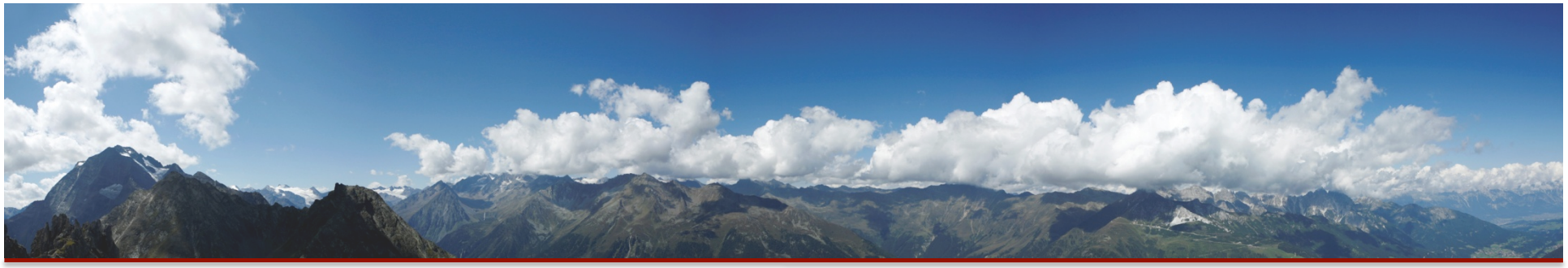
Semantic Web

Rule Interchange Format



Where are we?

STI · INNSBRUCK

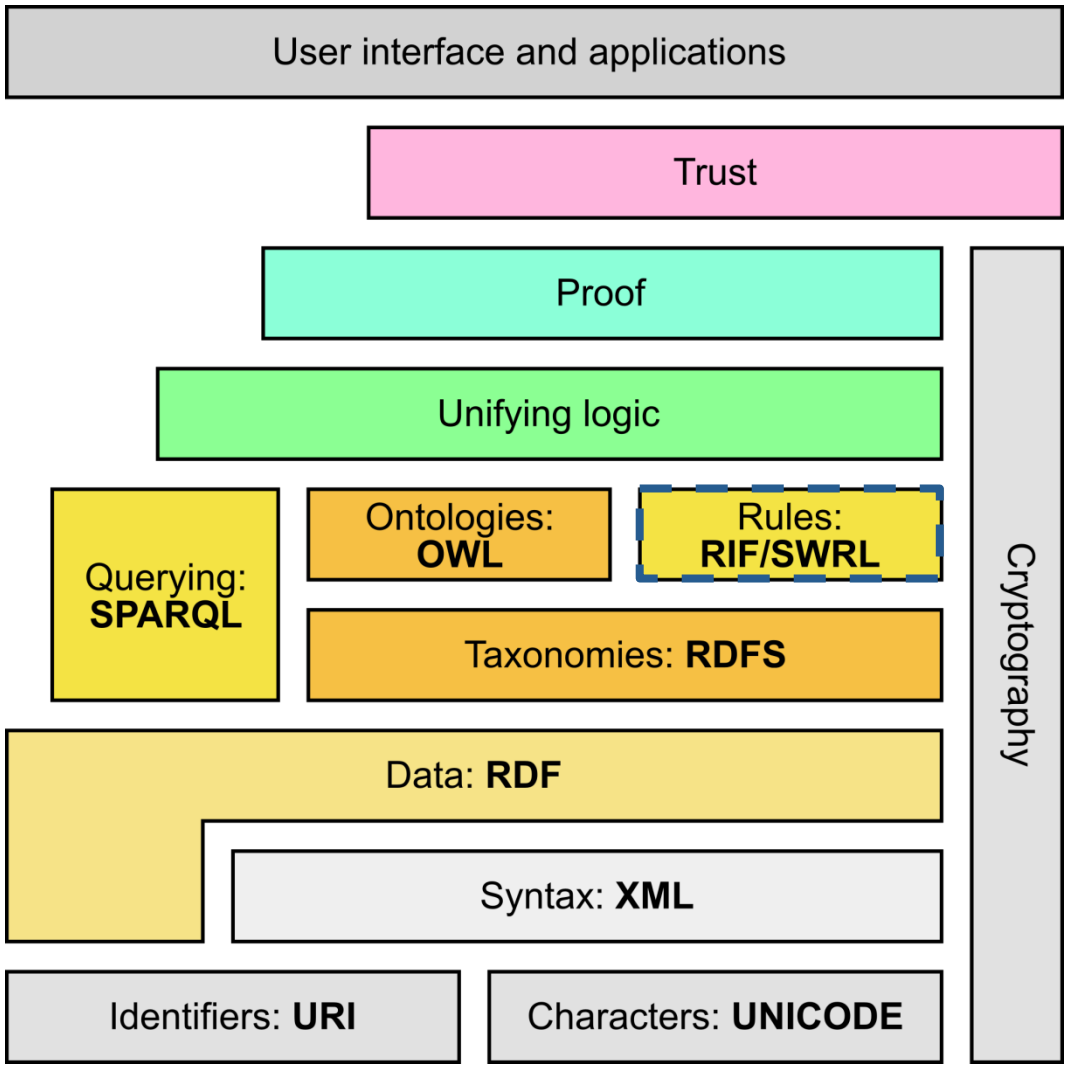
#	Title
1	Introduction
2	Semantic Web architecture
3	Resource Description Framework
4	Semantic Web of hypertext and Web of data
5	Generating Semantic Annotations
6	Repositories
7	OWL
8	**RIF**
9	Web-scale reasoning
10	Social Semantic Web
11	Ontologies and the Semantic Web
12	Service Web
13	Semantic Web Tools
14	Semantic Web Applications
15	Exam

