Semantic Web

Rule Interchange Format

Where are we?

STI · INNSBRUCK

#	Title
1	Introduction
2	Semantic Web Architecture
3	Resource Description Framework (RDF)
4	Web of data
5	Generating Semantic Annotations
6	Storage and Querying
7	Web Ontology Language (OWL)
8	**Rule Interchange Format (RIF)**
9	Reasoning on the Web
10	Ontologies
11	Social Semantic Web
12	Semantic Web Services
13	Tools
14	Applications

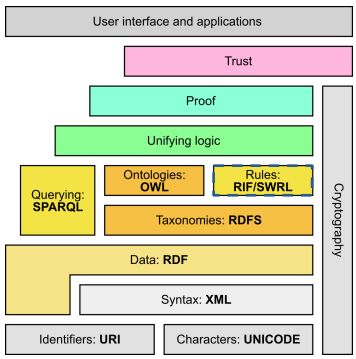
Agenda

STI · INNSBRUCK

1. Introduction and Motivation
2. Technical Solution
   1. Principles
   2. Dialects
   3. Syntax
   4. Semantic
   5. Tools
3. Illustration by an example
4. Extensions
5. Summary
6. References

Semantic Web Stack

STI · INNSBRUCK



Adapted from http://en.wikipedia.org/wiki/Semantic_Web_Stack

STI · INNSBRUCK

MOTIVATION

Why Rule Exchange? (and not The One True Rule Language)

STI · INNSBRUCK

- 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, ...

[Michael Kifer, *Rule Interchange Format: The Framework*]

Why Different Dialects? (and Not Just One Dialect)

STI · INNSBRUCK

- 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

[Michael Kifer, *Rule Interchange Format: The Framework*]

The importance of logic

STI · INNSBRUCK

- Rules needs a proper formalization
  - Disambiguation: a rules should be always be interpreted consistently according its underlying formal semantics
- Rules need to be expressed in a high level language
- Rules are generally structured in the form IF X THEN Y
  - Importance of consequence

Logic provides

- High-level language
- Well-understood formal semantics
- Precise notion of logical consequence
- Proof systems
  - Automatic derivation of statements from a set of premises
  - Sound and complete (Predicate logic)
  - More expressive logics (higher-order logics) are not

First Order Logic in Short

STI · INNSBRUCK

- "Classical" logic
  - Based on propositional logic (Aristotle, ¿300 BC)
  - Developed in 19th century (Frege, 1879)
- Semi-decidable logic
  - Enumerate all true sentences
  - If a sentence is false, the algorithm might not terminate
- FOL is the basis for
  - Logic Programming: Horn Logic
  - Description Logics: 2-variable fragment
- A logic for describing object, functions and relations
  - Objects are "things" in the world: persons, cars, etc.
  - Functions take a number of objects as argument and "return" an object, depending on the arguments: addition, father-of, etc.
  - Relations hold between objects: distance, marriage, etc.
  - Often, a function can also be modeled as a relation

First Order Logic in Short

STI · INNSBRUCK

- Propositional logic deals only with truth-functional validity: any assignment of truth-values to the variables of the argument should make either the conclusion true or at least one of the premises false.
  - All men are mortal
  - Socrates is a man
  - Therefore, Socrates is mortal
- which upon translation into propositional logic yields:
  - A
  - B
  - Therefore C

First Order Logic in Short

STI · INNSBRUCK

- According to propositional logic, this translation is invalid.
- Propositional logic validates arguments according to their structure, and nothing in the structure of this translated argument (C follows from A and B, for arbitrary A, B, C) suggests that it is valid.
- First Order Logic satisfies such needs!
- Is this enough to apply such logic on the Web? Unfortunately no… FOL is not decidable and not efficient.
- Neither families derived from it are ready to scale a the Web size (HL and DL).

Why cannot we use FOL in the SW?

STI · INNSBRUCK

- FOL is undecidable
- Reasoning is hard
- You do not need to use an atomic bomb to kill a mosquito!
- … we need a simpler logic family!
- Can we maintain some of the advantages of FOL and achieve decidability, without stepping back to propositional logic limitations?
- There are many logic families originated from FOL trying to deal with such problems

Horn Logic 1/3

- Simpler knowledge representation
  - "if ... then ..." rules
- Efficient reasoning algorithms, e.g.
  - Forward chaining
  - Backward chaining
  - SLDNF resolution
- Basis for Logic Programming and Deductive Databases

Horn Logic 2/3

- A Horn formula is a disjunction of literals with one positive literal, with all variables universally quantified:
  - $(\forall) \neg B_1 \vee ... \vee \neg B_n \vee H$
- Can be written as an implication:
  - $(\forall) \neg B_1 \vee ... \vee \neg B_n \rightarrow H$
- Decidable reasoning
  - without function symbols
  - limited use of function symbols
    - e.g., no recursion over function symbols

