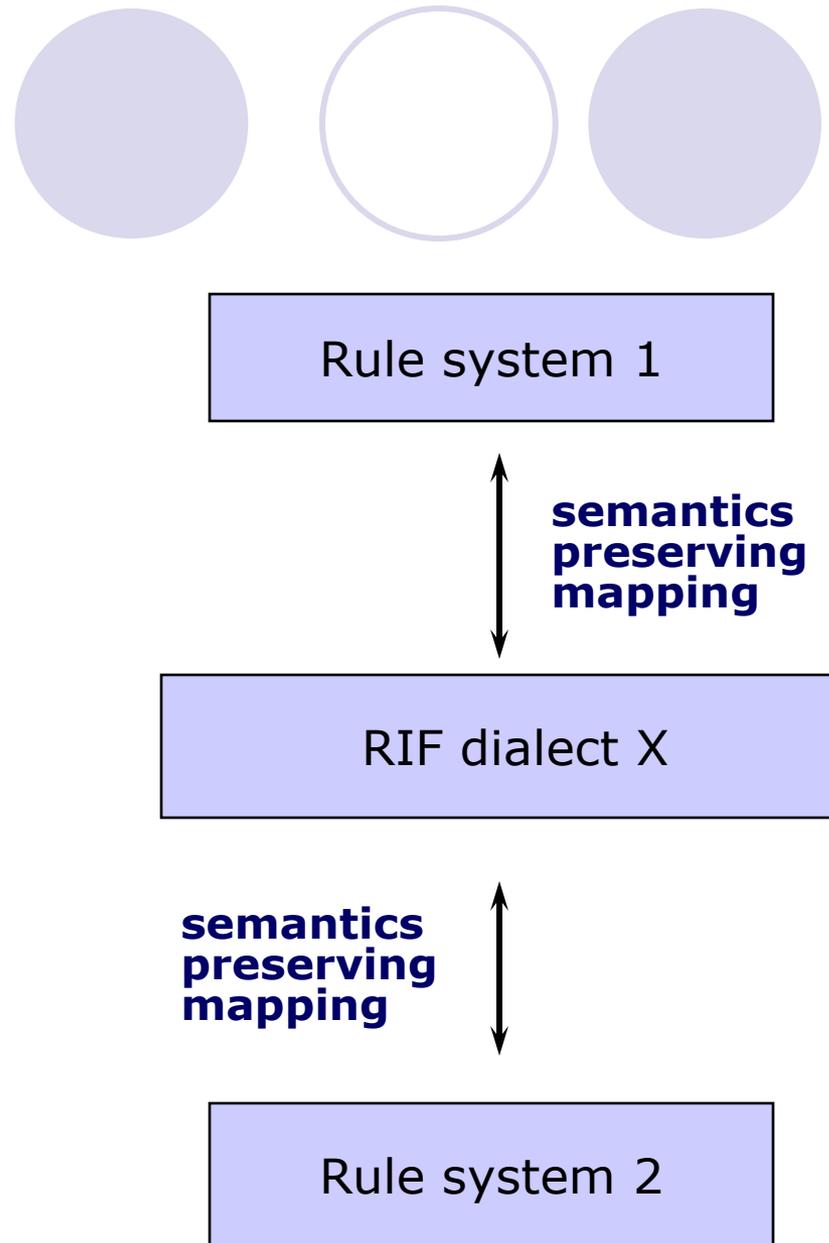
- What is Rule Interchange Format (RIF)?
- RIF Framework
- Basic Logic Dialect (BLD)

# 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



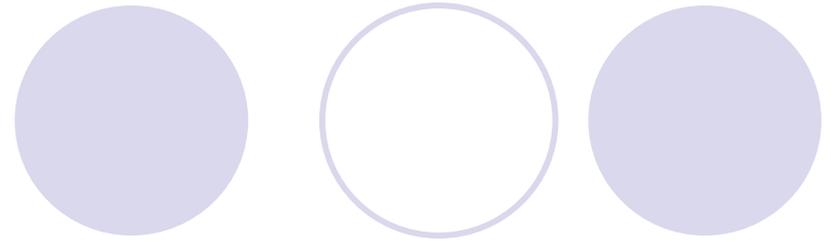
# Why Rule *Exchange*?

