Knowledge Modelling for Network Management

Resource Description Framework (RDF)
RDF Schema (RDFS)

Adapted from course notes of Simon Dobson, UCD and Declan O'Sullivan, TCD
Examples by Rob Brennan, TCD
Stack Architecture for Semantic Web

User Interface & Applications

Trust

Unifying Logic

Proof

Query: SPARQL

Ontology: OWL

RDF

Rule: RIF

Data interchange: RDF

XML

URI/IRI

Crypto

© Declan O’Sullivan
What is Knowledge Representation?

- Knowledge representation and reasoning (KR) is the field of artificial intelligence (AI) dedicated to representing information about the world in a form that a computer system can utilize to solve complex tasks such as diagnosing a medical condition or having a dialog in a natural language.
How to Represent Knowledge?

There are a number of possible options

• As *objects*, using the well-accepted techniques of object-oriented analysis and design to capture a model
• As *XML*, using the industry-standard structured mark-up language
• As *graphs*, making use of the things we know about graph theory
• As some combination of these

We are looking for: extensibility, ease of use, ease of querying

Which would *you* choose?
We can use the nodes of a graph for facts and the arcs as (binary) relationships between them

- Arches are typically called predicates or relationships in this view
- The set of arcs intersecting a node tells us the information we know about that fact or entity
Graphs as knowledge – 1

How do we use graphs to represent knowledge?

A 'key' from which to hang the different facts

fault1004

stackOverflow

SoftwareError

2 hours

eventDetail

probableCause

fromNetworkElement

manufacturer

Cisco

Lucan Exchange

router

ne01

serviceType

contains

Repair Team3

speciality

basedIn

contains

contains

contains

priority

maximum

Don’t Panic

additionalInfo

dateStamp

21:20::210509

SoftwareError

EsttimetoRepair

manufacturer

Cisco

Lucan Exchange

router

ne01

serviceType

contains

Repair Team3

speciality

basedIn

contains

contains

contains

priority

maximum

Don’t Panic

additionalInfo

dateStamp

21:20::210509
Graphs as knowledge – 2

Things to note

• Scaling – the same graph can represent a load of different knowledge simultaneously
• Agreement – need to know what the various predicates 'mean'
• Structure – you need to know what nodes are related by a predicate
• Plurality – the same relationship may appear several times
• Asymmetry – relationships are inherently directed, which sometimes makes things awkward

...and this can get very tricky
...and this can be difficult to keep straight
For example both NetworkElements and Faults might have estimateTimeToRepair

So a knowledge graph is inherently directed
Two ways to view a graph

As nodes and arcs
- Nodes store facts, edges store relationships between them

As triples
- A three-place relationship of 'subject, predicate, object'
- The node and edge structure is induced by the triples – each triple defines an edge, the different subjects/objects are the population of nodes, one node per individual string

Fault1004 probableCause SoftwareError

The clause form we saw earlier is essentially this triple form but using the order 'predicate subject object'
Graph based approach - 1

The promise

- natural distribution
- easy merging
- naturally extendible
- easy publication and consumption
- easy querying (?)
The Long Road for RDF

- Fits and Starts
  - Original specification 1999
  - Started to have more traction 2004
  - Adopted as baseline in Semantic Web community
  - Mainstream attempt RSS 1.0 mixed results

- Linked Data Movement
  - Tim Berners Lee driven
  - Treat schemas as vocabularies
  - Reuse existing schemas
    - Friend of a Friend, Dublin Core, Open Provenance Model, Creative Commons
Resource Description Framework (RDF)

RDF is an XML vocabulary that defines how to describe knowledge in XML in a standard way

- Standardised by the World Wide Web Consortium (W3C)

RDF isn’t a knowledge standard *per se*: it’s a way of *defining* knowledge standards in a way that maximises the potential for re-use across the web

- A way of defining knowledge graphs
- No standard predicates
- Tool support – editors, parsers, displays, …

Lassila and Swick, Resource Definition Framework model and specification. W3C report, 1999
What does RDF give us?