Horn Logic 3/3: Algorithms

- Forward chaining: An inference engine using forward chaining searches the inference rules until it finds one where the antecedent (If clause) is known to be true. When found it can conclude, or infer, the consequent (Then clause), resulting in the addition of new information to its data.
- Backward chaining: An inference engine using backward chaining would search the inference rules until it finds one which has a consequent (Then clause) that matches a desired goal. If the antecedent (If clause) of that rule is not known to be true, then it is added to the list of goals (in order for your goal to be confirmed you must also provide data that confirms this new rule).
- Selective Linear Definite clause resolution is the basic inference rule used in logic programming. SLDNF is an extension to deal with negation as failure.

Description Logics 1/2

- A family of logic based Knowledge Representation formalisms
  - Descendants of semantic networks and KL-ONE
  - Describe domain in terms of concepts (classes), roles (properties, relationships) and individuals
- Distinguished by:
  - Formal semantics (typically model theoretic)
    - Decidable fragments of FOL (often contained in C_2)
    - Closely related to Propositional Modal & Dynamic Logics
    - Closely related to Guarded Fragment
  - Provision of inference services
    - Decision procedures for key problems (satisfiability, subsumption, etc)
    - Implemented systems (highly optimized)

Description Logics 2/2

- Formalization for frame-based knowledge representation
- Frame = all information about a class
  - Superclasses
  - Property restrictions
- Description Logic Knowledge Base
  - Terminological Box (TBox)
    - Class definitions
  - Assertional Box (ABox)
    - Concrete (instance) data

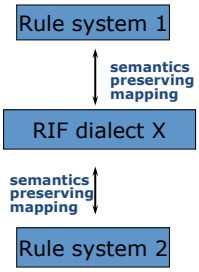
TECHNICAL SOLUTION

A rule interchange format to suite all the needs!

PRINCIPLES

What is the Rule Interchange Format (RIF)?

- A set of dialects to enable rule exchange among different rule systems



Rule Interchange Format Goals

- Exchange of Rules
  - The primary goal of RIF is to facilitate the exchange of rules
- Consistency with W3C specifications
  - A W3C specification that builds on and develops the existing range of specifications that have been developed by the W3C
  - Existing W3C technologies should fit well with RIF
- Wide scale Adoption
  - Rules interchange becomes more effective the wider is their adoption ("network effect")

RIF Requirements 1

- Compliance model
  - Clear conformance criteria, defining what is or is not a conformant to RIF
- Different semantics
  - RIF must cover rule languages having different semantics
- Limited number of dialects
  - RIF must have a standard core and a limited number of standard dialects based upon that core
- OWL data
  - RIF must cover OWL knowledge bases as data where compatible with RIF semantics

[http://www.w3.org/TR/rif-ucr/]

RIF Requirements 2

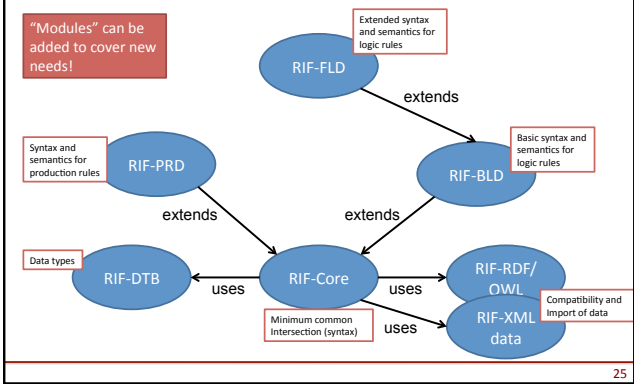
- RDF data
  - RIF must cover RDF triples as data where compatible with RIF semantics
- Dialect identification
  - The semantics of a RIF document must be uniquely determined by the content of the document, without out-of-band data
- XML syntax
  - RIF must have an XML syntax as its primary normative syntax
- Merge rule sets
  - RIF must support the ability to merge rule sets
- Identify rule sets
  - RIF must support the identification of rule sets

[http://www.w3.org/TR/rif-ucr/]

Basic Principle: a Modular Architecture

- RIF wants to cover: rules in logic dialects and rules used by production rule systems (e.g. active databases)
- Logic rules only add knowledge
- Production rules change the facts!
- Logic rules + Production Rules?
  - Define a logic-based core and a separate production-rule core
  - If there is an intersection, define the common core

RIF Architecture



Introduction to RIF Dialects

DIALECTS

RIF Dialects

- RIF Core
  - A language of definite Horn rules without function symbols (~ Datalog)
  - A language of production rules where conclusions are interpreted as assert actions
- RIF BLD
  - A language that lies within the intersection of first-order and logic-programming systems
- RIF FLD
- RIF PRD
- Other common specifications
  - RIF DTB – Defines data types and builtins supported by RIF
  - RIF OWL/RDF compatibility – Defines how OWL and RDF can be used within RIF
  - RIF XML data - Defines how XML can be used within RIF

