

CmpE 593

Multiagent Systems

Pinar Yolum
pyolum@cmpe.boun.edu.tr

Department of
Computer Engineering
Boğaziçi University

RDF

- Resource Description Framework (RDF)
- Provides a basis for knowledge representation
- Simple language to capture assertions (statements), which help capture knowledge, e.g., about resources on the Web
- Describe the properties and property values
- Allow machines to interpret resource descriptions
- Resources: Web page; person; book

RDF

- Provides a basis for knowledge representation
- Simple language to capture assertions (statements), which help capture knowledge, e.g., about resources
- RDF puts together old KR ideas but uses the Web to enhance their range and avoid some longstanding problems

Why RDF?

- XML
 - Gives us a document tree
 - Doesn't identify the content represented by a document
 - Enables multiple representations for the same content
- RDF expresses the content itself

Basic Concepts

- Describe things that have properties which have values
- Triples of <subject, predicate, object>
- Ex: <Eric Miller, hasTitle, Dr.>
- Need to uniquely identify each so that they can be processed without confusion

URL vs. URI

- Uniform Resource Locator (URL): String of characters that uniquely identifies Web pages
(www.cmpe.boun.edu.tr/~pyolum/index.html)
- Uniform Resource Identifier (URI): Identify resources of any kind; including resources that are not network-accessible
(<http://www.w3.org/People/EM/contact>)
- URI Reference (URIref): Contains an additional fragment identifier
(<http://www.w3.org/People/EM/contact#m4>)

RDF Graph

- Two nodes for the subject and object
- An arc for the predicate
- The resource identified by URI
<http://www.w3.org/People/EM/contact#m4> is of type Person, whose full name is Eric Miller, whose mailbox is em@w3.org and whose personal title is Dr.
- Ovals are URIs; rectangles are literals (only as objects)



Spring 2005— Pinar Yolum

7

RDF Syntax

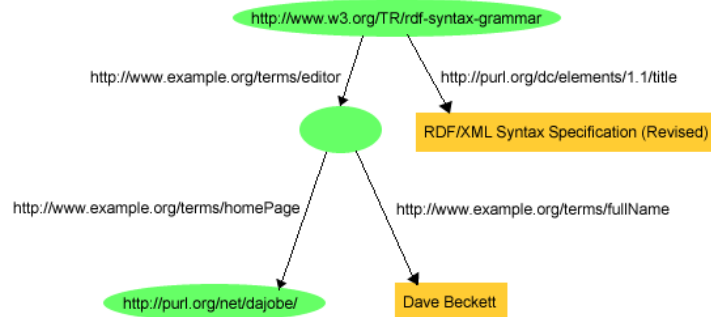
- Equivalent XML-based Syntax (RDF/XML)

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
  <contact:Person
    rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>
```

Spring 2005— Pinar Yolum

8

RDF Blank Nodes (1)



- The document 'http://www.w3.org/TR/rdf-syntax-grammar' has a title 'RDF/XML Syntax Specification (Revised)' and has an editor, the editor has a name 'Dave Beckett' and a home page 'http://purl.org/net/dajobe/'

Spring 2005— Pinar Yolum

9

RDF Blank Nodes (2)

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:exterm="http://example.org/stuff/1.0/">
  <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
    <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
    <exterm:editor rdf:nodeID="abc"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="abc">
    <exterm:fullName>Dave Beckett</exterm:fullName>
    <exterm:homePage
      rdf:resource="http://purl.org/net/dajobe"/>
  </rdf:Description>
</rdf:RDF>
```

Spring 2005— Pinar Yolum

10

RDF Typed Literals

- Typed Literal = String ^^ URIRef for a datatype
 - Ex:
"27"^^<http://www.w3.org/2001/XMLSchema#integer>
- Datatypes
 - Denote a value space (Ex: set of integers for xsd:integer)
 - Uses a lexical space; the characters used to for the datatype
 - Ex: "1", "2", "3" for xsd:integer
 - Ex: "one", "two", "three" for plnar:integer
 - Lexical-to-value mapping: Matches the lexical space to the value space
 - Ex: "1"→1 for xsd:integer
 - Ex: "one" →1 for plnar:integer

