AIFB

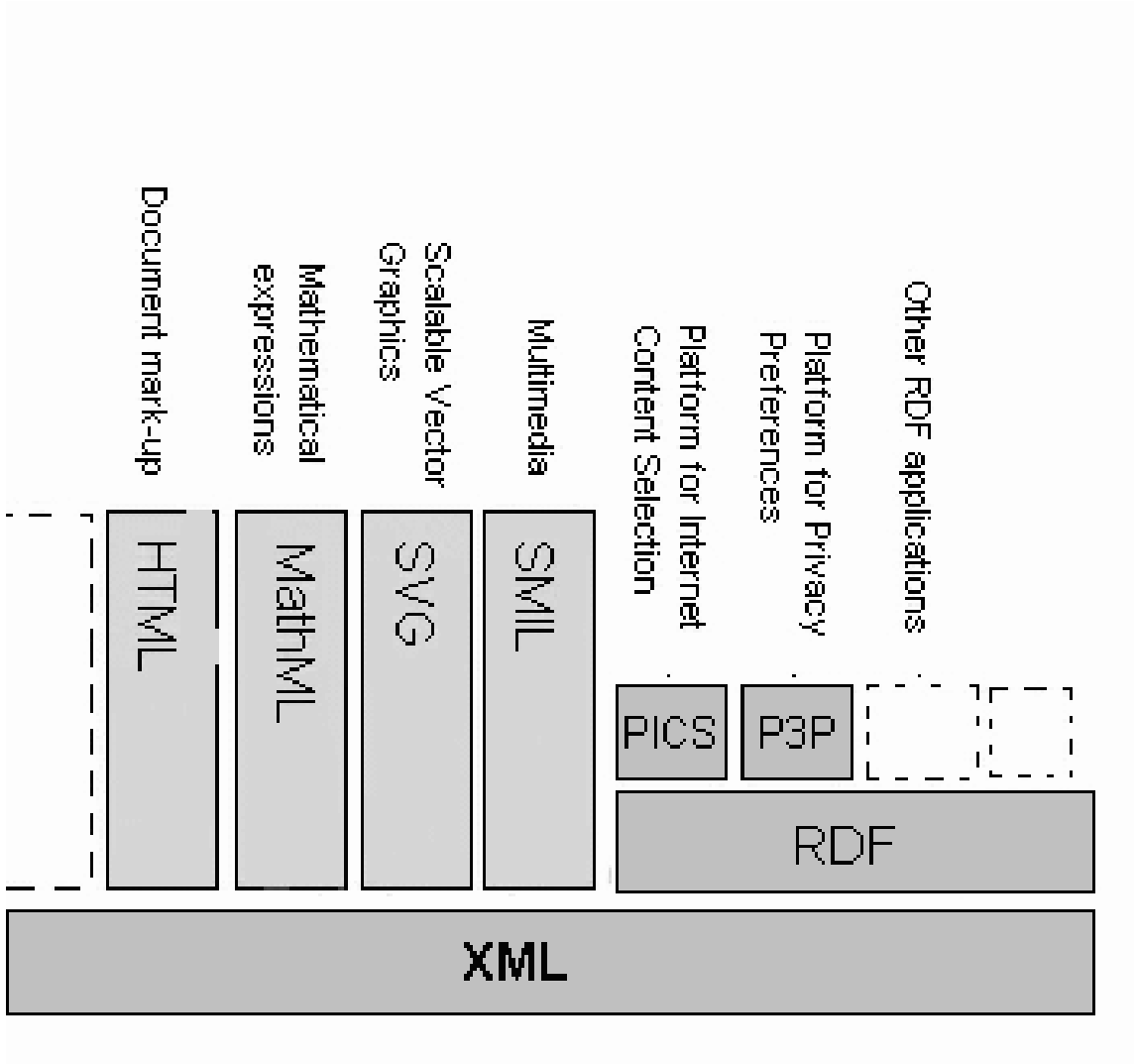
# Intelligent Systems on the World Wide Web

## RDF


Lecture Slides
York Sure
Institut AIFB, Universität Karlsruhe

AIFB

# Applications and Technologies on top of XML



# Outline


- Motivation: Why XML is not enough
- Introduction to RDF
  - Requirements for KR on the Web
  - The RDF Data Model
  - RDF Schema
- Extensions of RDF(S)
- Tools for RDF and RDF Schema
  - Parser, Query, and Inference Engines

# Why The Shift Towards More Semantics?

- Information Overload
  - Information on the Web currently aiming at Human Consumption
  - Information Consumption is to time consuming
- Search Engines fail more and more
  - combined coverage is less than 42% of the HTML-Web
- Data Interchange growing (eg. B2B)
  - needs a common semantics

*XML?*

AIFB

# Extensible Markup Language (XML) Revisited

- Key idea: separate structure  from presentation
- XML DTDs or Schema define document structure
- Replace HTML with two things
  - **A domain specific markup language (defined in XML)**
  - **A map from that markup language to HTML (defined using XSL)**
- DTD enables document recipients to tell whether they've received a well-formed document
  - Gives a minimal level of validation

RDF

# Why XML is not enough

- Only advantage of using XML is reusing the parser and document validation
- Many different possibilities to encode a domain of discourse
- Leads to difficulties when understanding of foreign documents is required

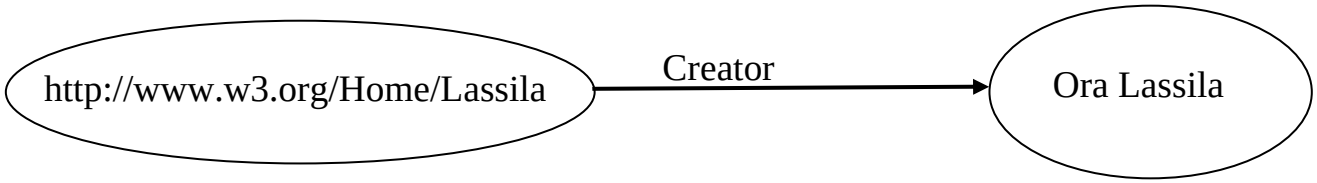
RDF

==> Next step: **separate content from structure!**

AIFB

# Encoding of Knowledge: Example

"The Creator of the Resource "http://www.w3.org/Home/Lassila" is Ora Lassila




RDF

Endless encoding possibilities in XML:

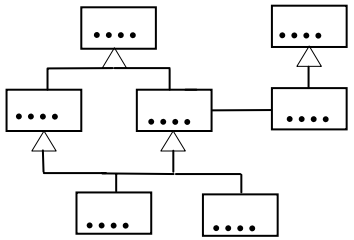
```
<Creator>
    <uri> http://www.w3.org/Home/Lassila </uri>
    <name>Ora Lassila</name>
</Creator>
```

```
<Document uri="http://www.w3.org/Home/Lassila"
    <Creator>Ora Lassila</Creator>
</Document>
```

```
<Document uri="http://www.w3.org/Home/Lassila" Creator="Ora Lassila"/>
```

AIFB

# Point to Point Communication for Machine-Understandable Data



Conceptual Domainmodel
(Objects and Relations)