RIF Core

- The Core dialect of the Rule Interchange Format
- A subset of RIF-BLD and of RIF-PRD
  - a well-formed RIF-Core formula (including document and condition formulas) is also a well-formed RIF-BLD formula
  - 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
  - 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
  - not the *maximal* common subset of RIF-BLD and RIF-PRD
- Based on built-in functions and predicates over selected XML Schema datatypes, as specified in RIF-DTB

RIF Core

STI · INNSBRUCK

- Datalog extensions to support
  - objects and frames as in F-logic
  - internationalized resource identifiers (or IRIs) as identifiers for concepts
  - XML Schema datatypes
  - RIF RDF and OWL Compatibility defines the syntax and semantics of integrated RIF-Core/RDF and RIF-Core/OWL languages
- A Web-aware language
  - Designed to enable interoperability among rule languages in general, and its uses are not limited to the Web

RIF Core Example

STI · INNSBRUCK

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

- The fact **Mary buys LeRif from John** can be logically derived by a *modus ponens* argument.

```
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)
    )
)
```


RIF BLD

STI · INNSBRUCK

- The Basic Logic Dialect of the Rule Interchange Format
- Corresponds to the language of definite Horn rules with equality and a standard first-order semantics
- Has a number of extensions to support
  - objects and frames as in F-logic
  - internationalized resource identifiers (or IRIs) as identifiers for concepts
  - XML Schema datatypes
  - RIF RDF and OWL Compatibility defines the syntax and semantics of integrated RIF-Core/RDF and RIF-Core/OWL languages
- RIF-BLD designed to be a *simple* dialect with limited expressiveness that lies within the intersection of first-order and logic-programming systems
  - RIF-BLD does not support negation

RIF FLD

STI · INNSBRUCK

- The RIF Framework for Logic Dialects
- A formalism for specifying all logic dialects of RIF
  - RIF BLD, RIF Core (albeit not RIF-PRD, as it is not a logic-based RIF dialect)
- Syntax and semantics described mechanisms that are commonly used for various logic languages (but rarely brought all together)
  - Required because the framework must be broad enough to accommodate several different types of logic languages and because various advanced mechanisms are needed to facilitate translation into a common framework
  -
- The needs of future dialects might stimulate further evolution of RIF-FLD

RIF PRD

- The RIF Production Rule Dialect
- A formalism for specifying production rules
  - RIF BLD, RIF Core (albeit not RIF-PRD, as it is not a logic-based RIF dialect)
- Datalog extensions to support
  - objects and frames as in F-logic
  - internationalized resource identifiers (or IRIs) as identifiers for concepts
  - XML Schema datatypes
  - RIF RDF and OWL Compatibility defines the syntax and semantics of integrated RIF-Core/RDF and RIF-Core/OWL languages

RIF PRD Example

A customer becomes a "Gold" customer when his cumulative purchases during the current year reach $5000

```
Prefix(ex <http://example.com/2008/prd1#>)
    Forall ?customer ?purchasesYTD (
        If    And(    ?customer#ex:Customer
                      ?customer[ex:purchasesYTD->?purchasesYTD]
                                  External(pred:numeric-greater-
than(?purchasesYTD 5000))
                  )
        Then Do( Modify(?customer[ex:status->"Gold"]))
    )
```


SYNTAX

Syntactic Framework

- Defines the mechanisms for specifying the formal presentation syntax of RIF logic dialects
- Presentation syntax is used in RIF to define the semantics of the dialects and to illustrate the main ideas with examples
- Syntax is not intended to be a concrete syntax for the dialects
  - the delimiters of the various syntactic components, parenthesizing, precedence of operators, … are left out
- Uses XML as its concrete syntax

BLD Alphabet

STI · INNSBRUCK

- A countably infinite set of constant symbols Const
- A countably infinite set of variable symbols Var (disjoint from Const)
- A countably infinite set of argument names, ArgNames (disjoint from Const and Var)
  - ArgNames is not supported in RIF-Core
- Connective symbols And, Or, and :-
- Quantifiers Exists and Forall
- The symbols =, #, ##, ->, External, Import, Prefix, and Base
  - ##, the subclass symbol is not supported in RIF-Core
- The symbols Group and Document
- The symbols for representing lists: List and OpenList
- The auxiliary symbols (,), [,], <, >, and ^^