RDF Typed Literals (2)

- RDF itself doesn't define any datatypes.
- Datatypes are defined externally; only referred to in RDF
- Common datatypes:
 - xsd:integer
 - xsd:date
 - xsd:string
- Valid RDF?
 - "pumpkin"^^xsd:integer
 - "August 12, 2004"^^xsd:date
 - "2000-03-12"^^xsd.date
 - "2000-03-12+02:00"^^xsd.date

RDF Typed Literals (3)

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
            ns#"
  xmlns:exterms="http://www.example.org/terms/">
  <rdf:Description
    rdf:about="http://www.example.org/index.html">
    <exterms:creation-date rdf:datatype=
      "http://www.w3.org/2001/XMLSchema#date">1999-08-16
    </exterms:creation-date>
  </rdf:Description>
</rdf:RDF>
```

RDF Typed Literals (4)

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd
  "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
            ns#"
  xmlns:exterms="http://www.example.org/terms/">
  <rdf:Description
    rdf:about="http://www.example.org/index.html">
    <exterms:creation-date rdf:datatype= "&xsd:date"> 1999-
08-16
    </exterms:creation-date>
  </rdf:Description>
</rdf:RDF>
```

RDF Typed Literals (5)

- `xml:base` defines a new URI for URIs
- `rdf:type` assigns a type to a description

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd
    "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:exterms="http://www.example.com/terms/"
  xml:base="http://www.example.com/2002/04/products">
  <rdf:Description rdf:ID="item10245">
    <rdf:type rdf:resource="http://www.example.com/terms/Tent"/>
    <exterms:model
      rdf:datatype="&xsd:string">Overnighter</exterms:model>
    <exterms:sleeps rdf:datatype="&xsd:integer">2</exterms:sleeps>
    <exterms:weight
      rdf:datatype="&xsd:decimal">2.4</exterms:weight>
  </rdf:Description>
</rdf:RDF>
```

Spring 2005—Pinar Yolum

15

RDF Containers (1)

- Containers
 - `rdf:Bag` (no order; may include duplicates)
 - `rdf:Seq` (order important; may include duplicates)
 - `rdf:Alt` (list of alternatives)
- Container resource (parent); members (child)
- Membership property names are enumerated as `rdf:_1`, `rdf:_2`, ..., `rdf:_n`
- Semantics vs. Intended Meaning
 - RDF only intends that members of `rdf:Alt` will be specified and interpreted as alternatives.
 - RDF cannot and does not check that they are really alternatives or not.

Spring 2005—Pinar Yolum

16

RDF Containers (2)

- Well-formed rdf:alt container

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
  xmlns:s="http://example.org/packages/vocab#">
  <rdf:Description
    rdf:about="http://example.org/packages/X11">
    <s:DistributionSite>
      <rdf:Alt>
        <rdf:li rdf:resource="ftp://ftp.example.org"/>
        <rdf:li rdf:resource="ftp://ftp1.example.org"/>
        <rdf:li rdf:resource="ftp://ftp2.example.org"/>
      </rdf:Alt>
    </s:DistributionSite>
  </rdf:Description>
</rdf:RDF>
```

Spring 2005—Pinar Yolum

17

RDF Containers (3)