Person is_a Mammal
Student is_a Person
---

Translation Step

<xsd:schema xmlns:xsd="http://...">
<xsd:annotation> A-Schema

</xsd:...
</xsd:schema>

DTD or XML Schema

Deployment

XML-based Communication
using DTD A

XML-
Parser

Parse Tree

Sender using DTD A

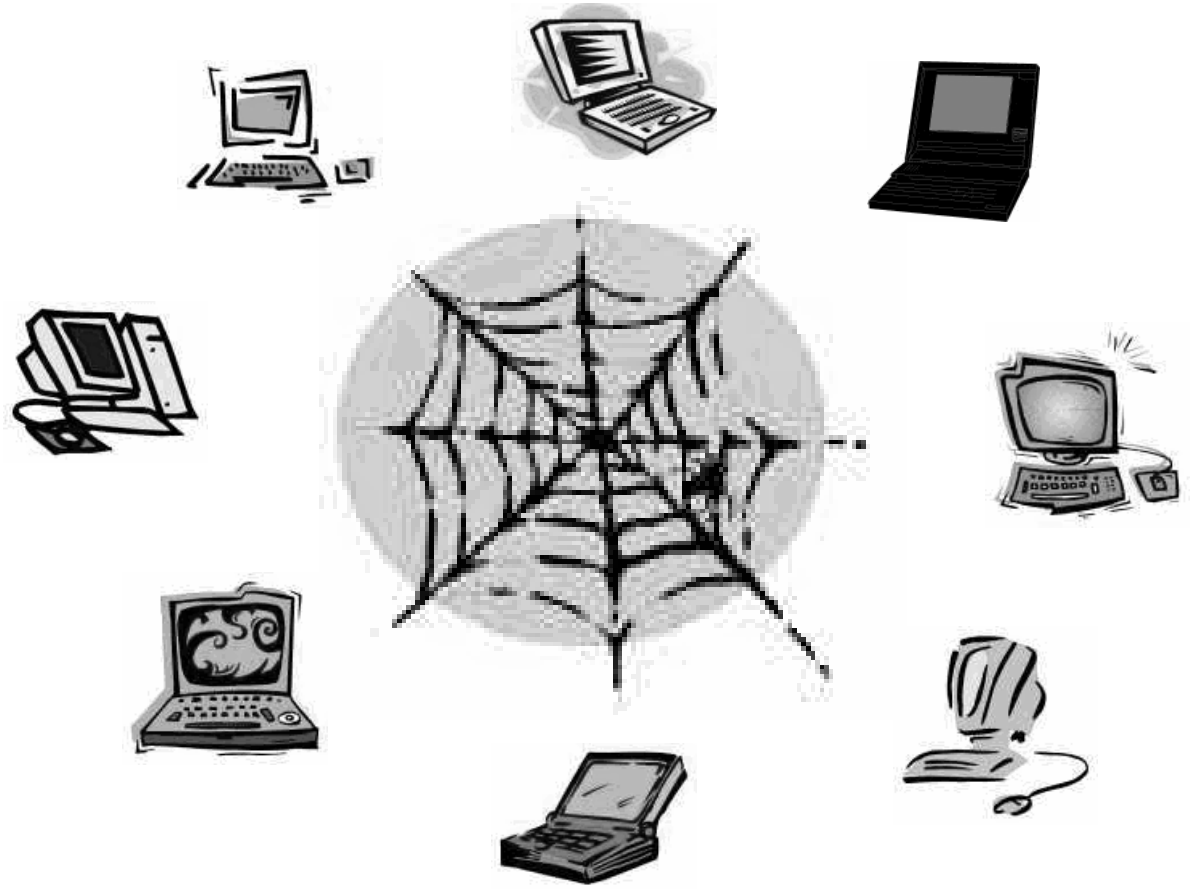
Recipient using DTD A

Common Semantics

AIFB

RDF

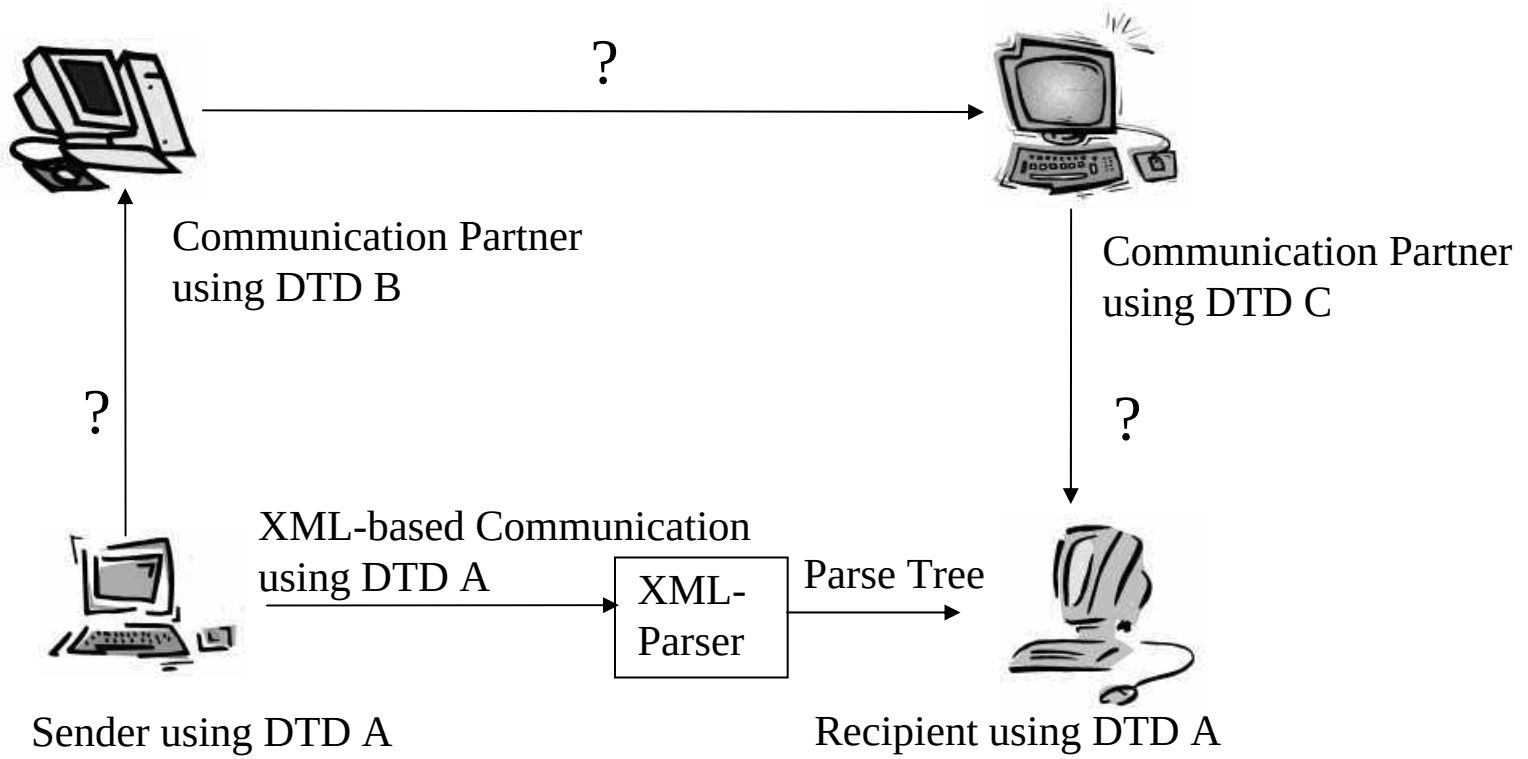
# Many previously unknown Communication Partners



AIFB

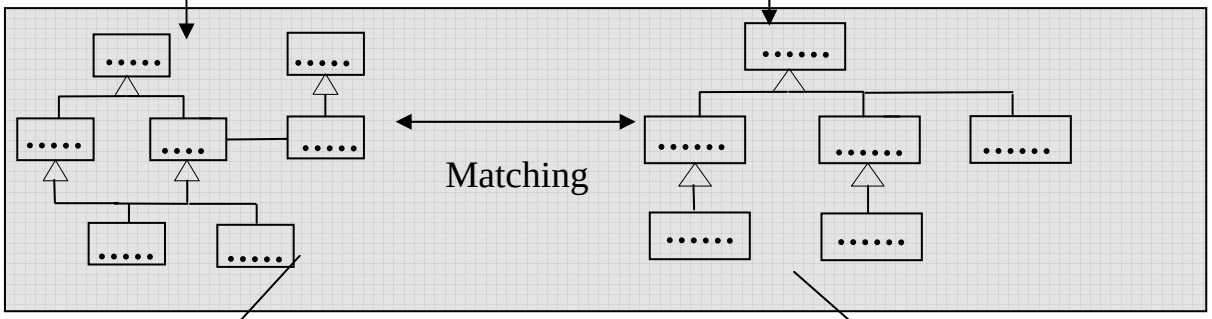
RDF

# New Partner don't understand each other



AIFB

# Merging Steps between Models

| Steps | DTD A | DTD B |
|---|---|---|
| Reengineering of the conceptual model | <xsd:schema xmlns:xsd="http://..."> <xsd:annotation>A-Schema </xsd:...> </xsd:schema> | <xsd:schema xmlns:xsd="http://..."> <xsd:annotation>B-Schema </xsd:...> </xsd:schema> |
| Matching |   | |
| XML Document Translation Generation (e.g. in XSLT) | <xsl:stylesheet version="1.0" xmlns:xsl="http://....Transform" <xsl:template match="/"> .... </xsl:template> </xsl:stylesheet> | <xsl:stylesheet version="1.0" xmlns:xsl="http://....Transform" <xsl:template match="/"> .... </xsl:template> </xsl:stylesheet> |
| XML Document Translation from DTD A to DTD B (and B to A) | | |

RDF

AIFB

# Merging/Aligning Models

- Reengineering step is costly and unnecessary, when a conceptual language is is use
- Generation document translation procedures is again complicated and unnecessary

==> use a level on top of XML

- What are Requirements for such a level?

RDF

# Postulates: Fundamental Requirements for KR on the Web


1.) Knowledge on the Web is distributed
(link Knowledge on the Web)

2.) Knowledge on the Web is biased - there is no universal truth
it must be possible to dispute statements

3.) Many different user communities:
Extensibility and Simplicity

==> Resource Description Framework (RDF)

AIFB

# Introduction to RDF

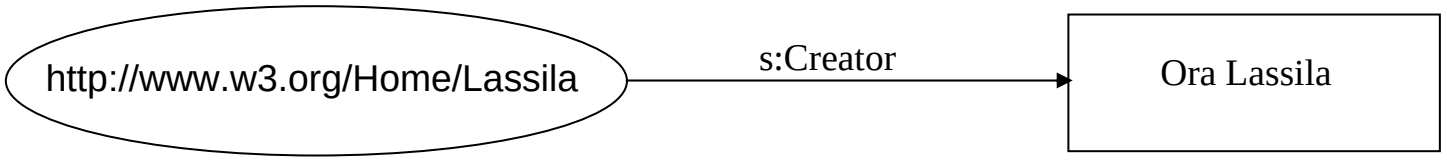
- RDF (Resource Description Framework)
  - Beyond Machine readable to ***Machine understandable***
- RDF unites a wide variety of stakeholders:
  - Digital librarians, content-raters, privacy advocates, B2B industries, AI...
  - Significant (but less than XML) industrial momentum, lead by W3C
- RDF consists of two parts
  - RDF Model (a set of triples)
  - RDF Syntax (different XML serialization syntaxes)
- RDF Schema for definition of Vocabularies (simple Ontologies) for RDF (and in RDF)

# RDF Data Model

- **Resources**
  - A resource is a thing you talk about (can reference)
  - Resources have URI's
  - RDF definitions are itself Resources (linkage)
- **Properties**
  - slots, defines relationship to other resources or atomic values
- **Statements**
  - "Resource has Property with Value"
  - (Values can be resources or atomic XML data)
- **Similar to Frame Systems**