BLD Terms I

STI · INNSBRUCK

- Constants and variables
  - If t ∈ Const or t ∈ Var then t is a ***simple term***
- Positional terms
  - If t ∈ Const and t_1, ..., t_n, n≥0, are base terms then $t(t_1 \ldots t_n)$ is a ***positional term***
- Terms with named arguments
  - A ***term with named arguments*** is of the form $t(s_1->v_1 \ldots s_n->v_n)$, where n≥0, t ∈ Const and v_1, ..., v_n are base terms and s_1, ..., s_n are pairwise distinct symbols from the set ArgNames
  - Not valid in RIF-Core
- List terms
  - There are two kinds of list terms: *open* and *closed*
  - A ***closed list*** has the form List($t_1 \ldots t_m$), where m≥0 and t_1, ..., t_m are terms
  - An ***open list*** (or a list with a tail) has the form OpenList($t_1 \ldots t_m$ t), where m>0 and t_1, ..., t_m, t are terms
  - In RIF-Core there are only *closed ground* lists. A ***closed ground list*** is a closed list where t_1, ..., t_m are ground terms (no tail and no variables are allowed)

BLD Terms II

STI · INNSBRUCK

- Equality terms
  - t = s is an ***equality term***, if t and s are base terms
- Class membership terms (or just membership terms)
  - t#s is a ***membership term*** if t and s are base terms
- Subclass terms
  - t##s is a ***subclass term*** if t and s are base terms
  - Not valid in RIF-Core
- Frame terms
  - $t[p_1->v_1 \ldots p_n->v_n]$ is a ***frame term*** (or simply a ***frame***) if t, p_1, ..., p_n, v_1, ..., v_n, n ≥ 0, are base terms
- Externally defined terms
  - If t is a positional or a named-argument term then External(t) is an ***externally defined term***

BLD Terms Example I

STI · INNSBRUCK

- Positional term
  - "http://example.com/ex1"^^rif:iri(1 "http://example.com/ex2"^^rif:iri(?X 5) "abc")
- Term with named arguments
  - "http://example.com/Person"^^rif:iri(id->"http://example.com/John"^^rif:iri "age"^^rif:local->?X "spouse"^^rif:local->?Y)
- Frame term
  - "http://example.com/John"^^rif:iri["age"^^rif:local->?X "spouse"^^rif:local->?Y]
- Empty list
  - List()
- Closed list with variable inside
  - List("a"^^xs:string ?Y "c"^^xs:string)

BLD Terms Example II

- Open list with variables
  - List("a"^^xs:string ?Y "c"^^xs:string | ?Z)
- Equality term with lists inside
  - List(Head | Tail) = List("a"^^xs:string ?Y "c"^^xs:string)
- Nested list
  - List("a"^^xs:string List(?X "b"^^xs:string) "c"^^xs:string)
- Membership
  - ?X # ?Y
- Subclass
  - ?X ## "http://example.com/ex1"^^rif:iri(?Y)

FLD Alphabet (as extension of BLD Alphabet)

- Connective symbols, which includes Naf, Neg, and NEWCONNECTIVE
  - NEWCONNECTIVE is a RIF-FLD extension point
- A countably infinite set of quantifiers including the extension point, NEWQUANTIFIER
- The symbols Dialect, Import, and Module
- A countable set of aggregate symbols
  - RIF-FLD reserves the following symbols for standard aggregate functions: Min, Max, Count, Avg, Sum, Prod, Set, and Bag (and NEWAGGRFUNC)
- Auxiliary symbols {, }, |, ?, @, and an extension point NEWSYMBOL

FLD Terms (as extension of BLD Alphabet)

- Formula term
  - If S is a connective or a quantifier symbol and t_1, ..., t_n are terms then $S(t_1 \ldots t_n)$ is a ***formula term***
- Aggregate term
  - An ***aggregate term*** has the form $sym_{?V[?X1 \ldots ?Xn]}(\tau)$, where $sym_{?V[?X1 \ldots ?Xn]}$ is an aggregate symbol, n≥0, and τ is a term
- Remote term reference
  - A remote term reference (also called remote term) is a term of the form φ@r where φ is a term; r can be a constant, variable, a positional, or a named-argument term. Remote terms are used to query remote RIF documents, called *remote modules*
- NEWTERM
  - This is not a specific kind of term, but an extension point.

FLD Terms Example

- Formula terms
  - :-("p"^^rif:local(?X) ?X("q"^^xs:string)) (usually written as "p"^^rif:local(?X) :- ?X ("q"^^xs:string))
  - $Forall_{?X,?Y}(Exists_{?Z}$("p"^^rif:local(?X ?Y ?Z))) (usually written as Forall ?X ?Y (Exists ?Z ("p"^^rif:local(?X ?Y ?Z)))
  - Or("http://example.com/to-be"^^rif:iri(?X) Neg("http://example.com/to-be"^^rif:iri(?X)))
- Aggregate term
  - avg{?Sal [?Dept]|Exists ?Empl "http://example.com/salary"^^rif:local(?Empl ?Dept ?Sal)}
- Remote term
  - ?O[?N -> "John"^^rif:string "http://example.com/salary"^^rif:iri -> ?S]@"http://acme.foo"^^xs:anyURI

PRD Alphabet

- A countably infinite set of constant symbols Const,
- A countably infinite set of variable symbols Var (disjoint from Const),
- Syntactic constructs to denote:
  - lists,
  - function calls,
  - relations, including equality, class membership and subclass relations
  - conjunction, disjunction and negation,
  - and existential conditions.