- ill-formed rdf:alt container

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
  xmlns:s="http://example.org/packages/vocab#">
  <rdf:Description
    rdf:about="http://example.org/packages/X11">
    <s:DistributionSite>
      <rdf:Alt>
        <rdf:_2 rdf:resource="ftp://ftp.example.org"/>
        <rdf:_2 rdf:resource="ftp://ftp1.example.org"/>
        <rdf:_5 rdf:resource="ftp://ftp2.example.org"/>
      </rdf:Alt>
    </s:DistributionSite>
  </rdf:Description>
</rdf:RDF>
```

Spring 2005—Pinar Yolum

18

RDF Collections (1)

- Collections
 - Contains members (like Containers)
 - Implies that the listed members are the entire collection
- Predefined collection vocabulary (similar to Lisp lists)
 - `rdf:List` (entire list)
 - `rdf:first` (the first element of a given list)
 - `rdf:rest` (take the first element out; the remaining list)
 - `rdf:nil` (a list without any elements)

RDF Collections (2)

- Collections
 - Contains members (like Containers)
 - Implies that the listed members are the entire collection
- Predefined collection vocabulary (similar to Lisp lists)
 - `rdf:List` (entire list)
 - `rdf:first` (the first element of a given list)
 - `rdf:rest` (take the first element out; the remaining list)
 - `rdf:nil` (a list without any elements)

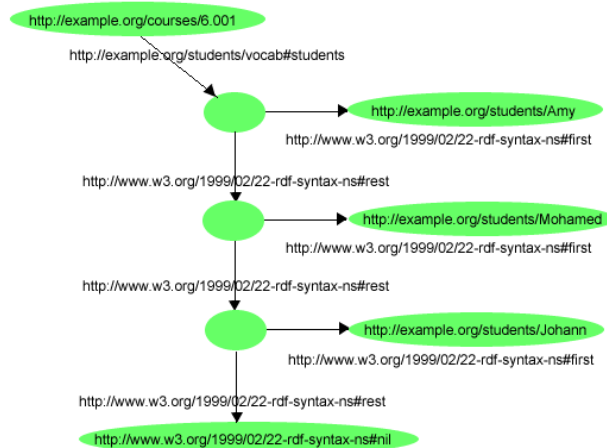
RDF Collections (3)

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://example.org/students/vocab#">
  <rdf:Description rdf:about="http://example.org/courses/6.001">
    <s:students rdf:nodeID="sch1"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="sch1">
    <rdf:first rdf:resource="http://example.org/students/Amy"/>
    <rdf:rest rdf:nodeID="sch2"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="sch2">
    <rdf:first rdf:resource="http://example.org/students/Mohamed"/>
    <rdf:rest rdf:nodeID="sch3"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="sch3">
    <rdf:first rdf:resource="http://example.org/students/Johann"/>
    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
  </rdf:Description>
</rdf:RDF>
```

Spring 2005— Pinar Yolum

21

RDF Collections (4)



Spring 2005— Pinar Yolum

22

RDF Reification (1)

- RDF statements about other RDF statement
 - Meta information about RDF
 - Who wrote this RDF? The date of creation, etc.
- RDF Reification Vocabulary
 - rdf:Statement
 - rdf:subject
 - rdf:predicate
 - rdf:object
- Example:

exproducts:triple12345	rdf:type	rdf:Statement .
exproducts:triple12345	rdf:subject	exproducts:item10245 .
exproducts:triple12345	rdf:predicate	exterms:weight .
exproducts:triple12345	rdf:object	"2.4"^^xsd:decimal .

RDF Reification (2)

```
<rdf:Description rdf:ID="item10245">
  <exterms:weight
    rdf:datatype="&xsd;decimal">2.4</exterms:weight>
</rdf:Description>
<rdf:Statement rdf:about="#triple12345">
  <rdf:subject rdf:resource =
    "http://www.example.com/2002/04/products#item10245"/>
  <rdf:predicate rdf:resource=
    "http://www.example.com/terms/weight"/>
  <rdf:object rdf:datatype="&xsd;decimal">2.4</rdf:object>
  <dc:creator rdf:resource=
    "http://www.example.com/staffid/85740"/>
</rdf:Statement>
```

- Staff 85740 stated that item 10245 weights 2.4 (kg).
- We don't really know the unit; but a unit is necessary!

RDF Structured Values

```
<rdf:Description
  rdf:about="http://www.example.com/2002/04/products#item1
  0245">
  <exterms:weight rdf:parseType="Resource">
    <rdf:value rdf:datatype="&xsd;decimal">2.4</rdf:value>
    <exterms:units
      rdf:resource="http://www.example.org/units/kilograms"/>
  </exterms:weight>
</rdf:Description>
```

- `rdf:parseType="Resource"` creates a blank node
- `exterms:weight` contains two element: value and unit

RDF Schema

- Used to describe the vocabulary for resources
- Examples: `exterms:Tent`; `ex2:Book`
- RDF Schema provides the types that will be used in RDF statements
- RDF talks about objects; RDF schema defines classes for objects
- Prefix- `rdfs`:

RDF Schema Classes (1)