Agenda

STI · INNSBRUCK

1. Introduction and Motivation
2. Technical Solution
   1. The Rule Interchange Format (RIF)
   2. RIF Framework
   3. Basic Logic Dialect (BLD)
3. Illustration by a large example
4. Extensions
5. Summary
6. References

Semantic Web Stack



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

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)

- 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*]

Some Background on Logics …

STI · INNSBRUCK

- Logic is the study of the principles of valid demonstration and inference
- Logic concerns the structure of statements and arguments, in formal systems of inference and natural language

  [http://en.wikipedia.org/wiki/Logic]

The importance of logic

- 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

Why cannot we use FOL in the SW?

STI · INNSBRUCK

- FOL is undecidable
- Reasoning is hard

- … we need a simpler logic family!

Horn Logic 1/2

- 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 2/2

STI · INNSBRUCK

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

Description Logics 1/2

STI · INNSBRUCK

- 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

STI · INNSBRUCK

- 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

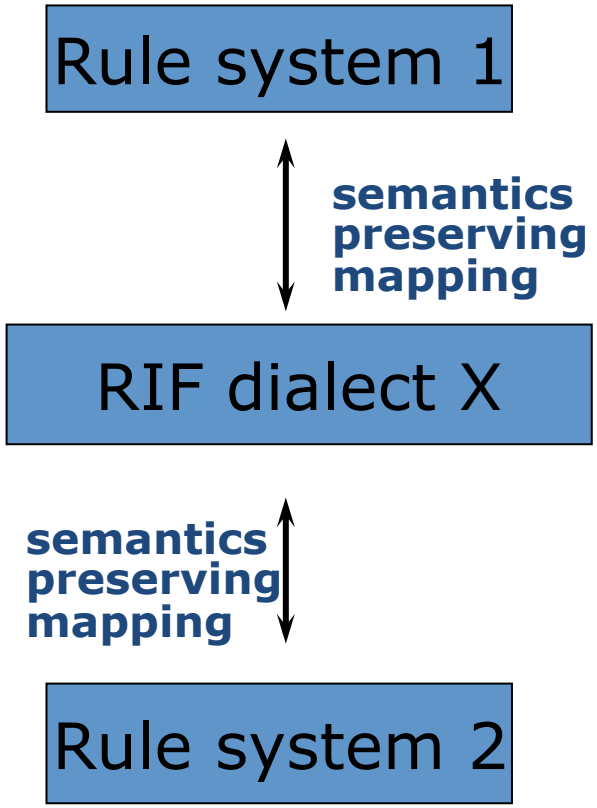
An exchange format for rules

THE RULE INTERCHANGE FORMAT

What is the Rule Interchange Format (RIF)?