# A simple Example

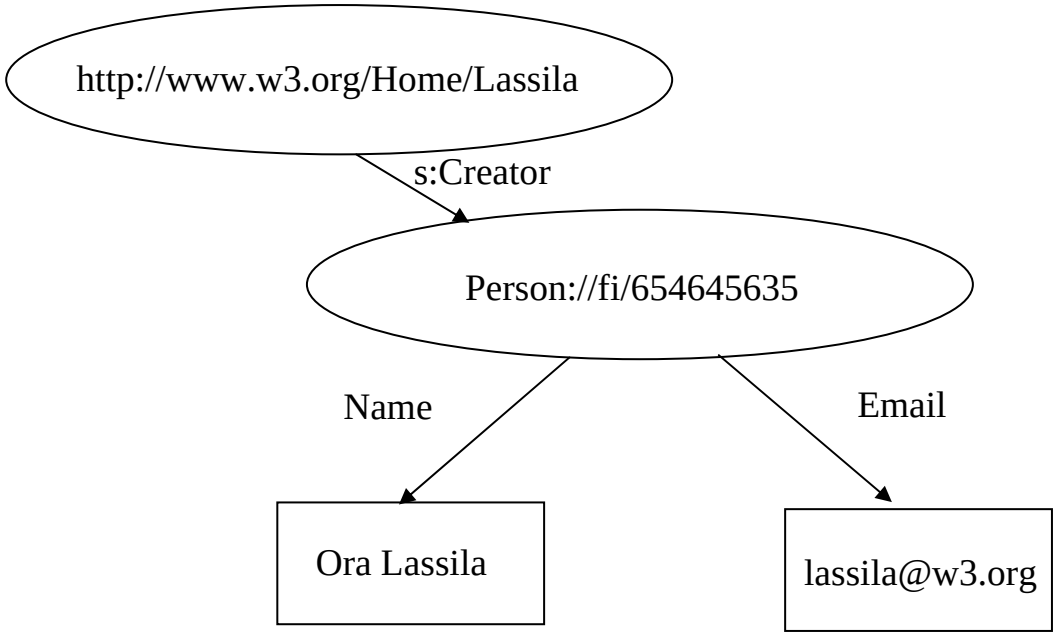
- **Statement**
  - "Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila"
- **Structure**
  - Resource (subject) http://www.w3.org/Home/Lassila
  - Property (predicate) http://www.schema.org/#Creator
  - Value (object) "Ora Lassila"
- **Directed graph**



AIFB

# Another Example

- To add properties to Creator, point through a intermediate Resource.



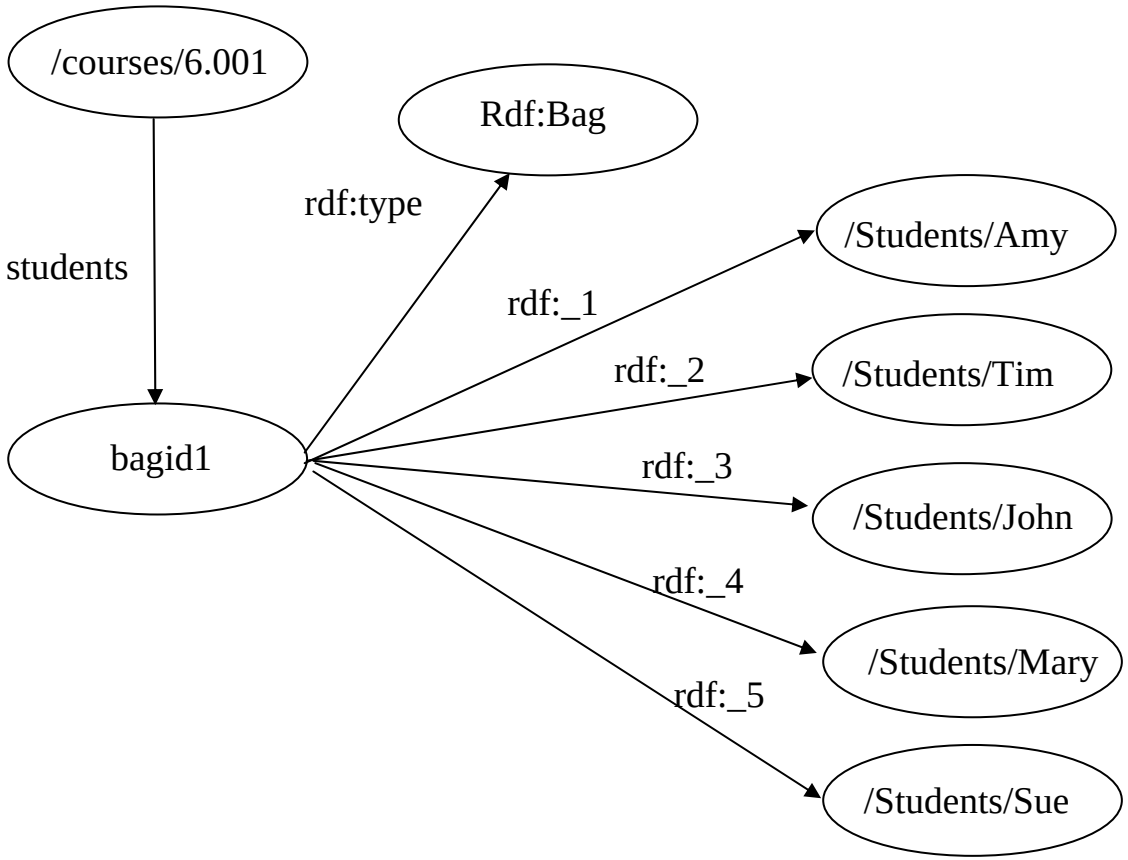

RDF

# Collection Containers

- Multiple occurrences of the same PropertyType doesn't establish a relation between the values
  - The Millers own a boat, a bike, and a TV set
  - The Millers need (a car or a truck)
  - (Sarah and Bob) bought a new car
- RDF defines three special Resources:
  - **Bag** unordered values rdf:Bag
  - **Sequence** ordered values rdf:Seq
  - **Alternative** single value rdf:Alt
    - **Core RDF does not enforce 'set' semantics amongst values**

# Example: **Bag**

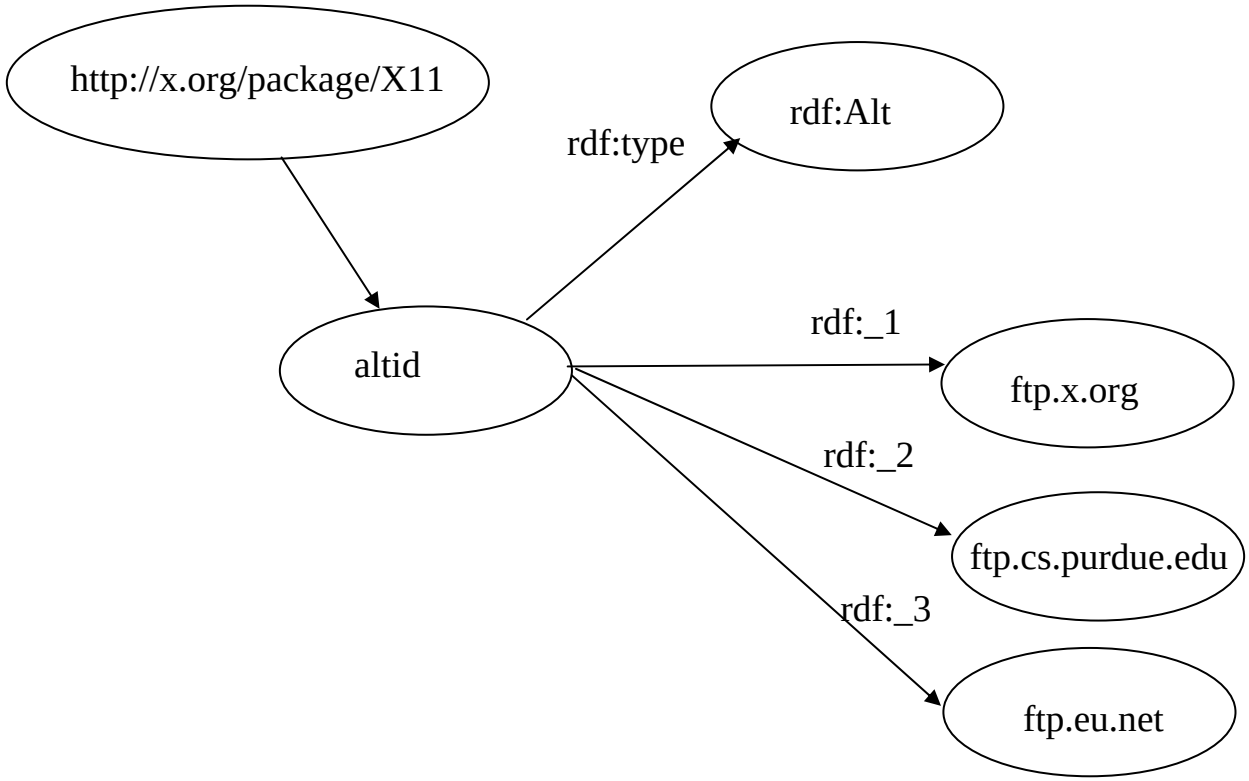
- *The students in course 6.001 are Amy, Tim, John, Mary, and Sue*



AIFB

# Example: **Alternative**

- *The source code for X11 may be found at ftp.x.org, ftp.cs.purdue.edu, or ftp.eu.net*