(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, syntaxes
- Different commercial interests
- Many egos, different preferences, ...

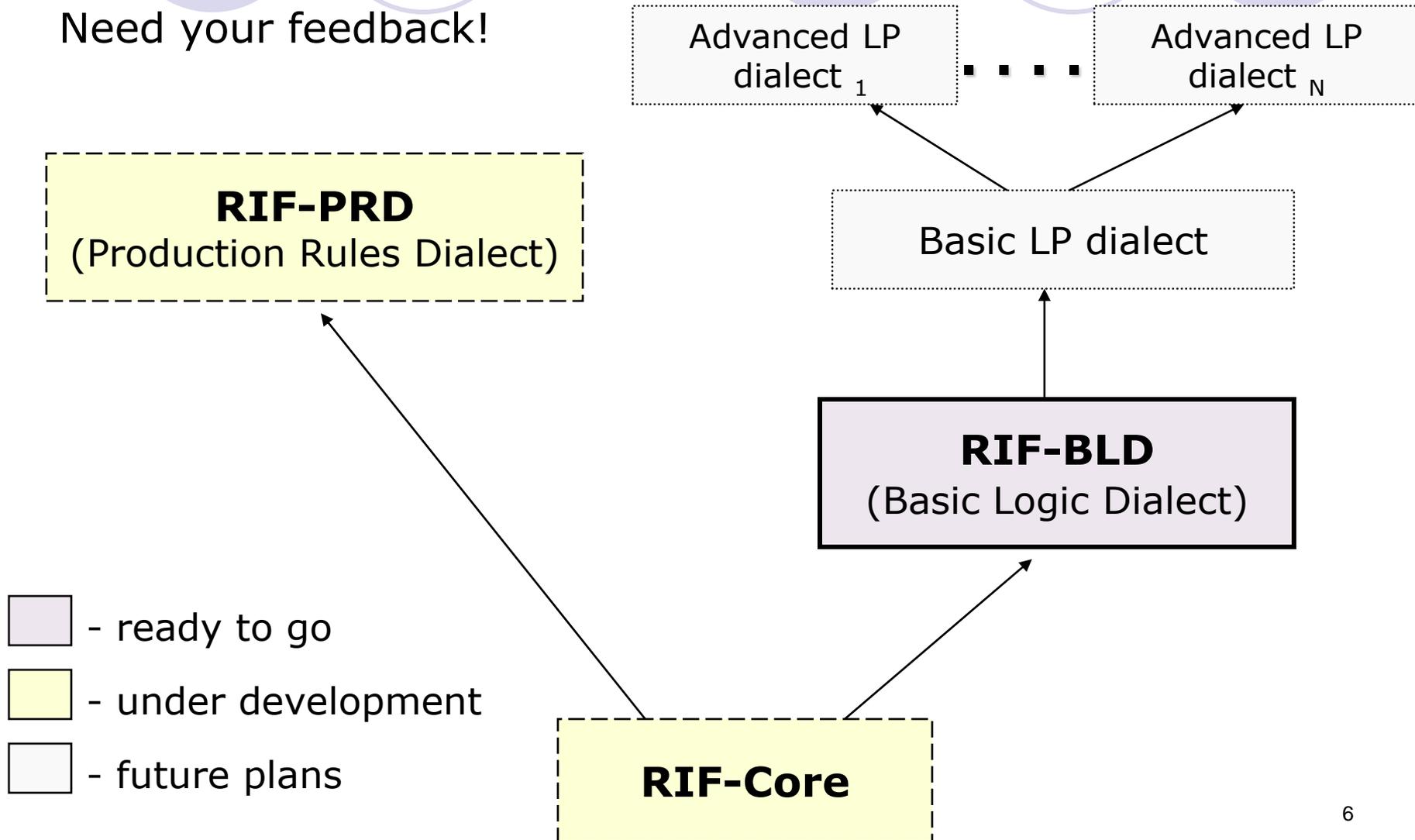
# Why RIF *Dialects*? (and not just *one* dialect)



- 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

Need your feedback!