STI · INNSBRUCK

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



Rule Interchange Format Goals

STI · INNSBRUCK

- 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
- Widescale Adoption
  - Rules interchange becomes more effective the wider adoption there is of the specification ("network effect“)

RIF Requirements 1

STI · INNSBRUCK

- 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

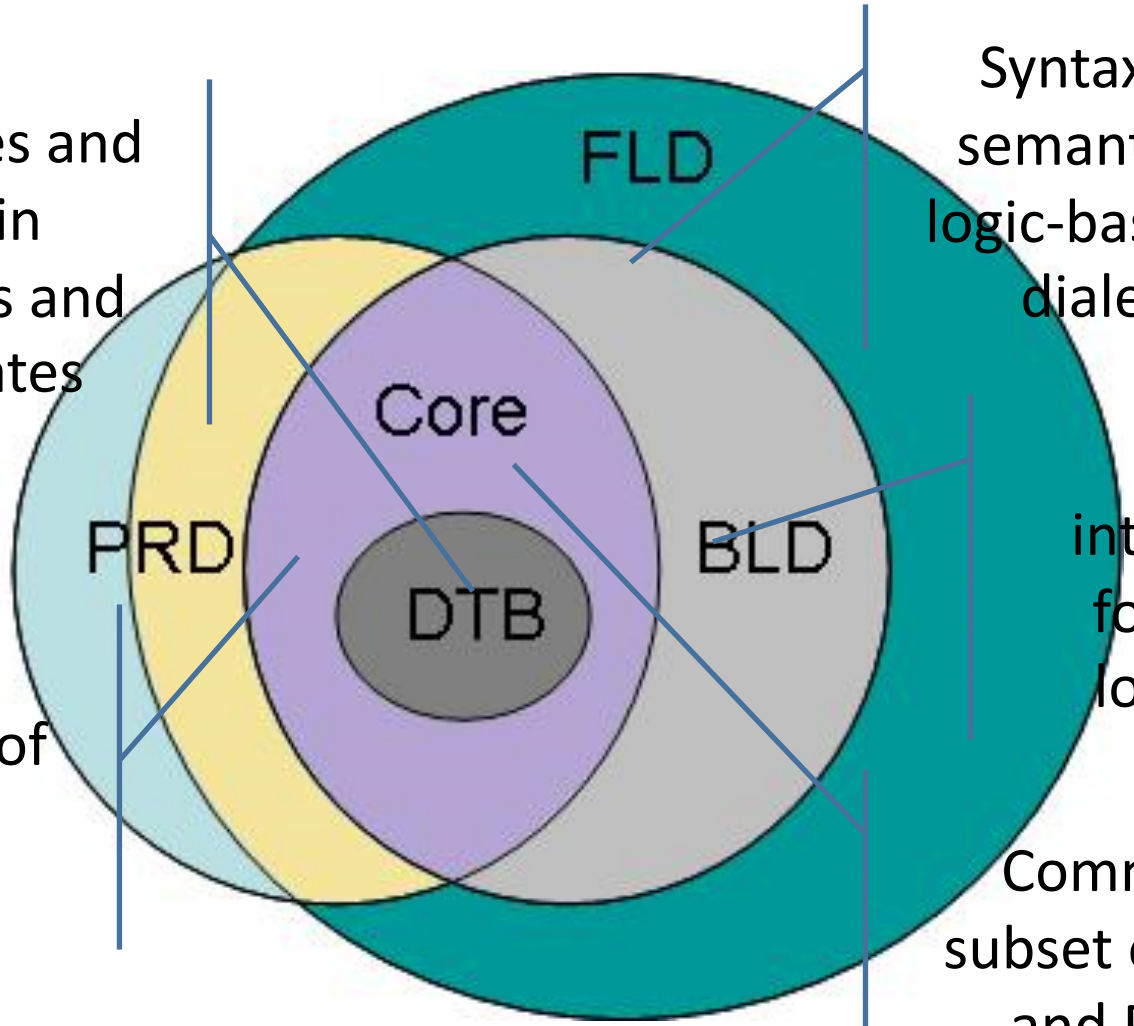
STI · INNSBRUCK

- 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/]