RDF

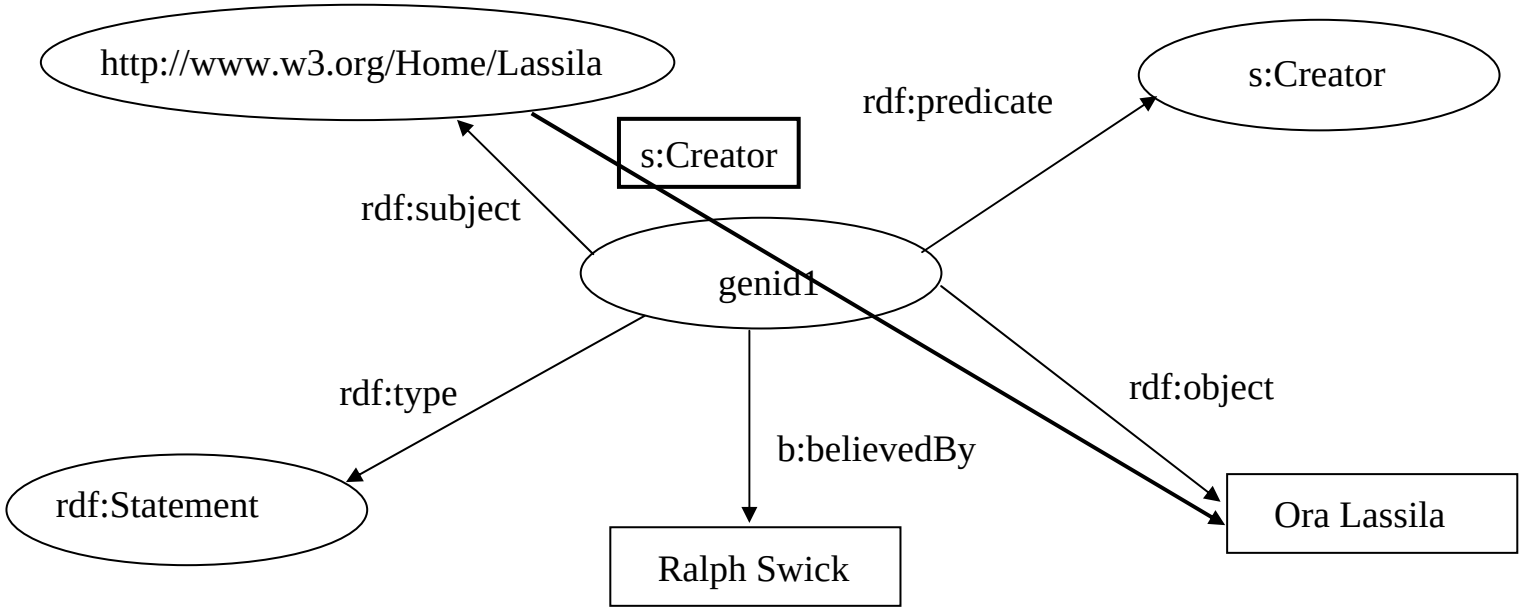
# Statements about Statements (Requirement 2: Dispute Statements)

- Making statements about *statements* requires a process for transforming them into Resources
  - **subject** the original referent
  - **predicate** the original property type
  - **object** the original value
  - **type** rdf:Statement

AIFB

# Example: **Reification**

- *Ralph Swick believes that*
  - *the creator of the resource http://www.w3.org/Home/Lassila is Ora Lassila*



# A Formal Model of RDF

- RDF itself is mathematically straightforward:
  - Basic Definitions
    - Resources
    - Properties $\subset$ Resources
    - Literals
    - Statements = Resources $\times$ Properties $\times\{$Resources $\cup$ Literals$\}$
  - Typing
    - rdf:type $\in$ Properties
    - {RDF:type, sub, obj} $\in$ Statements $\Rightarrow$ obj $\in$Resources

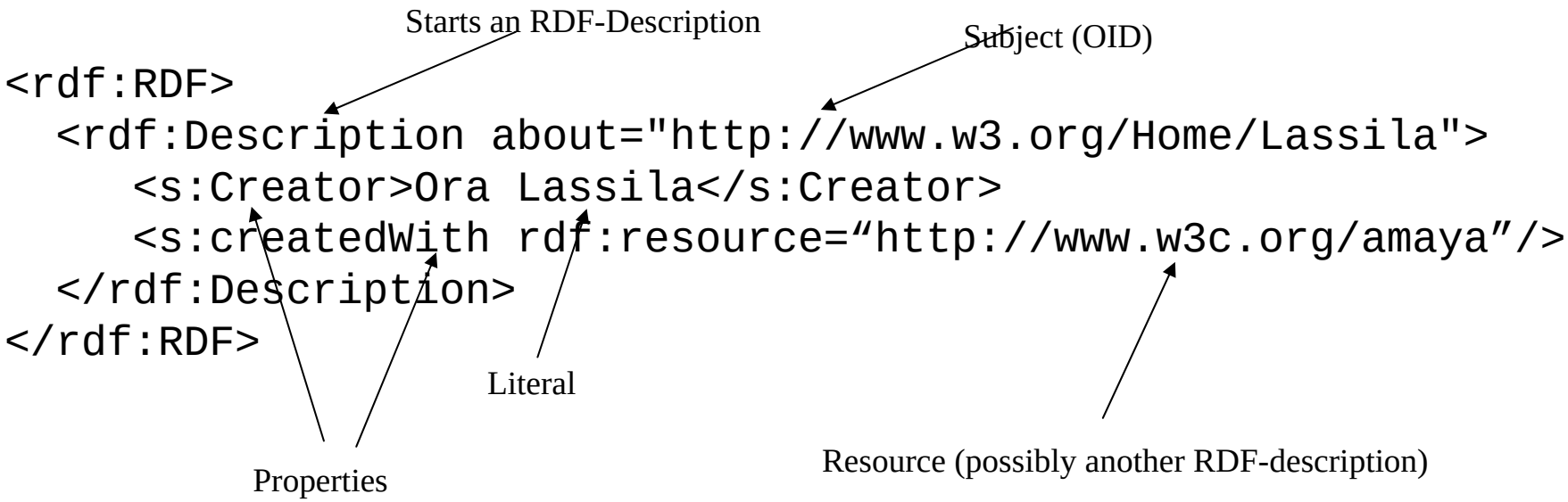
# Formal Model of RDF II

- Reification
  - rdf:Statement $\in$ Resource-Properties
  - $\{$rdf:predicate, rdf:subject, rdf:object $\} \subset$ Properties
  - Reification of a triple $\{$ pred, sub, obj $\}$ of Statements is an element r of Resources representing the reified triple and the elements s1, s2, s3, and s4 of Statements such that
    - s1: {RDF:predicate, r, pred}
    - s2: {RDF:subject, r, subj}
    - s3: {RDF:object, r, obj}
    - s4: {RDF:type, r, [RDF:Statement]}
- Collections
  - $\{$ RDF:Seq, RDF:Bag, and RDF:Alt $\} \subset$ Resources-Properties
  - There is a subset of Properties corresponding to the ordinals (1, 2, 3, ...) called Ord. We refer to
  - elements of Ord as RDF:_1, RDF:_2, RDF:_3, ...

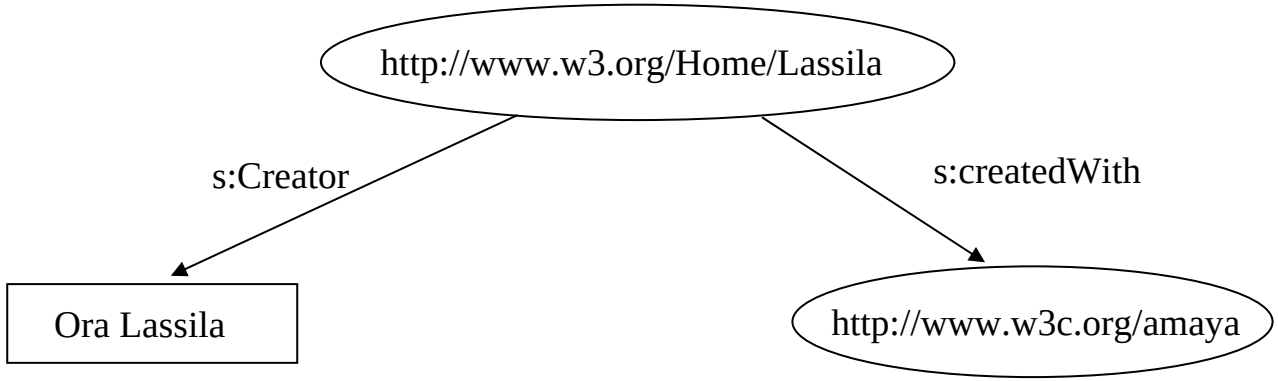
AIFB

# RDF Syntax I

- Datamodel does not enforce particular syntax
- Specification suggests many different syntaxes based on XML
- General form:

# Resulting Graph

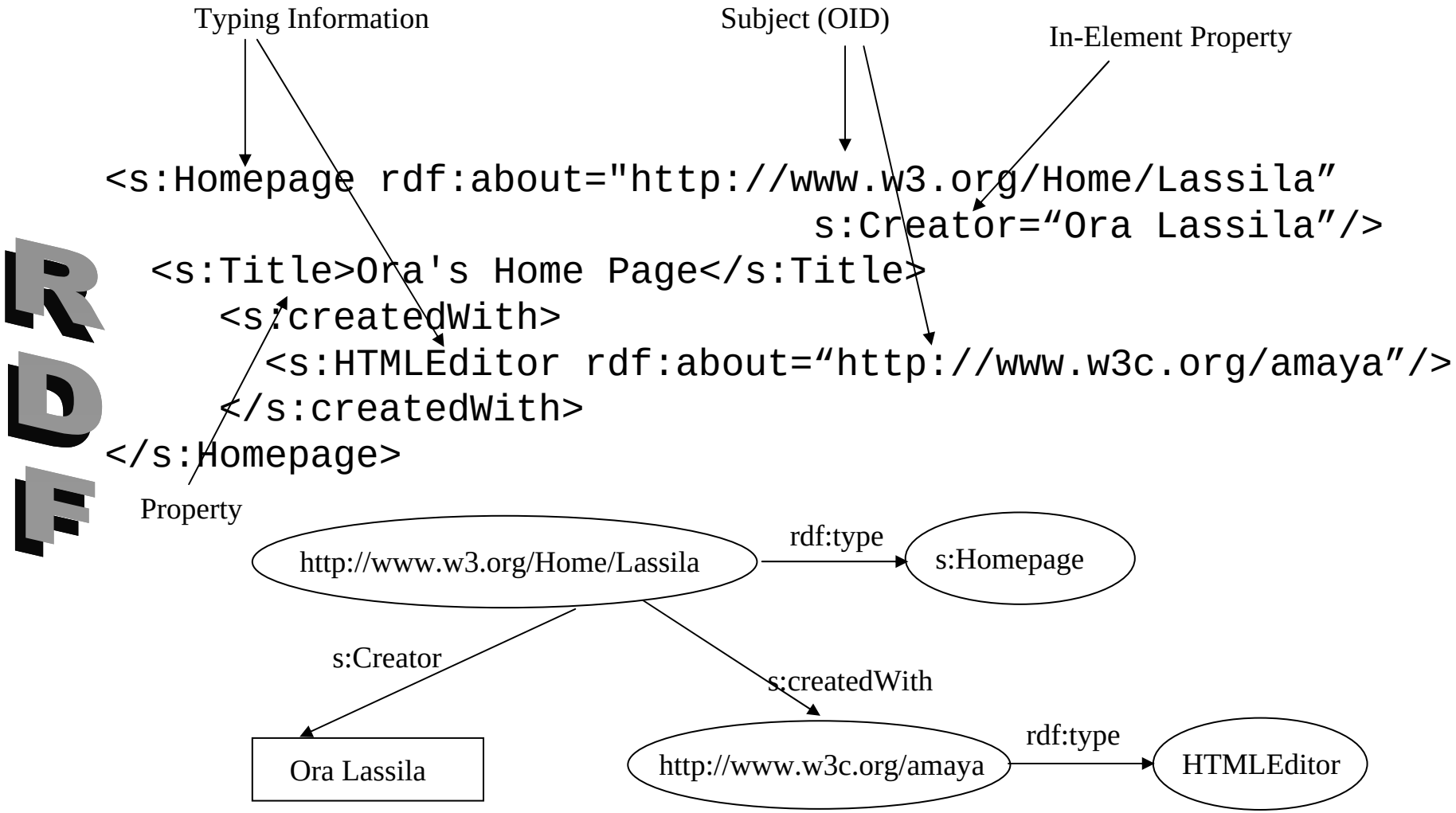



RDF

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
     <s:Creator>Ora Lassila</s:Creator>
     <s:createdWith rdf:resource="http://www.w3c.org/amaya"/>
  </rdf:Description>
</rdf:RDF>
```

AIFB

# RDF Syntax II: Syntactic Varieties



# RDF Schema (RDFS)

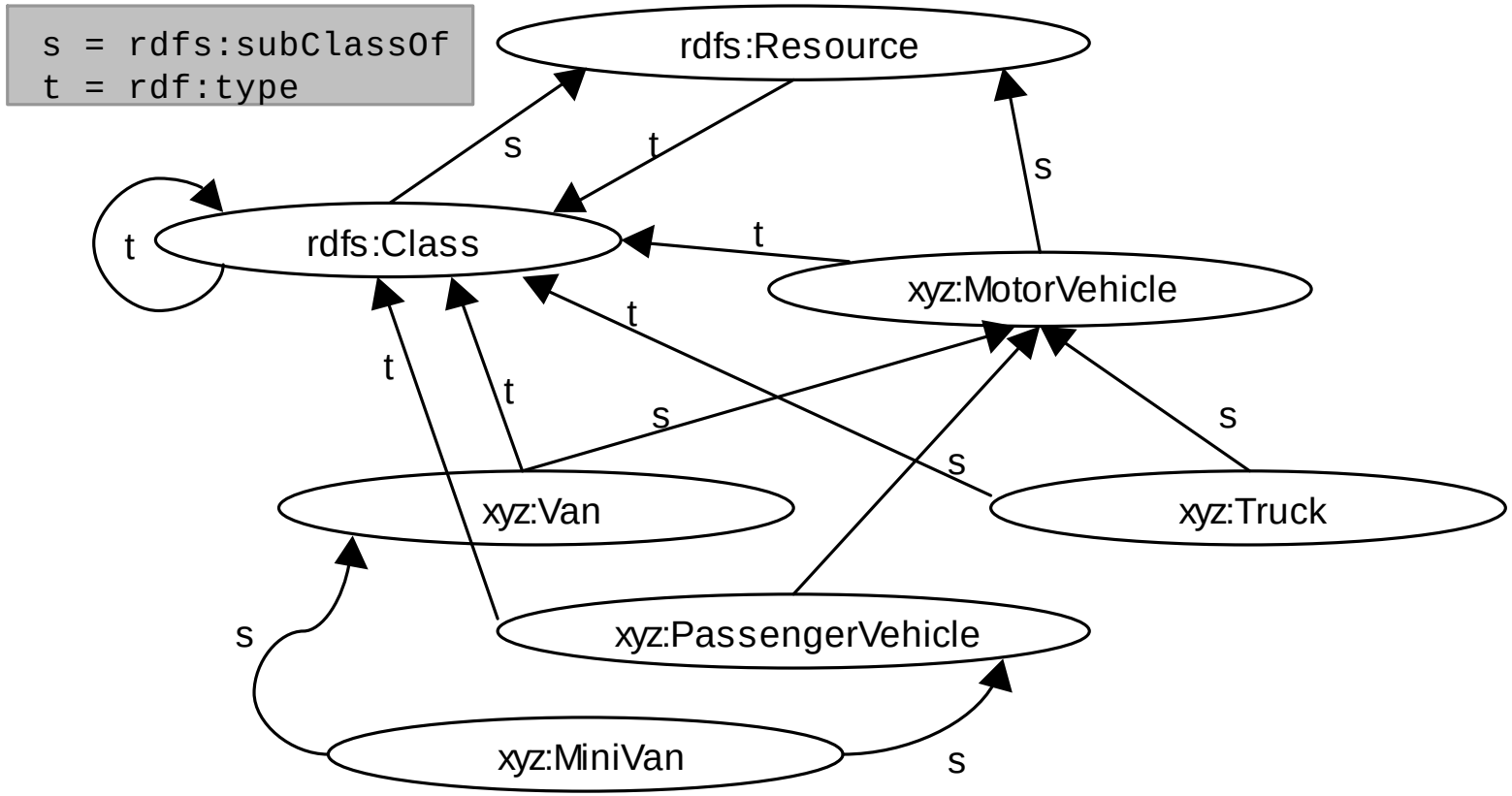
- RDF just defines the datamodel
- Need for definition of vocabularies for the datamodel - an Ontology Language!
- RDF schemas are Web resources (and have URIs) and can be described using RDF

# Most Important Modeling Primitives

- Core Classes
  - Root-Class `rdfs:Resource`
  - MetaClass `rdfs:Class`
  - Literals `rdfs:Literal`
- `rdfs:subclassOf`-property
- Inherited from RDF: properties (slots)
- `rdfs:domain` & `rdfs:range`
- `rdfs:label`, `rdfs:comment`, etc.
- Inherited from RDF: InstanceOf (`rdf:type`)

AIFB

# RDF-Schema: Example




RDF

# Rdfs:subclassOf

```
<rdfs:description about=„xyz:Minivan“>
    <rdfs:subclassOf about=„xyz:Van“/>
</rdfs:description>
<rdfs:description about=„myvan“>
    <rdf:type about=„xyz:MiniVan“/>
</rdfs:description>
```


**Predicate Logic Consequences:**

Forall X: type(X,MiniVan) -> type(X,Van).

Forall X: subclassOf(X,MiniVan) -> subclassOf(X,Van).

# Rdf:property

```
<rdf:description about=„possesses“>
    <rdf:type about=„....property“/>
    <rdfs:domain about=„person“/>
    <rdfs:range about=„vehicle“/>
</rdf:description>
<rdf:description about=„peter“>
    <possesses>petersminivan</possesses>
</rdf:description>
```

**Predicate Logic Consequences:**

Forall X,Y: possesses (X,Y) -> (type(X,person) & type(Y,vehicle)).

AIFB

# More Examples...

RDF

AIFB

# Proxy Resource

```
<rdf:RDF xmlns:rdf="..." xmlns:xx="..." >
 <rdf:Description ID="John_Cowan">
  <xx:email>cowan@ccil.org</xx:email>
  <xx:email>jcowan@haxmail.com</xx:email>
  <xx:homepage
   resource="http://www.ccil.org/~cowan"/>
 </rdf:Description>
</rdf:RDF>
```

# Attributes instead of elements

A property/value pair can be expressed as an attribute instead of a sub-element if:

- The value is a string
- There is only one value for the property

```
<rdf:Description ID="John_Cowan"
  email="cowan@ccil.org">
  <xx:homepage resource="..."/>
</rdf:Description>
```

# Nested descriptions

- When a value is a proxy resource, it may be placed inside the property element

```
<rdf:Description about="...">
 <xx:author>
  <rdf:Description ID="John_Cowan">
   ..
  </rdf:Description>
 </xx:author>
</rdf:Description>
```

# Simplification rules

- People specifying text for arbitrary RDF processors can use any simplification
- Processors of arbitrary RDF therefore must accept all simplifications
- Special-purpose XML formats can be RDF-compliant while disallowing simplifications, requiring them, or exploiting them in specific ways

# Containers

- Proxy resources that hold one or more values (Web resources or strings)
- Bag: a simple collection with no ordering
- Seq: a collection with implicit ordering
- Alt: a set of alternatives (first one preferred)
- Elements are properties named _1, _2, _3, etc.

# Container examples

- Bag: committee members, documents in a folder, checks in a bag
- Seq: book authors (order counts!), chapters in a book, items in an agenda
- Alt: document home and mirrors, mailing-list moderators, translations of a document

# A Bag, the hard way