- Classes have a property *rdf:type* and a value *rdfs:Class*
 - `exterms:Tent` `rdf:type` `rdfs:Class`
 - `exterms:outdoorsTent` `rdf:type` `rdfs:Class`
- Resources can be defined based on new classes
 - `exterms:myGreenTent` `rdf:type` `exterms:Tent`
- Classes can be specialized using *rdfs:subClassOf* property
 - `exterms:outdoorsTent` `rdfs:subClassOf` `exterms:Tent`
 - `rdfs:Resource`

Spring 2005— Pinar Yolum

27

RDF Schema Classes (2)

- Example:

```
ex:MotorVehicle      rdf:type      rdfs:Class
ex:PassengerVehicle  rdf:type      rdfs:Class .
ex:Van               rdf:type      rdfs:Class .
ex:Truck             rdf:type      rdfs:Class .
ex:MiniVan          rdf:type      rdfs:Class .
```

```
ex:PassengerVehicle rdfs:subClassOf ex:MotorVehicle
ex:Van              rdfs:subClassOf ex:MotorVehicle .
ex:Truck            rdfs:subClassOf ex:MotorVehicle .
ex:MiniVan          rdfs:subClassOf ex:Van .
ex:MiniVan          rdfs:subClassOf ex:PassengerVehicle .
```

Spring 2005— Pinar Yolum

28

RDF Schema Classes (3)



RDF Schema Classes (4)

```
<rdf:Description rdf:ID="MotorVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
<rdf:Description rdf:ID="PassengerVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
<rdf:Description rdf:ID="Van">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
<rdf:Description rdf:ID="MiniVan">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>
```

RDF Schema Classes (5)

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-
    rdf-syntax-ns#"
    xmlns:ex="http://example.org/schemas/vehicles#"
    xml:base="http://example.org/things">

    <ex:MotorVehicle rdf:ID="companyCar"/>
</rdf:RDF>
```

- Company car will expand into <http://example.org/things/companyCar>
- MotorVehicle will expand into <http://example.org/schemas/vehicles#MotorVehicle>

RDF Schema Properties (1)

- Classes have a properties denoted by *rdf:property*
ex:Person rdf:type rdfs:Class .
ex:author rdf:type rdf:Property .
- *rdfs:range* denotes the range of values for a property
Ex1: ex:author rdfs:range ex:Person .
Ex2: ex:age rdf:type rdf:Property .
ex:age rdfs:range xsd:integer .
xsd:integer rdf:type rdfs:Datatype .

RDF Schema Properties (2)

- *rdfs:range* denotes the values for which a property applies.

Ex: `ex:Book rdf:type rdfs:Class .`
`ex:author rdf:type rdf:Property .`
`ex:author rdfs:domain ex:Book .`

- A property can have 0 or more domain and range.
- Properties can be specialized using *rdfs:subPropertyOf*

Ex: `ex:driver rdf:type rdf:Property .`
`ex:primaryDriver rdf:type rdf:Property .`
`ex:primaryDriver rdfs:subPropertyOf ex:driver`

RDF Schema Example

```
<rdfs:Class rdf:ID="Person"/>
<rdfs:Datatype rdf:about="&xsd;integer"/>
<rdf:Property rdf:ID="registeredTo">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Property>
<rdf:Property rdf:ID="rearSeatLegRoom">
  <rdfs:domain rdf:resource="#PassengerVehicle"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>
<rdf:Property rdf:ID="driver">
  <rdfs:domain rdf:resource="#MotorVehicle"/> </rdf:Property>
<rdf:Property rdf:ID="primaryDriver">
  <rdfs:subPropertyOf rdf:resource="#driver"/>
</rdf:Property>
```

RDF Example

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd
  "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ex="http://example.org/schemas/vehicles#"
  xml:base="http://example.org/things">
  <ex:PassengerVehicle rdf:ID="johnSmithsCar">
    <ex:registeredTo
      rdf:resource="http://www.example.org/staffid/85740"/>
    <ex:rearSeatLegRoom
      rdf:datatype="&xsd:integer">127</ex:rearSeatLegRoom>
    <ex:primaryDriver
      rdf:resource="http://www.example.org/staffid/85740"/>
  </ex:PassengerVehicle>
</rdf:RDF>
```

RDF Application

- RSS 1.0 (RDF Site Summary)
- RDF file that contains the content of a page
 - Description of a news article
 - Its list of main topics
 - URL of the full article
 - Last modification date
- An application that can speak HTTP (e.g., a browser) can request an RSS
- Usage depends on the application
 - Just show the contents
 - Filter articles based on date, topic, etc.