RIF Family Dialects

STI · INNSBRUCK



Data types and Built-in functions and predicates

Syntax and semantics of logic-based RIF dialects

Basic interchange format for logic rules

Enable the interchange of production rules

Common subset of BLD and PRD

RIF Example

STI · INNSBRUCK

- All movies listed at http://altmd.example.org but not listed at http://imd.example.org are independent movies.

```
?Movie#ex:IndependentMovie :-
listed(?Movie#ex:Movie,<http://altmd.example.org>)
not(listed(?Movie#ex:Movie,<http://
imd.example.org>)).
```

- All movies with budgets below 5 million USD are low-budget movies

```
?Movie#ex:LowBudgetMovie :-
?Movie#ex:Movie [date -> ?Date, budget -> ?Budget]
?Budget < 5000000^^xs:long.
```

Syntax and semantics of logic-based RIF dialects

FRAMEWORK OF LOGIC DIALECT

What Is The RIF Framework of Logic Dialect?

STI · INNSBRUCK

- 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

Syntactic Framework

STI · INNSBRUCK

- 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

Examples of Syntactic Forms 1

STI · INNSBRUCK

- Function/predicate application
  - Point(?X abc)
  - ?X(Amount(20) ?Y(cde fgh))
- Functions/predicates with named arguments
  - ?F(name->Bob age->15)

Examples of Syntactic Forms 2

STI · INNSBRUCK

- Frame (object-oriented F-logic notation)
  - Obj[Prop1->Val1 ... Propn->Valn]
- 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)

Examples of Syntactic Forms 3

STI · INNSBRUCK

- Equality
  - Including in rule conclusions
- Negation
  - Symmetric (classical, explicit): Neg
  - Default (various kinds – stable, 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

Symbols

STI · INNSBRUCK

- Used to identify constants, variables, functions, predicates
- "*literal*"^^<symspace-*identifier*>
  - Notable symbol spaces: xsd:string, rif:local, rif:iri
  - “Chris”^^<xsd:string>
  - “http://www.w3.org/1999/02/22-rdf-syntax-ns#type”^^<rif:iri>
  - “Person1”^^rif:local

STI · INNSBRUCK

- Rules occur in Groups

```
Group(  (Forall ?x _Q(?x) :- _P(?x))
        (Forall ?x _Q(?x) :- _R(?x)) )
```

- Groups occur in Documents

```
Document(
  Group((Forall ?x _Q(?x) :- _P(?x))
        (Forall ?x _Q(?x) :- _R(?x)))
  Group((Forall ?y _R(?y) :- ex:op(?y))))
```

FDL Example: Database mapping

STI · INNSBRUCK

```
Document( Prefix(dbp http://dbpedia.org/property/)
   Prefix(my http://mydata.org/resource#)
   Prefix(rdfs http://www.w3.org/2000/01/rdf-schema#)
   Group ( Forall ?mname ?aname ?movie ?actor
       my:actorIn(?aname ?mname) :-
       And( dbp:starring(?movie ?actor)
            rdfs:label(?movie ?mname)
            rdfs:label(?actor ?aname))))
```

Semantic Framework

STI · INNSBRUCK

- 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

- 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 …

- p <- not p
  - In first-order logic:
    - ≡ 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

STI · INNSBRUCK

- Solution: under-specify
  - Define entailment parametrically, leave parameters to dialects
  - Parameters: intended models, truth values, etc.
  - Entailment (between sets of formulas)
    - P |= Q iff for every intended model I of P, I is also a model of Q