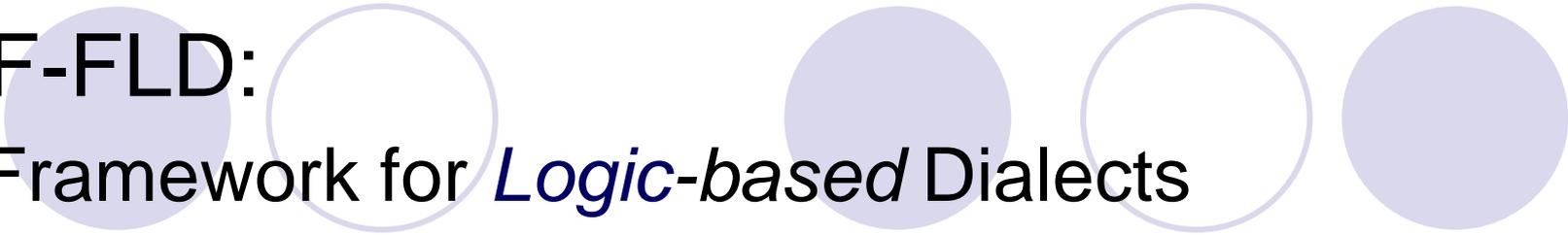


# What Is The RIF Framework?

- A set of rigorous guidelines for constructing RIF dialects in a consistent manner
  - Initially: just the logic-based dialects
- Includes several aspects:
  - Syntactic framework
  - Semantic framework
  - XML framework

# Why Create The 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 another dialect
  - RIF-BLD can be specified in < 3pp in this way
- A “super-dialect” is needed to ensure that all dialects use the same set of concepts and constructs
- RIF Framework is intended to be just such a super-dialect

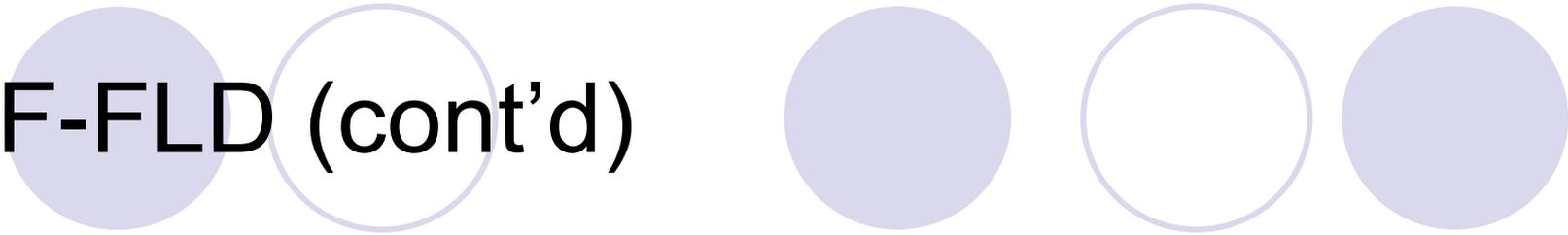


# RIF-FLD:

## A Framework for *Logic-based* Dialects

- Too hard to come to an agreement across all paradigms
- Not clear if a super-dialect for all rule paradigms is feasible
- Defining a super-dialect even for one paradigm (logic) is quite hard
- Logical framework may also help with other paradigms
- So, let's start with just a framework for logic-based dialects (FLD)

# RIF-FLD (cont'd)



- “Super”, but not really a dialect ...  
... rather a framework for dialects
- Very general syntax, but several parameters are not specified – left to dialects
- Very general semantics, but several aspects are under-specified – left to dialects
- General XML syntax – dialects can specialize
- Currently 90% complete