PRD Terms

- Constants and variables
  - If $t \in Const$ or $t \in Var$ then t is a ***simple term***
- List terms
  - A ***list*** has the form $List(t_1 \ldots t_m)$, where $m \geq 0$ and $t_1, \ldots, t_m$ are ground terms, i.e. without variables. A list of the form List() (i.e., a list in which m=0) is called the ***empty list***
- Positional terms
  - If $t \in Const$ and $t_1, \ldots, t_n$, $n \geq 0$, are base terms then $t(t_1 \ldots t_n)$ is a ***positional term***

PRD Terms I (Atomic Formulas)

- Positional atomic formulas
  - If $t \in Const$ and $t_1, \ldots, t_n$, $n \geq 0$, are terms then $t(t_1 \ldots t_n)$ is a ***positional atomic formula*** (or simply an ***atom***)
- Equality atomic formulas
  - t = s is an ***equality atomic formula*** (or simply an ***equality***), if t and s are terms
- Class membership atomic formulas
  - t#s is a ***membership atomic formula*** (or simply ***membership***) if t and s are terms. The term t is the *object* and the term s is the *class*

PRD Terms II (Atomic Formulas)

- Subclass atomic formulas
  - t##s is a ***subclass atomic formula*** (or simply a ***subclass***) if t and s are terms
- Frame atomic formulas
  - $t[p_1 \text{->} v_1 \ldots p_n \text{->} v_n]$ is a ***frame atomic formula*** (or simply a ***frame***) if t, $p_1, \ldots, p_n, v_1, \ldots, v_n$, $n \geq 0$, are terms
- Externally defined atomic formulas
  - If t is a positional atomic formula then External(t) is an ***externally defined atomic formula***.

PRD Terms II (Atomic Formulas)

- Subclass atomic formulas
  - t##s is a ***subclass atomic formula*** (or simply a ***subclass***) if t and s are terms
- Frame atomic formulas
  - t[p_1->v_1 ... p_n->v_n] is a ***frame atomic formula*** (or simply a ***frame***) if t, p_1, ..., p_n, v_1, ..., v_n, n ≥ 0, are terms
- Externally defined atomic formulas
  - If t is a positional atomic formula then External(t) is an ***externally defined atomic formula***.

PRD Terms III (Condition Formulas)

- Atomic formula
  - If φ is an atomic formula then it is also a condition formula
- Conjunction
  - If φ_1, ..., φ_n, n ≥ 0, are condition formulas then so is And(φ_1 ... φ_n), called a *conjunctive* formula
  - As a special case, And() is allowed and is treated as a tautology, i.e., a formula that is always true
- Disjunction
  - If φ_1, ..., φ_n, n ≥ 0, are condition formulas then so is Or(φ_1 ... φ_n), called a *disjunctive* formula
  - As a special case, Or() is permitted and is treated as a contradiction, i.e., a formula that is always false

PRD Terms IV (Condition Formulas)

- Negation
  - If φ is a condition formula, then so is Not(φ), called a *negative* formula
- Existentials
  - If φ is a condition formula and $?V_1$, ..., $?V_n$, n>0, are variables then Exists $?V_1$ ... $?V_n$ (φ) is an *existential* formula

PRD Terms V (Atomic Actions)

- Assert
  - If φ is a positional atom, a frame or a membership atomic formula in the RIF-PRD condition language, then Assert(φ) is an atomic action. φ is called the *target* of the action
- Retract
  - If φ is a positional atom or a frame in the RIF-PRD condition language, then Retract(φ) is an atomic action
- Retract object
  - If t is a term in the RIF-PRD condition language, then Retract(t) is an atomic action
- Modify
  - if φ is a frame in the RIF-PRD condition language, then Modify(φ) is an atomic action
- Execute
  - if φ is a positional atom in the RIF-PRD condition language, then Execute(φ) is an atomic action

PRD Terms VI (Atomic Blocks)