```
<rdf:Description ID="committee">
 <rdf:type
  resource="http://www.w3.org/1999/02/22-rdf-syntax-
   ns#Bag"/>
 <rdf:_1>Jack Robinson</rdf:_1>
 <rdf:_2>John Doe</rdf:_2>
 <rdf:_3>Richard Roe</rdf:_3>
</rdf:Description>
```

# What is the “rdf:type” property?

- It specifies a class (there may be more than one) to which the resource belongs
- Its value is always a Web resource representing the class
- It can be expressed as a “type” attribute on a Description element
- It can also be implied by using a special element instead of a Description element

# A Bag, the easy way

```
<rdf:Bag ID="committee">
 <rdf:li>Jack Robinson</rdf:li>
 <rdf:li>John Doe</rdf:li>
 <rdf:li>Richard Roe</rdf:li>
</rdf:Bag>
```

- Using an "rdf:Bag" element means the value of "type" is "http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag"

# Kinds of “about” attributes

- “about”: specifies the URL of the Web resource directly
- “aboutEach”: specifies the URL of a container; the properties apply to the individual members of the container
- “aboutEachPrefix”: specifies an URL prefix; the properties apply to all Web resources with that prefix

# Containers vs. multiple values

- A property can appear more than once with different values
- What is true of a container isn't necessarily true of its contents and vice versa
- "aboutEach" lets us get to the contents when we already have a container
- "aboutEachPrefix" in effect manufactures a container based on URLs

# Internationalization

- RDF property names are the same in all locales
- RDF string values take their language from the nearest enclosing “xml:lang” attribute
- Typical values of “xml:lang”: en-US, en-UK, ja-JP, he, i-no-nyn

# Reified statements

- We reify statements so that we can talk about them rather than asserting them
- “Charles Dickens is the author of *Bleak House*” asserts a property of Charles Dickens
- “Jack believes that Charles Dickens is the author of *War and Peace*” asserts a property of Jack, not Charles Dickens

# Reification properties

```
<rdf:Description about="...">
 <xx:creator>Charles Dickens</xx:creator>
</rdf:Description>
```

reifies as:

```
<rdf:Statement>
 <rdf:subject resource="..."/>
 <rdf:predicate resource="...#creator/>
 <rdf:object>Charles Dickens</rdf:object>
</rdf:Statement>
```

# Bags of reified statements

- A Description element can be seen as specifying a Bag of reified statements, one for each property element or attribute
- This Bag can be given a name using the “bagID” attribute
- Referring to a “bagID” attribute from an “aboutEach”-type Description element allows statements about statements

# RDF Schemas

- Describe rules for using RDF properties
- Are expressed in RDF
- Are not to be confused with XML Schemas (eventual DTD replacements)
- **W3C Recommendation since 14th Feb 2004**

# RDF Classes

- Are groups of Web resources
- Have URLs to identify them
- The special class “rdfs:Literal” consists of all possible RDF string values

AIFB

# Property-centric classes

- In typical OO classes,each class specifies completely what properties it has and what their types are
- In RDF classes, each property specifies what classes of subjects and objects it relates
- Therefore, new properties can be added to a class without modifying the class

RDF

# Specifying classes

- To specify a class, create an RDF resource of type rdfs:Class

```
<rdfs:Class id="MyClass">
 <rdfs:label>My Class</rdfs:label>
 <rdfs:comment>John Cowan's demonstration
   Class</rdfs:comment>
</rdfs:Class>
```

# Specifying properties

- To specify a property, create an RDF resource of type rdfs:Property

```
<rdfs:Property id="myProperty">
 <rdfs:comment>John Cowan's demo
     property</rdfs:comment>
 <rdfs:domain resource="#MyClass"/>
 <rdfs:range resource="..#Literal"/>
<rdfs:Property>
```

# Schema URIs

- Ordinary XML namespace URIs are just to guarantee uniqueness: there is no assumption that the URI refers to anything useful (or even refers at all)
- URIs for namespaces used in RDF, though, should refer to an RDF schema document

# Useful properties

- “rdf:type” relates any resource to its class
- “rdfs:subClassOf” relates a subclass to its superclass (multiple inheritance is OK)
- “rdfs:subPropertyOf” relates a subproperty to its superproperty

# Useful properties

- “rdfs:seeAlso” relates a resource to another resource explaining it (use a subproperty to specify the nature of the explanation)
- “rdfs:isDefinedBy” is a subproperty of “rdfs:seeAlso” and relates a resource to its definition, typically an RDF schema

# Useful properties

- “rdfs:domain” specifies the domain of a property (the classes of its subjects); if unknown, anything can be a subject
- “rdfs:range” specifies the range of a property (the single class of its objects); if unknown, anything can be an object

# Useful properties

- “rdf:subject” is the property relating a reified statement to its subject (resource)
- “rdf:predicate” is the property relating a reified statement to its predicate (property)
- “rdf:object” is the property relating a reified statement to its object (value)

# Useful properties

- “rdfs:label” specifies a human-readable name for this Class, Property, or whatever
- “rdfs:comment” specifies human-readable documentation
- Multiple values are useful for specifying multiple languages

# Useful classes

- “rdfs:Resource” is the class of all resources
- “rdfs:Literal” is the class of all strings
- “rdfs:Class” is the class of all classes
- “rdfs:Property” is the class of all properties
- “rdf:Statement” is the class of all asserted RDF statements

# Useful classes

- “rdfs:Container” is the superclass of all container classes
- “rdf:Bag”, “rdf:Seq”, “rdf:Alt” are the classes of Bags, Seqs, and Alts
- (Any other class that is a subclass of “rdfs:Container” can be used in RDF syntax in place of a standard container)

# Dublin Core

- A set of fifteen basic properties for describing generalized Web resources
- The “obvious” mapping of Dublin Core properties into RDF properties has not yet been approved by the Dublin Core initiative, but is generally a good example

# Dublin Core

- “Title”: the name given to the resource
- “Creator”: the person or organization primarily responsible for the resource
- “Subject”: what the resource is about
- “Description”: a description of the content

AIFB

# Dublin Core

- “Publisher”: the person or organization responsible for making the resource available
- “Contributor”: someone who has provided content to the resource other than the creator
- “Date”: date of creation or publication

RDF

AIFB

# Dublin Core

- "Type": type of resource, such as home page, technical report, novel, photograph…
- "Format": data format of the resource
- "Identifier": URL, ISBN number, …
- "Source": another resource that this resource is derived from

RDF

AIFB

# Dublin Core

- “Language”: the language of the content
- “Relation”: another resource and its relationship to this one
- “Coverage”: the portion of time or space described by this resource (atlases, histories, etc.)

RDF

# Dublin Core

- “Rights”: the intellectual property rights adhering to this resource, or a pointer to them

# Where to look next

- RDF Concepts and Abstract Syntax
  http://www.w3.org/RDF/

# Example: RDF Schema in RDF-Schema

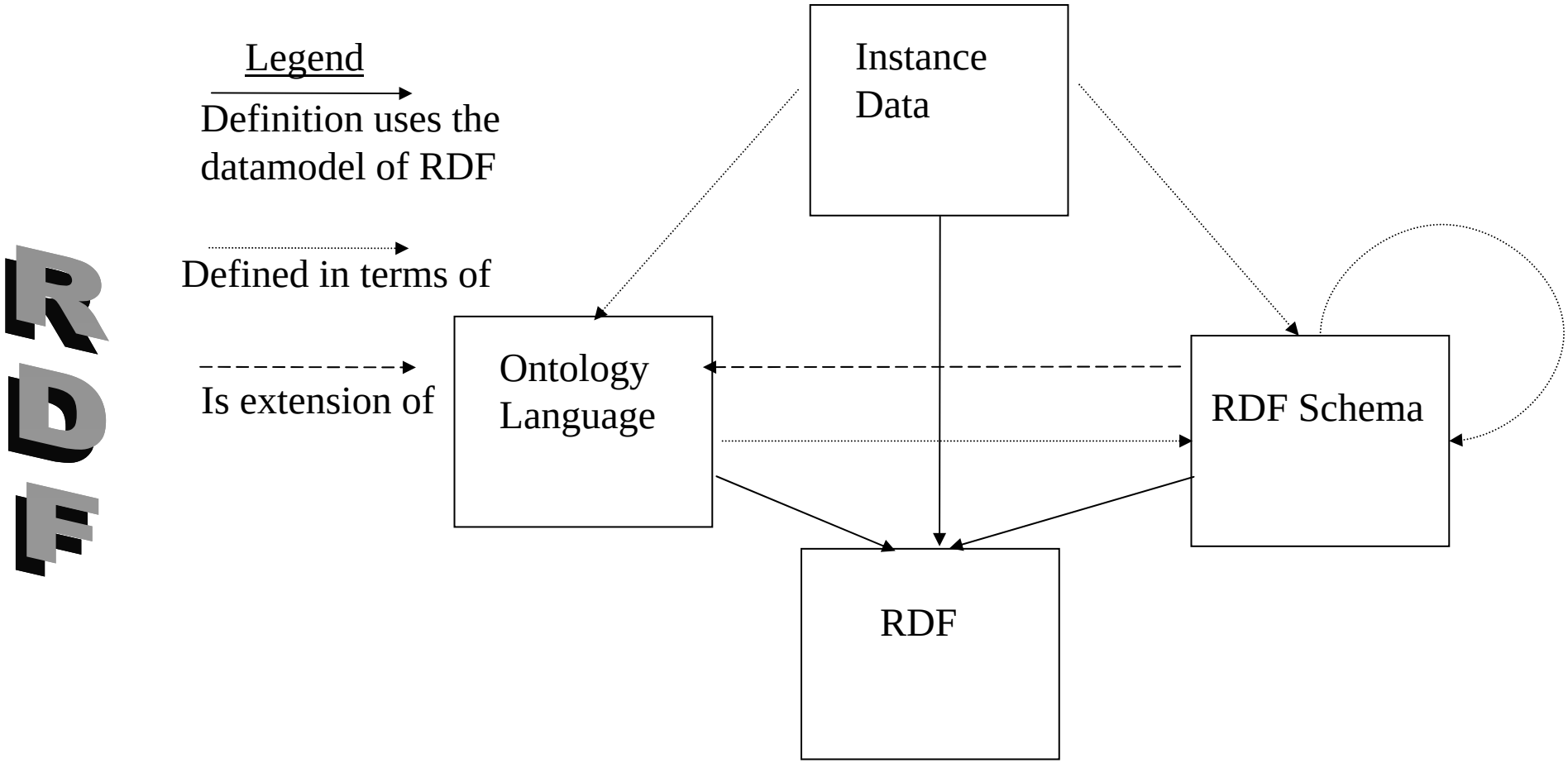
- Namespace-URL: http://www.w3.org/2000/01/rdf-schema#