# RIF-FLD's Syntactic Framework

- Presentation syntax
  - Human-oriented
  - Designed for
    - Precise specification of syntax and semantics
    - Examples
    - Perhaps even rule authoring
  - Mappable to XML syntax
- XML syntax
  - For exchange through the wire
  - Machine consumption
- Will use only the presentation syntax in this talk

# RIF-FLD Syntactic Framework (cont'd)

- Must be general (and extensible) so that other dialects' syntaxes could be expressed by *specializing* the syntax of FLD
- Should be interpretable in model-theoretic terms
  - because FLD is intended as a framework for dialects with model-theoretic semantics

# Why So Many Syntactic Forms?

- Richer syntax allows more direct interchange
- Exchange should be *round-trippable*
  - RIF  $\neq$  encoding
    - Translation to RIF and back should preserve modeling aspects of rule sets:
      - Relations should be mapped to relations
      - Objects to objects
      - Subclass/membership to be preserved
      - Etc.
- Otherwise, meaningful sharing and reuse of rules among systems will be impossible

# Examples of Syntactic Forms Supported in RIF-FLD

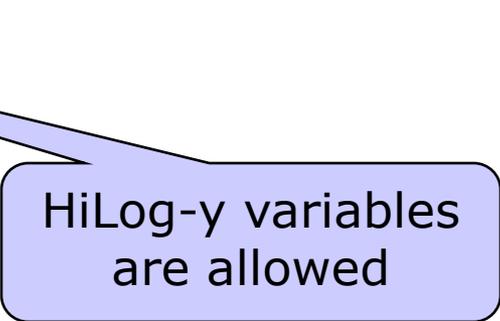
- **Function/predicate application**

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<sub>1</sub>->Val<sub>1</sub> ... Prop<sub>n</sub>->Val<sub>n</sub>]

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

Member#Class

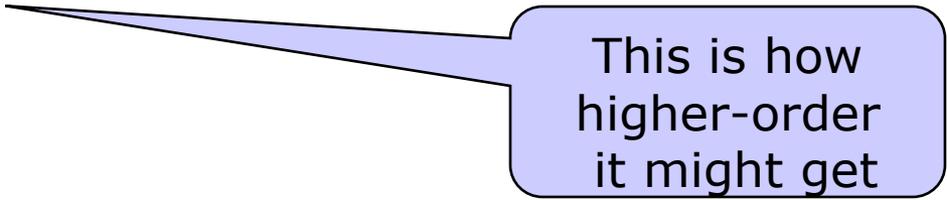
SubCl##SupCl

- Higher-order functions

?F(a)(b c)

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

?O[?P->a](f(?X b) c)



This is how  
higher-order  
it might get

# Examples of Syntactic Forms (cont'd)

- Equality

- Including in rule conclusions

- Negation

- Symmetric (classical, explicit): **Neg**
- Default (various kinds – stable/ASP, well-founded):  
**Naf**

- Connectives, quantifiers

Or (And(?X And p(?X ?Y)) ?Z(p))

Forall ?X ?Y (Exists ?Z

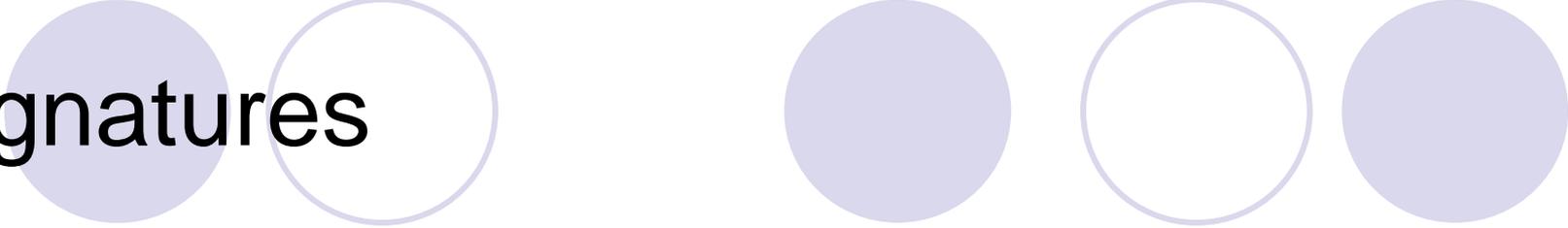
(f(?X(a b)(c)(d ?E) ?X ?Y(ab)(?Z))))