RDF Discussion (1)

- JAVA: Class *book* has an attribute *author* of type *person*
- RDF: There is an *author* property between a *book* and a *person*
- JAVA: If you are talking about a *newspaper*, you need to define a new *author* attribute (Local scope)
- RDF: Define an *author* property once. (Global scope)
- JAVA: You can't talk about an *author* attribute without a class
- RDF: You can if you don't specify a domain

RDF Discussion (2)

- JAVA:
 - Class *sportsarticle* has an attribute *author* of type *male*
 - Class *newsarticle* has an attribute *author* of type *female*
- RDF: Cannot match different domains with ranges
- JAVA is prescriptive
 - Won't allow a male as the author of a news article
- RDF is descriptive; usage is application-dependent
 - Enforce constraints (like JAVA)
 - If the author of a news article is not known infer female
 - Accept the existence of a news article without an author
 - Accept a news article with an *editor* attribute instead

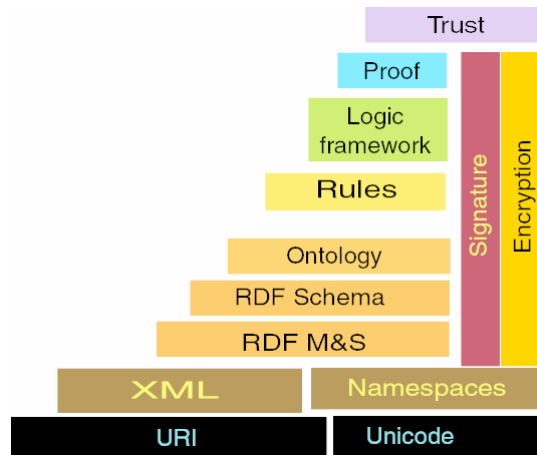
RDF Discussion (3)

- What you cannot say in RDF:
 - Cardinality: Each book has one author.
 - Transitivity: A ex:hasAncestor B, and B ex:hasAncestor C, then A ex:hasAncestor C.
 - Class Equivalence: Two classes (with different URIs) are the same
 - Instance Equivalence: Two objects (with different URIs) are the same
 - Different cardinality requirements for the same property: A homework has one author; but a book may have several.
 - Classes based on others: Two classes are disjoint

RDF Challenges ([Tim Berners-Lee](#))

- Indexing rule files by terms used and translated
- Building data flow and queries dynamically
- Scalable algorithms over fractal space
- Using same rules in forward and backward-chaining contexts
- Incremental, reversible inference - diff, patch and generic synch
- Generating and checking lists of rules used: proofs
- The UI to the Semantic Web
- Secure systems

RDF Big Picture



- Ontology layer (transitivity, cardinality, uniqueness)
- Rule languages for querying (similar to SQL), processing (filtering)
- Usually monotonic logic; used for inference

Spring 2005— Pinar Yolum

41

RDF Challenges

- RDF for policy management
 - Specification of privacy policies (P3P)
 - Formal models of policy verification
 - Merging policies (detecting inconsistencies)
 - Business contracts and rules
- RDF for trust management
 - Anyone can create and put an RDF document on the Web
 - Doesn't mean that the information is correct
 - Trust depends on application purpose
 - Web of trust: Find trustworthy sources using your trusted friends

Spring 2005— Pinar Yolum

42

RDF Web of Trust

(Tim Berners-Lee)

URI
variable



Alan:

- 1) If X is AC rep of Y, X can delegate W3C member access rights in Y.
- 2) Kari is AC rep of Elisa.



Kari:

- 1) If X is employee of Elisa, X has W3C member access rights.
- 2) Tiina is employee of Elisa.



Tiina: I have W3C member access rights
Proof: Alan 1, Alan 2, Kari 1, Kari 2



RDF Web of Trust

(Tim Berners-Lee)