- Action variable declaration
  - An ***action variable declaration*** is a pair, *(v p)* made of an *action variable*, *v*, and an *action variable binding* (or, simply, *binding*), *p*, where *p* has one of two forms: *frame object declaration* and *frame slot value*
  - *frame object declaration*: if the action variable, *v*, is to be assigned the identifier of a new frame, then the action variable binding is a *frame object declaration*: New()
  - *frame slot value*: if the action variable, *v*, is to be assigned the value of a slot of a ground frame, then the action variable binding is a frame: *p* = o[s->*v*], where o is a term that represents the identifier of the ground frame and s is a term that represents the name of the slot
- Action block
  - If $(v_1\ p_1)$, ..., $(v_n\ p_n)$, $n \geq 0$, are action variable declarations, and if a_1, ..., a_m, $m \geq 1$, are atomic actions, then $Do((v_1\ p_1) \ldots (v_n\ p_n)\ a_1 \ldots a_m)$ denotes an **action block**

XML Serialization Framework

- Defines
  - a normative mapping from the RIF-FLD presentation syntax to XML
  - and a normative XML Schema for the XML syntax
- Any conformant XML document for a logic RIF dialect must also be a conformant XML document for RIF-FLD
  - i.e. each mapping for a logic RIF dialect must be a restriction of the corresponding mapping for RIF-FLD.
  - e.g. the mapping from the presentation syntax of RIF-BLD to XML in RIF-BLD is a restriction of the presentation-syntax-to-XML mapping for RIF-FLD.

XML Syntax

```
And (Exists ?Buyer (cpt:purchase(?Buyer ?Seller
                    cpt:book(?Author bks:LeRif)
                    curr:USD(49)))
    ?Seller=?Author )

<And><formula><Exists><declare><Var>Buyer</Var></declare>
        <formula><Atom><op><Const type="&rif;iri">&cpt;purchase</Const></op>
                    <args ordered="yes"><Var>Buyer</Var> <Var>Seller</Var>
                      <Expr><op><Const type="&rif;iri">&cpt;book</Const></op>
                          <args ordered="yes">
                            <Var>Author</Var>
                            <Const type="&rif;iri">&bks;LeRif</Const></args>
                      </Expr>
                      <Expr><op><Const type="&rif;iri">&curr;USD</Const></op>
                          <args ordered="yes">
                          <Const type="&xsd;integer">49</Const></args>
                      </Expr>
                    </args>
        </Atom></formula>
    </Exists></formula>
    <formula><Equal>
        <side><Var>Seller</Var></side>
        <side><Var>Author</Var></side>
    </Equal></formula>
</And>
```


SEMANTICS

RIF Dialects and Semantics

STI · INNSBRUCK

Semantics of a dialect is derived from these notions by specializing:

- The effect of the syntax
- Truth values
- Data types
- Logical entailment

BLD Semantics

STI · INNSBRUCK

- The effect of the syntax
  - RIF-BLD does not support negation
- Truth values
  - The set ***TV*** of truth values in RIF-BLD consists of just two values, **t** and **f** such that **f** $<_t$ **t**. Clearly, $<_t$ is a total order here.
- Data types
  - xsd:long
  - xsd:integer
  - xsd:decimal
  - xsd:string
  - xsd:time
  - xsd:dateTime
  - rdf:XMLLiteral
  - rif:text

BLD Semantics

STI · INNSBRUCK

- Logical entailment
  - Logical entailment in RIF is defined with respect to an unspecified set of intended semantic structures and that dialects of RIF must make this notion concrete. For RIF-BLD, this set is defined in one of the two following equivalent ways:
    - as a set of all models
    - as the unique minimal model
  - These two definitions are equivalent for entailment of RIF-BLD conditions by RIF-BLD rulesets, since all rules in RIF-BLD are Horn

BLD Semantic Structures

STI · INNSBRUCK

- A ***semantic structure***, ***I***, is a tuple of the form:
  $<TV, DTS, D, I_C, I_V, I_F, I_{frame}, I_{SF}, I_{sub}, I_{isa}, I_{=}, I_{Truth}>$
  - ***D*** is a non-empty set of elements called the ***domain*** of ***I***
  - ***TV*** denotes the set of truth values that the semantic structure uses
  - ***DTS*** is the set of primitive data types used in ***I***
  - I_C maps constants to elements of ***D***
  - I_V maps variables to elements of D_{ind}.
  - I_F maps ***D*** to functions $D^* \rightarrow D$ (here D^* is a set of all sequences of any finite length over the domain ***D***)
  - I_{SF} interprets terms with named arguments
  - I_{frame} is a total mapping from ***D*** to total functions of the form SetOfFiniteBags($D \times D$) $\rightarrow D$
  - I_{sub} gives meaning to the subclass relationship. It is a total function $D \times D \rightarrow D$
  - I_{isa} gives meaning to class membership. It is a total function $D \times D \rightarrow D$
  - $I_{=}$ gives meaning to the equality. It is a total function $D \times D \rightarrow D$
  - I_{Truth} is a total mapping $D \rightarrow TV$
  - It is used to define truth valuation of formulas