- New connectives/quantifiers can be added

# Syntactic Forms (Cont'd)

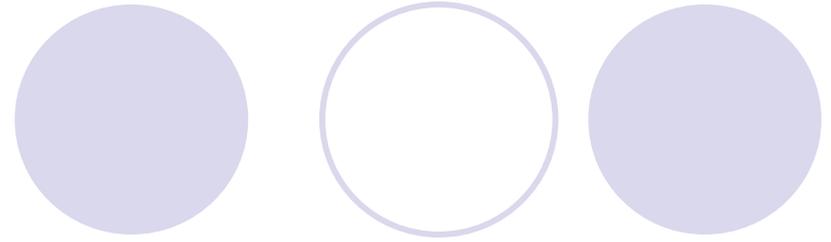
- Some dialects may allow/disallow some syntactic forms
  - For instance, no frames
- Some may restrict certain symbols to only certain contexts
  - For instance, no variables over functions, no higher-order functions
- A syntactic form can occur
  - as a *term* (i.e., in an object position)
  - or as 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

# Signatures (cont'd)



- Arrow expression

$(\text{sigName}_1 \dots \text{sigName}_k) \Rightarrow \text{sigName}$

- Signature

$\text{sigName}\{\text{arrEx}_1, \dots, \text{arrEx}_m\}$

where the  $\text{arrEx}_i$  are arrow expressions

# Examples of Signatures

- Functions of arities 1 or 2:

$\text{fun}_{1,2}\{(\text{obj})\Rightarrow\text{obj}, (\text{obj obj})\Rightarrow\text{obj}\}$

- Unary functions or predicates:

$\text{pf}_1\{(\text{obj})\Rightarrow\text{obj}, (\text{obj})\Rightarrow\text{atomic}\}$

- Higher-order binary functions that yield unary predicates or functions:

$\text{hf}_2\text{pf}_1\{(\text{obj obj})\Rightarrow\text{pf}_1\}$

- Binary functions that take arbitrary predicates as arguments:

$\text{f}_2\text{p}\{(\text{atomic atomic})\Rightarrow\text{obj}\}$

# Signatures of Terms

- Composite terms can also have signatures
- If  $f$  has the signature  $\text{fun}_1\{(\text{obj})\Rightarrow\text{obj}\}$ ,  
and  $t$  has the signature  $\text{obj}$   
*then  $f(t)$  has the signature  $\text{obj}$*
- If  $f$  has the signature  
 $\text{hf}_2p_1\{(\text{obj } \text{obj})\Rightarrow p_1\}$ , where  $p_1$  denotes  
the signature:  $p_1\{(\text{obj})\Rightarrow\text{atomic}\}$ .  
*then  $f(t \ t)(t)$  has the signature  $\text{atomic}$*

# Well-formed Terms and Formulas

- Well-formed *term*

- Each symbol occurs only in the contexts allowed by that symbol's signature

If  $f$  has signature  $\text{fun}_1\{(\text{obj})\Rightarrow\text{obj}\}$ ,

and  $t$  has signature  $\text{obj}$

then  $f(t)$  is well-formed with signature  $\text{obj}$ .