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

XML Serialization Framework

STI · INNSBRUCK

- 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

STI · INNSBRUCK

```
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>
```

Basic interchange format for logic rules

BASIC LOGIC DIALECT

Basic Logic Dialect

STI · INNSBRUCK

- 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

STI · INNSBRUCK

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

Basic Logic Dialect

- 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

BLD Example 1

STI · INNSBRUCK

- A rule can be written in English to derive the buy relationships (rather than store them) from the sell relationships that are stored as facts:
  - A buyer buys an item from a seller if the seller sells the item to the buyer
  - John sells LeRif to Mary
- Using the modus pones argument we can derive:
  - Mary buys LeRif from John

BLD Example 2

STI · INNSBRUCK

```
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)
   )
)
```

Interchange of production rules

PRODUCTION RULE DIALECT

Production Rule Dialect

STI · INNSBRUCK

- Production rules are rule statements defined in terms of both individual facts or objects, and groups of facts or classes of objects
- They have
  - if part (or condition)
  - then part (or action)
- The condition is like the condition part of logic rules
- The then part contains actions
  - Not the same as the conclusion part of logic rules
  - Actions can add, delete, or modify facts in the knowledge base, and have other side-effects.

Production Rule Examples

STI · INNSBRUCK

1. A customer becomes a "Gold" customer as soon as his cumulative purchases during the current year top $5000
2. Customers that become "Gold" customers must be notified immediately, and a golden customer card will be printed and sent to them within one week
3. For shopping carts worth more than $1000, "Gold" customers receive an additional discount of 10% of the total amount

Production Rules Dialect Syntax

STI · INNSBRUCK

- Production rules
  - *FOR <variables> WITH <binding patterns>, IF <condition> THEN <actions>*
  - FORALL Var* (actions :- And(patterns, condition))
- Patterns and condition
  - Conjunction, disjunction, negation, existentials
  - Relations, frames (and objects), membership, subclass
  - XML Schema data types and builtins (RIF-DTB)
  - internationalized resource identifiers (IRIs) as identifiers
  - RIF-BLD condition language minus logic functions plus negation
- Assert, Retract, New
  - Assign, Modify, Execute
- Rules occur in Groups
- Groups occur in Documents
  - or in other groups
- Metadata can be attached to any class element

Example in RIF-PRD

STI · INNSBRUCK

```
Prefix(ex1 http://example.com/2008/prd#)
 (* ex1:rule_1 *)
 Forall ?customer ?purchasesYTD (
  If And( ?customer#ex1:Customer
   ?customer[ex1:purchasesYTD>?purchasesYTD]
   External(pred:numeric-greater-than(?purchasesYTD
     5000)))
  Then ex1:Gold(?customer)
 )
```

An example of usage of RIF

ILLUSTRATION BY A LARGER 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 no control signal indicating use of a desired band by a priority user is detected then assume the band is available

RIF-BLD Example Device Rule 1

STI · INNSBRUCK

```
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.org/example#)

   Group(
        Forall ?device ?band ?level (
               ex:used(?device ?band) :-
               ex:detect(ex:energy(?level ?band))
               External(pred:numeric-greater-than(?level 0))
        )
   )
)
```

RIF-BLD Example Device Rule 2

STI · INNSBRUCK

```
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.org/example#)

   Group(
        Forall ?device ?band ?user (
               ex:used(?device ?band) :-
               ex:detect(ex:signal(?user ?band))
               ex:priority(?user,"high").
        )
   )
)
```

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

STI · INNSBRUCK

- 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

STI · INNSBRUCK

- 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

STI · INNSBRUCK

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

Next Lecture

STI · INNSBRUCK

#	Title
1	Introduction
2	Semantic Web architecture
3	Resource Description Framework
4	Semantic Web of hypertext and Web of data
5	Generating Semantic Annotations
6	Repositories
7	OWL
8	RIF
9	**Web-scale reasoning**
10	Social Semantic Web
11	Ontologies and the Semantic Web
12	Service Web
13	Semantic Web Tools
14	Semantic Web Applications
15	Exam

Questions?

STI · INNSBRUCK