BLD Interpretation of Formulas I

Truth valuation for well-formed formulas in RIF-BLD is determined using the following function, denoted $TVal_I$

•Positional atomic formulas: $TVal_I(r(t_1 \ldots t_n)) = \boldsymbol{I}_{Truth}(\boldsymbol{I}(r(t_1 \ldots t_n)))$

•Atomic formulas with named arguments: $TVal_I(p(s_1\text{->}v_1 \ldots s_k\text{->}v_k)) = \boldsymbol{I}_{Truth}(\boldsymbol{I}(p(s_1\text{->}v_1 \ldots s_k\text{->}v_k)))$

•Equality: $TVal_I(x = y) = \boldsymbol{I}_{Truth}(\boldsymbol{I}(x = y))$.

•Subclass: $TVal_I(sc \#\# cl) = \boldsymbol{I}_{Truth}(\boldsymbol{I}(sc \#\# cl))$

•Membership: $TVal_I(o \# cl) = \boldsymbol{I}_{Truth}(\boldsymbol{I}(o \# cl))$

•Frame: $TVal_I(o[a_1\text{->}v_1 \ldots a_k\text{->}v_k]) = \boldsymbol{I}_{Truth}(\boldsymbol{I}(o[a_1\text{->}v_1 \ldots a_k\text{->}v_k]))$.

61

BLD Interpretation of Formulas II

- Conjunction: $TVal_I(And(c_1 \ldots c_n)) = \min_t(TVal_I(c_1), \ldots, TVal_I(c_n))$.
- Disjunction: $TVal_I(Or(c_1 \ldots c_n)) = \max_t(TVal_I(c_1), \ldots, TVal_I(c_n))$.
- Quantification: $TVal_I(Exists\ ?v_1 \ldots ?v_n\ (\varphi)) = \max_t(TVal_{I'}(\varphi))$ and $TVal_I(Forall\ ?v_1 \ldots ?v_n\ (\varphi)) = \min_t(TVal_{I'}(\varphi))$.
- Rules: $TVal_I(conclusion :\!- condition) = \mathbf{t}$, if $TVal_I(conclusion) \geq_t TVal_I(condition)$; $TVal_I(conclusion :\!- condition) = \mathbf{f}$ otherwise.

A ***model*** of a set Ψ of formulas is a semantic structure ***I*** such that $TVal_I(\varphi) = \mathbf{t}$ for every $\varphi \in \Psi$. In this case, we write $\boldsymbol{I} \models \Psi$.

62

BLD Logical Entailment

- Let **R** be a set of RIF-BLD rules and φ an existentially closed RIF-BLD condition formula. We say that **R** ***entails*** φ, written as **R** |= φ, if and only if for every semantic structure ***I*** of **R** and every $\psi \in \mathbf{R}$, it is the case that $TVal_I(\psi) \leq TVal_I(\varphi)$.
- Equivalently, we can say that **R** |= φ holds iff whenever ***I*** |= **R** it follows that also ***I*** |= φ.

63

PRD semantics

- Patterns and conditions have a model-theoretic semantics (compatible with BLD)
- Groups of production rules have an operational semantics as a labelled transition system
- Actions define the transition relation
- Metadata has no semantics

64

PRD: Condition satisfaction and matching substitution

- A set of ground formulas Φ = {φ1, ..., φm}, m ≥ 1 satisfies a condition formula ψ iff
  - Φ |= (Exists ?v0 ... ?vn (ψ)), where {?v0, ..., ?vn}, n ≥ 0 = Var(ψ)
- Let ψ be a condition formula; let σ be a ground substitution for the free variables of ψ, that is, such that: Var(ψ) ⊆ Dom(σ); and let Φ be a set of ground formulas that satisfies ψ. We say that σ is matching ψ to Φ (or simply, matching, when there is no ambiguity with respect to ψ nor Φ) if an only if, for every semantic structure I that is a model of all the ground formulas φi in Φ, there is at least one semantic structure I*, such that:
  - I* is a model of ψ: I* |= ψ;
  - I* is exactly like I, except that a mapping I*V is used instead of IV, such that I*V is defined to coincide with IV on all variables except, possibly, on the variables ?v0 ... ?vn that are free in ψ, that is, such that: Var(ψ) = {?v0 ... ?vn};
  - I*V(?xi) = IC(σ(?xi)), for all ?xi in Var(ψ).