But  $f(t\ t)$  is *not* well-formed

- Well-formed *atomic formula*

- Well-formed term that has the designated signature  $\text{atomic}$

# Is the syntactic framework too fancy?

- Cannot be rich enough!
- Cf. languages like
  - FLORA-2
  - And especially Vulcan's SILK project  
<http://silk.projects.semwebcentral.org>

# RIF-FLD Semantic Framework

- Defines *semantic structures* (a.k.a. *interpretations*)
  - Structures that determine if a formula is true
  - Must be very general to allow:
    - Interpretation of all the supported syntactic forms
    - Higher-order features
    - Reification
  - Multivalued logics, not necessarily Boolean
    - 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)

$p \leftarrow \text{not } p$

- In first-order logic:
  - $\equiv p$
  - 2-valued
- In logic programming:
  - Well-founded semantics
    - $p$  is undefined
    - 3-valued
  - Stable model semantics
    - inconsistent
    - 2-valued
- And there is more ...

# Semantic Framework (cont'd)

- Solution: under-specify
  - Define entailment parametrically, leave parameters to dialects
  - Parameters: *intended models*, truth values, etc.
  - Entailment (between sets of formulas)
    - $P \models Q$  iff for every intended model  $\mathbb{I}$  of  $P$ ,  $\mathbb{I}$  is also a model of  $Q$
  - Overall framework is based on Shoham (IJCAI 1987)

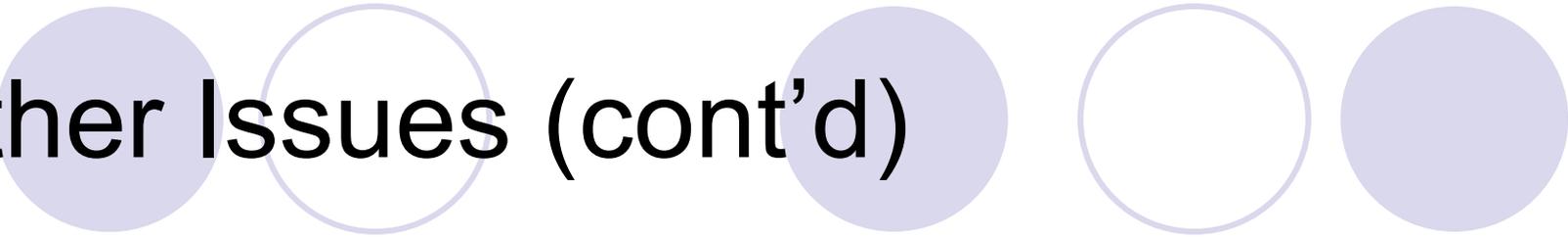
# What Are These Intended Models?

- Up to the dialect designers!
- First-order logic:
  - All models are intended
- Logic programming/Well-founded semantics
  - 3-valued well-founded models are intended
- Logic programming/Stable model semantics
  - Only stable models are intended

# Other Issues: Link to the Web World

- Symbol spaces
  - Partition all constants into subsets; each subset can be given different semantics
- Some RIF symbol spaces
  - `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
- Other symbol spaces: Data types
  - Symbol spaces with fixed interpretation (includes most of the XML data types + more)
- Document formulas, meta-annotations, ...

# Other Issues (cont'd)

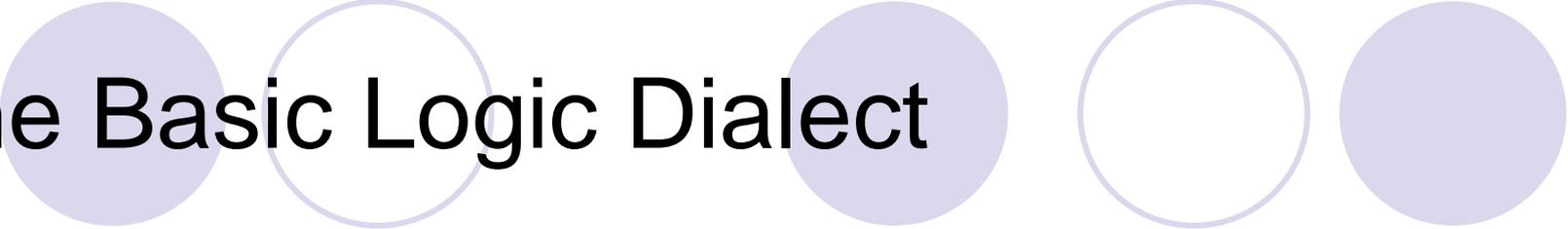


- Built-ins (mostly adapted from XPath)
- Aggregate functions
- Externally defined objects/predicates
  - External sources accessed by rule sets
- Imports/Modules
  - Integration of different RIF rule sets