```
<rdfs:Class rdf:ID="Resource">
  <rdfs:label xml:lang="en">Resource</rdfs:label>
  <rdfs:label xml:lang="fr">Ressource</rdfs:label>
  <rdfs:comment>The most general class</rdfs:comment>
</rdfs:Class>
<rdfs:Class rdf:ID="Class">
  <rdfs:label xml:lang="en">Class</rdfs:label>
  <rdfs:label xml:lang="fr">Classe</rdfs:label>
  <rdfs:comment>The concept of Class</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Resource"/>
</rdfs:Class>
<rdf:Property ID="subClassOf">
  <rdfs:label xml:lang="en">subClassOf</rdfs:label>
  <rdfs:label xml:lang="fr">sousClasseDe</rdfs:label>
  <rdfs:comment>Indicates membership of a class</rdfs:comment>
  <rdfs:range rdf:resource="#Class"/>
  <rdfs:domain rdf:resource="#Class"/>
</rdf:Property>
```

# Extensibility of RDF

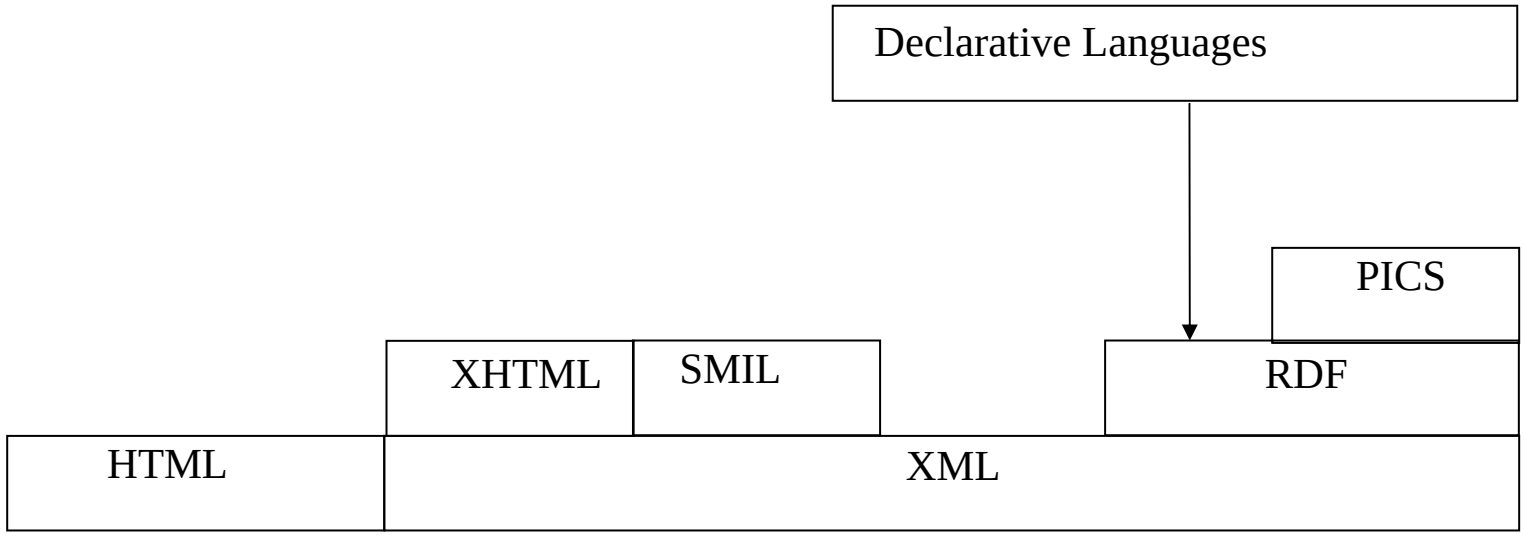
- Define an Ontology of your Language with RDF Schema (like RDF-Schema itself)
- Describe Instance Data using your new Vocabulary
- Advantage: all Languages use the same Data Model (simplifies Interoperability)

# Ontology Languages on Top of RDF: The Principle



# The Semantic Web

- A Web of machine understandable Data, based on declarative languages on top of RDF (all use the same datamodel!)
- Intelligent Agent enabling architecture
- W3C's vision for the Semantic Web Architecture:



# Formal Models of RDF I

- **Official Semantics: RDF Model Theory (aka 'RDF Semantics')**
  [ → http://www.w3.org/TR/rdf-mt/ ]

*Specification of a precise semantics for RDF (and RDFS), and of corresponding entailment and inference rules which are sanctioned by the semantics.*

- **Other Proposals:**
  - RDFS(FA)
    [ → http://dl-web.man.ac.uk/rdfsfa/ ]
    UML-Like Stratification

# Formal Models of RDF II

- **RDF-MT is based on classical Tarski-style Model Theory**
- **Some Entailment rules for RDFS:**

| | **If E contains:** | **then add:** |
|---|---|---|
| rdf1 | xxx aaa yyy . | aaa rdf:type rdf:Property . |
| rdfs2 | xxx aaa yyy .<br>aaa rdfs:domain zzz . | xxx rdf:type zzz . |
| rdfs3 | xxx aaa uuu .<br>aaa rdfs:range zzz . | uuu rdf:type zzz . |
| rdfs4a | xxx aaa yyy . | xxx rdf:type rdfs:Resource |
| rdfs4b | xxx aaa uuu . | uuu rdf:type rdfs:Resource |
| rdfs5a | aaa rdfs:subPropertyOf bbb .<br>bbb rdfs:subPropertyOf ccc . | aaa rdfs:subPropertyOf ccc |
| | | |

| | If E contains: | then add: |
|---|---|---|
| rdfs5b | xxx rdf:type rdf:Property . | xxx rdfs:subPropertyOf xxx |
| rdfs6 | xxx aaa yyy .<br>aaa rdfs:subPropertyOf bbb . | xxx bbb yyy . |
| rdfs7a | xxx rdf:type rdfs:Class . | xxx rdfs:subClassOf rdfs:Resource . |
| rdfs7b | xxx rdf:type rdfs:Class . | xxx rdfs:subClassOf xxx . |
| rdfs8 | xxx rdfs:subClassOf yyy .<br>yyy rdfs:subClassOf zzz . | xxx rdfs:subClassOf zzz . |
| rdfs9 | xxx rdfs:subClassOf yyy .<br>aaa rdf:type xxx . | aaa rdf:type yyy . |
| rdfs10 | xxx rdf:type rdfs:ContainerMembershipProperty . | xxx rdfs:subPropertyOf rdfs:member . |

# Formal Model of RDF III

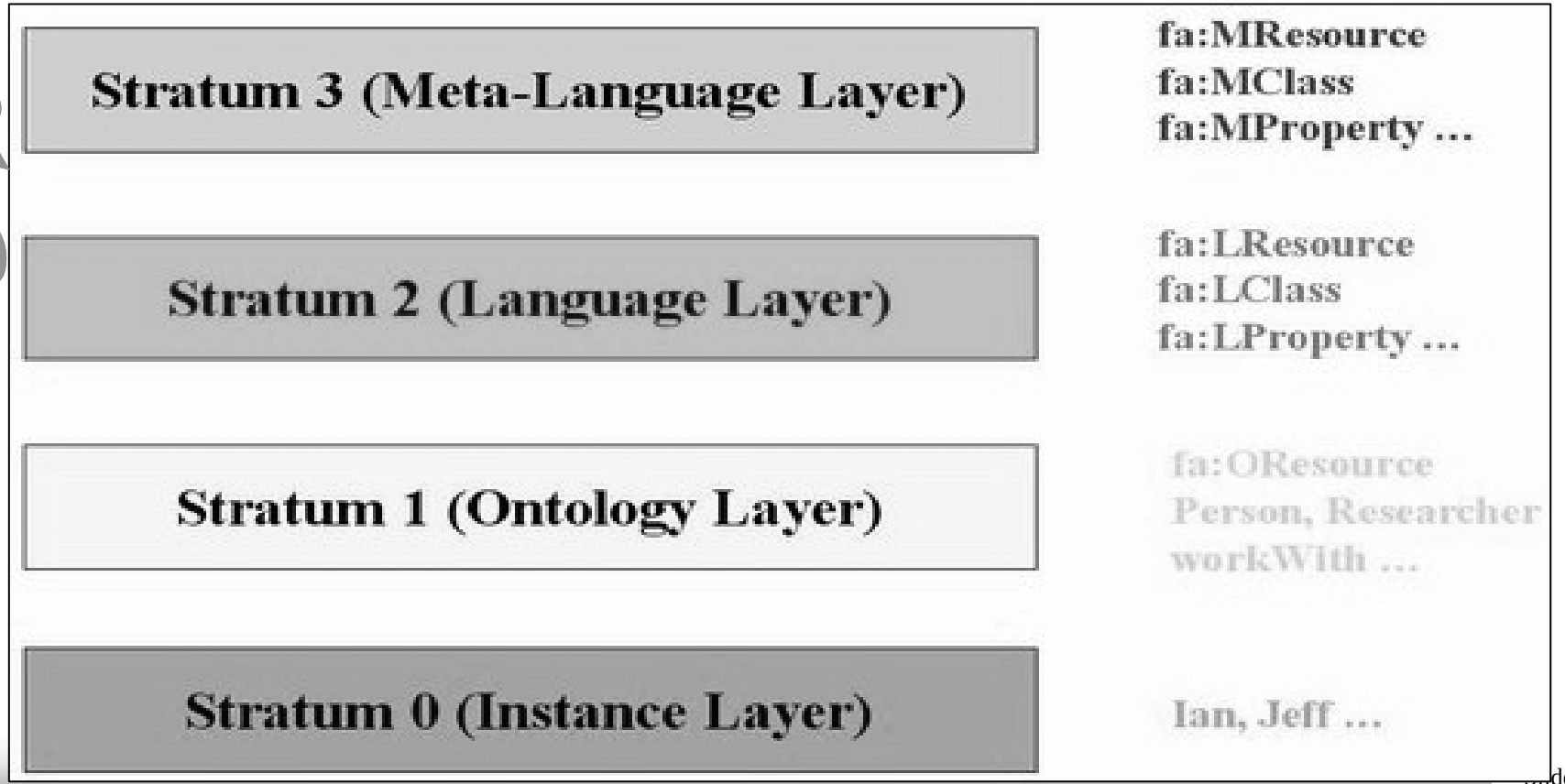
- The entailment process terminates on any finite RDF graph
  → only finitely many possible triples can be formed from a given finite vocabulary.
- Example Graph (Single Triple):
  [foo bar baz ].
- Closure (for mentioned rules only !):

| | | |
|---|---|---|
| 1. | foo bar baz . | Source |
| 2. | foo rdf:type rdfs:Resource . | Rule 4a on (1) |
| 3. | baz rdf:type rdfs:Resource . | Rule 4a on (1) |
| 4. | bar rdf:type rdf:Property . | Rule 1 on (1) |
| 5. | rdf:type rdf:type rdf:Property . | Rule 1 on (4) |
| 6. | rdf:type rdfs:subPropertyOf rdf:type. | Rule 5b on (5) |
| 7. | bar rdfs:subPropertyOf bar. | Rule 5b on (4) |
| 8. | rdfs:subPropertyOf rdf:type rdf:Property | Rule 1 on (6) |

# RDFS(FA) – Stratified Semantics

Similar to UML → Stratification

Avoids cyclic definitions (Class is a Class)



# RDF Applications

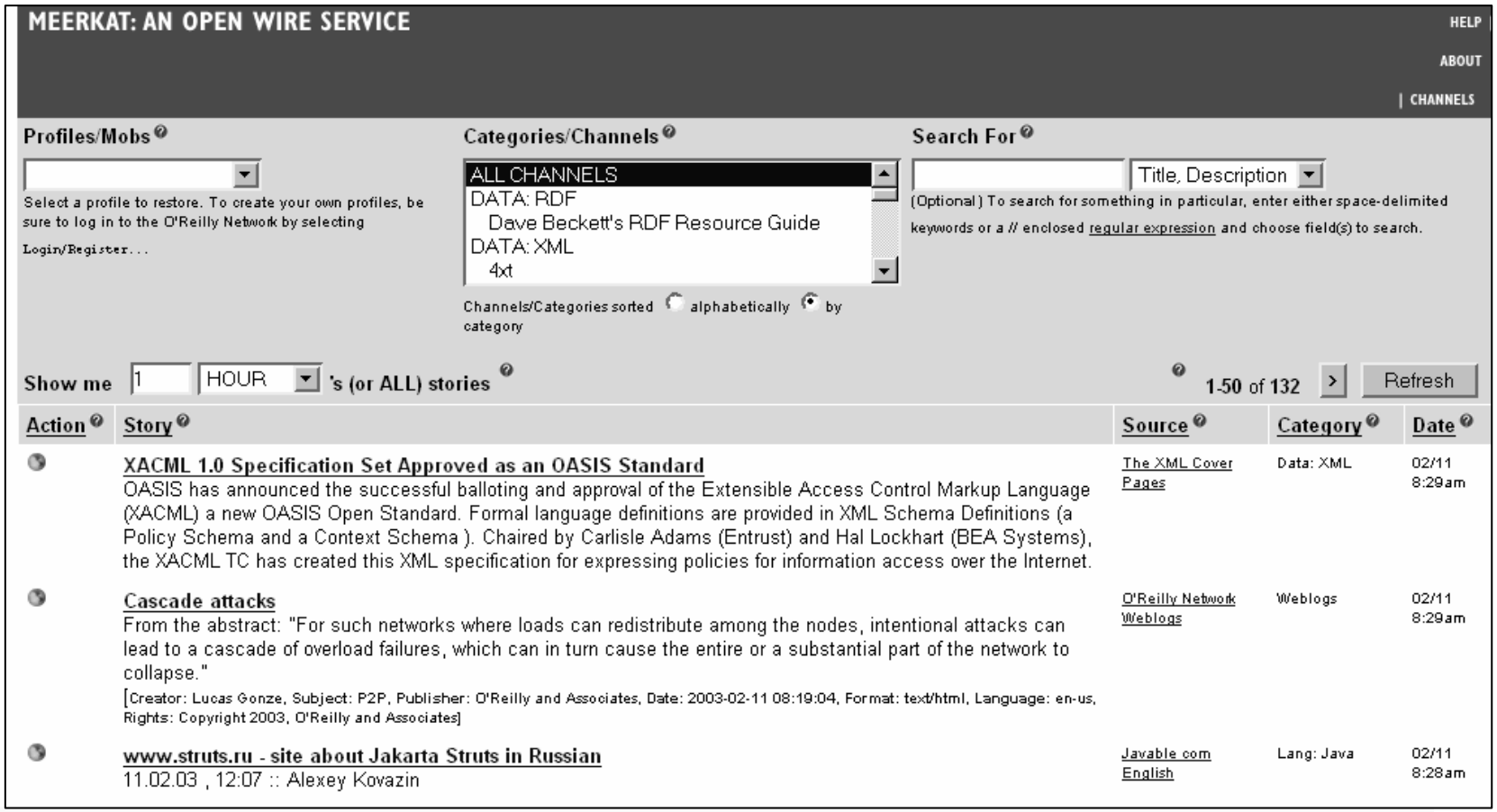
- RDF did not receive strong industry support .. *so far*
- It‘s killer applications are:
  - Carrier Syntax for OWL
    *(W3C Web Ontology Language)*
  - Alternative Syntax for Dublin Core
    *(Metadata Standard)*
  - Carrier Syntax for RSS, OCS
    *(News Syndication)*
  - Alternative Syntax for P3P
    *(Platform for Privacy Preferences)*
  - Data Format used within Mozilla Web-Browser
    [ → http://www.mozilla.org/rdf/doc/ ]

# RSS & OCS

- Web content syndication formats
- Acronym for
  - RSS: „Really Simple Syndication“
  - OCS: „Open Content Syndication
- Data Organization
  - Several Channels publish News
  - Each Channel publishs several (News)Items
- Aggregators
  - Collect News from Providers
  - E.g. Oreilly Meerkat [ → http://www.oreillynet.com/meerkat/]
- Providers
  - Blogs (Persobal Web logs)
  - News Companies (cnn.com etc.)
  - ... Your Portal ??? ...

AIFB

# RSS – OReilly Meercat



# RDF Calendaring - RETSINA

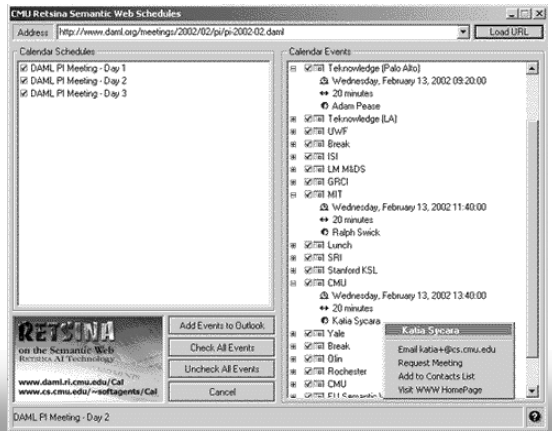
[Payne et al, 2002]

- The Retsina Semantic Web Calendar Agent provides interoperability between
  - RDF based calendar descriptions on the web, and
  - Personal Information Manager (PIM) Systems such as Microsoft's Outlook.

- Online available for download at [ → http://www.daml.ri.cmu.edu/Cal/ ]

- Example:

```
<ical:VEVENT>
- <ical:DTSTART>
  - <ical:DATE-TIME>
      <ical:TZID rdf:resource="#PDT" />
      <rdf:value>20010730T080000</rdf:value>
      <util:hour>08</util:hour>
      <util:minute>00</util:minute>
    </ical:DATE-TIME>
  </ical:DTSTART>
- <ical:DTEND>
  - <ical:DATE-TIME>
      <ical:TZID rdf:resource="#PDT" />
      <rdf:value>20010730T090000</rdf:value>
      <util:hour>09</util:hour>
      <util:minute>00</util:minute>
    </ical:DATE-TIME>
```



# RDF-Resources

- RDF-Editor: KAON and Protege
  - http://kaon.semanticweb.org
  - http://protege.stanford.edu/
- RDF-Parser and APIs/Query Engines
  - http://jena.sourceforge.net/
  - http://openrdf.org/
- RDF Validator
  - http://www.w3.org/RDF/Validator/
- RDF Knowledge Sources
  - DMOZ - Open Directory (largest human created Web-directory) http://www.dmoz.org
- General Information:
  - RDF Interest Mailing list: `www-rdf-interest@w3.org` Archive: `http//lists.w3.org/Archives/Public/www-rdf-interest/`
  - http://www.semanticweb.org