PRD: Operational semantics

- →RIF-PRD : w $\xrightarrow{\text{actions}}$ w'
  - w → w': w' |= φ if and only if And(w, f) |= φ
- →RS : w $\xrightarrow{\text{Assert(f)}}$ w'
  - actions = extractACTIONS(PICK(instances(w), strategy))
  - (w, actions, w') if $\xrightarrow{\text{actions}}$*RIF-PRD
  - instance(w') = {(r σ) | r ∈ RS and σ is matching w}
  - w ∉ TRS
- Halts iff FINAL(c, RS)
  - FINAL(w, RS) if PICK(w, strategy) = {}

Core Semantics

- Model theoretic semantics of RIF BLD restricted to Core syntax
  - Essentially datalog
  - No subclass, no logic functions, no name argument
  - Only atoms and frames in the head
- Operational semantics of RIF PRD, restricted to Core syntax
  - No negation
  - Pattern2Condition equivalence
  - Only Asserts in RHS: Do2And equivalence

An example of usage of RIF

ILLUSTRATION BY AN EXAMPLE

Collaborative Policy Development for Dynamic Spectrum Access

STI · INNSBRUCK

- Recent technological and regulatory trends are converging toward a more flexible architecture in which reconfigurable devices may operate legally in various regulatory and service environments
- Suppose the policy states:
  - A wireless device can transmit on a 5 GHz band if no priority user is currently using that band
- Suppose devices with different rules:
  1. If no energy is detected on a desired band then assume no other device is using the band
  2. If a control channel communicates that a signal is detected on a particular device , then the device is being used

RIF-BLD Example Device Rule 1

STI · INNSBRUCK

If no energy is detected on a desired band then assume no other device is using the band

```
Document(
    Prefix(pred http://www.w3.org/2007/rif-builtin-predicate#)
    Prefix(func http://www.w3.org/2007/rif-builtin-function#)
    Prefix(ex <http://example.com/2009/ucr4.2.3#>)

    Group(
        Forall ?device ?band? ?user
            If And (ex:detectenergyonband(?user ?band)
                    ?band[energylevel-> ?level]
                    External(pred:numeric-greater-than(?level 0)))
            Then Do (Assert (ex:useband(?user ?band)))
    )
)
```


RIF-BLD Example Device Rule 2

STI · INNSBRUCK

If a control channel communicates that a signal is detected on a particular device , then the device is being used

```
Document(
    Prefix(pred http://www.w3.org/2007/rif-builtin-predicate#)
    Prefix(func http://www.w3.org/2007/rif-builtin-function#)
    Prefix(ex <http://example.com/2009/ucr4.2.3#>)

    Group(
        Forall ?device ?band ?user ?controlchannel
            If And (ex:iscontrolchannel(controlchannel)
                    ex:communicatesignal(?controlchannel ?user
                    ?device ?band))
            Then Do (Assert (ex:useband(?user ?band)))
    )
)
```


Why RIF is Perfect in This Example?

STI · INNSBRUCK

- Each type of device will need to employ different "interpretations" or "operational definitions" of the policy in question.
- Suppose
  - 10 manufacturers of these 2 different types of wireless devices
  - Each of these manufacturers uses a distinct rule-based platform
  - Each manufacturer needs to write 2 interpretations of the policy (for each of the two types of device).
- That means that 20 different versions of the policy must be written, tested and maintained.
- This can be automated adopting RIF as interchange format and automating the translation process

EXTENSIONS

Current Ongoing Works

- RIF is still under development
- Other dialects are foreseen
  - A logic programming dialect that support well-founded and stable-model negation
  - A dialect that supports higher-order extensions
  - A dialect that extends RIF-BLD with full F-logic support

SUMMARY

Summary

- RIF is an interchange rule format
  - Enable to exchange rules across different formalisms
- RIF is based on different dialects
  - FLD
  - BLD
  - PRD
  - Core
- RIF is OWL and RDF compatible

References

- Mandatory reading:
  - RIF Web site:
    - http://www.w3.org/2005/rules/wiki/RIF_Working_Group
- Further reading:
  - CORE
    - http://www.w3.org/TR/rif-core/
  - BLD
    - http://www.w3.org/TR/rif-bld/
  - FLD
    - http://www.w3.org/TR/rif-fld/
  - Production rules:
    - http://www.w3.org/TR/rif-prd/

References

- Wikipedia links:
  - http://en.wikipedia.org/wiki/Rule_Interchange_Format
  - http://en.wikipedia.org/wiki/Production_rule
  - http://en.wikipedia.org/wiki/Production_system
  - http://en.wikipedia.org/wiki/Horn_logic
  - http://en.wikipedia.org/wiki/Datalog

Next Lecture

#	Title
1	Introduction
2	Semantic Web Architecture
3	Resource Description Framework (RDF)
4	Web of data
5	Generating Semantic Annotations
6	Storage and Querying
7	Web Ontology Language (OWL)
8	Rule Interchange Format (RIF)
9	**Reasoning on the Web**
10	Ontologies
11	Social Semantic Web
12	Semantic Web Services
13	Tools
14	Applications

Questions?