# Describing RIF Logic-based Dialects By Specializing RIF-FLD

- Syntactic parameters
  - Signatures, syntactic forms
  - Quantifiers, connectives
  - Symbol spaces
  - Types of allowed formulas (e.g., just Horn rules or rules with Naf in premises)
- Semantic parameters
  - Truth values
  - Data types
  - Interpretation of new connectives and quantifiers (beyond And, Or, :-, Forall, Exists)
  - Intended models
- Much easier to specify (< 10% of the size of a direct specification)
- Much easier to learn/understand/compare different dialects

# The Basic Logic Dialect



- Basically Horn rules (no negation) plus
  - Frames
  - Predicates/functions with named arguments
  - Equality both in rule premises and conclusions
  - But no polymorphic or polyadic symbols
- This dialect is called “*basic*” because
  - Here classical semantics = logic programming semantics
  - Bifurcation starts from here on

# Basic Logic Dialect (cont'd)

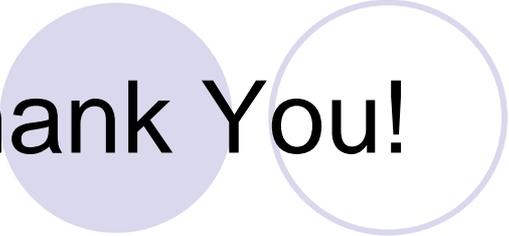
- Can import RDF and OWL
  - RIF RDF+OWL Compatibility document:  
<http://www.w3.org/2005/rules/wiki/SWC>
  - Semantics: a la Rosati et. al.
- BLD with imported OWL
  - Is essentially SWRL (but has frames and other goodies)

# Basic Logic Dialect (cont'd)

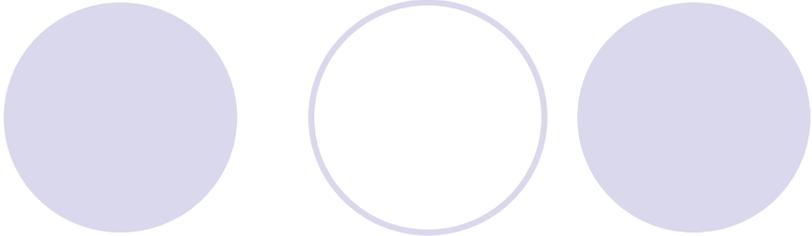
- Specified in two *normative* ways:
  - As a long, direct specification
  - As a specialization from RIF-FLD
- These two specifications are supposed to be equivalent
  - This double specification has already helped debugging both BLD and FLD

# Conclusions

- RIF is good for rules research and industry.
- Need help – take part, save the world!
  
- RIF Web site:  
[http://www.w3.org/2005/rules/wiki/RIF\\_Working\\_Group](http://www.w3.org/2005/rules/wiki/RIF_Working_Group)
- FLD – the latest:  
<http://www.w3.org/2005/rules/wiki/FLD>
- BLD – the latest:  
<http://www.w3.org/2005/rules/wiki/BLD>
- Production rules (in progress):  
<http://www.w3.org/2005/rules/wiki/PRD>

A decorative graphic consisting of two overlapping circles. The left circle is solid light purple, and the right circle is white with a light purple outline. The text "Thank You!" is centered over these circles.

Thank You!

A decorative graphic consisting of three circles in a row. The left and right circles are solid light purple, and the middle circle is white with a light purple outline.

Questions?