RDF is intended to address some of the issues we’ve identified in representing knowledge

- Extensible – easy to add new information
- Simple – XML is (pretty) simple to manipulate
- Standard – defined by a standards body
- Scalable – used on the widest (Internet) scale

The way it accomplishes this is what we’ll look at next

Note that this isn’t a complete list of issues – no mention of query complexity or how to actually represent large models.
Basic structure – triples

RDF represent knowledge using a triple structure

- Subject
- Predicate
- Object

Knowledge is built up as a collection of these triples, contained within an XML file

Remember, a triple structure is one way of viewing a graph, so RDF essentially defines a knowledge graph
Representing predicates

How should we represent predicates?

The obvious way is to use some convention for labelling, so we 'know' what the names 'mean'  
- For example, `is_a` to represent 'the subject is a some sort of thing'

Most early systems use this approach  
- Underscores to separate parts of a name  
- A common vocabulary

Any comments?
Drawbacks – informality

The set of predicates is only defined informally
Moreover pre-supposes a human language and some common understanding

- `is_a` makes sense to an English speaker, but isn’t so good for a Swahili-speaker

We need to get broad agreement on what the various symbols mean

- Meaningless to a computer anyway, of course…
- …but we have to make sure we use them consistently
- …and on an Internet scale
Drawbacks – semantic capture

Consider the word 'takes'

Could also mean it like this:

The word 'takes' – and indeed any word – is open to mis-interpretation

- Not precisely defined without a context (sic)
- …and we need something better for machines to work with
Namespaces

The solution is to make sure we have a completely unambiguous statement of what 'takes' means

Now, there are two possible meanings of that too:

- Agree in an unambiguous manner what the word means, for example using a formal system
- Make sure we can distinguish meanings so that we can indicate which one we mean, without necessarily defining the meanings themselves

The first would be great (more later);

.........the second will do us for the moment
What are XML Namespaces?

W3C recommendation (January 1999)
Each XML vocabulary is considered to own a namespace in which all elements (and attributes) are unique
A single document can use elements and attributes from multiple namespaces
• A prefix is declared for each namespace used within a document.
• The namespace is identified using a URI (Uniform Resource Identifier)
An element or attribute can be associated with a namespace by placing the namespace prefix before its name (i.e. 'prefix:name')
• Elements (and attributes) belonging to the default namespace do not require a prefix
Example: XML Namespaces

St. James’s Hospital

```xml
<!ELEMENT Patient (Name, DOB)>
<!ELEMENT Name (First, Last)>
<!ELEMENT First (#PCDATA)>
<!ELEMENT Last (#PCDATA)>
<!ELEMENT DOB (#PCDATA)>
```

Airport Pharmacy

```xml
<!ELEMENT Drug ((Name|Substance), Code)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Substance (#PCDATA)>
<!ELEMENT Code (#PCDATA)>
```

```xml
<?xml version='1.0'?>

<Accident Report xmlns:sjh="http://hospital/sjh"
xmlns:dub=http://airport/dub >

<sjh:Patient>
<sjh:Name>
<sjh:First>Mike</sjh:First>
<sjh:Last>Murphy</sjh:Last>
</sjh:Name>
<sjh:DOB>12/12/1950</sjh:DOB>
</sjh:Patient>

<dub:Drug>
<dub:Name>Nurofen</dub:Name>
<dub:Code>IE-975-2</dub:Code>
</dub:Drug>

[...]
</Accident Report>
```
Namespaces and URLs

Namespaces use URLs, and URLs can be made unique

- If I want to define a new structure I just define a namespace and assign it a unique URL
- If I own a domain I can give it any name I want under my domain name, secure in the knowledge that no-one else will (should!) use it

So a set of predicates, subjects, objects defined using a namespace can be uniquely differentiated from any other set of predicates across the entire web

- Cheap, decentralised model

Common sets of predicates may be given well-known names and URLs
Basic RDF structure

RDF data can live inside other XML documents
Identified by a top-level element from the well-known RDF namespace

This is the standard RDF namespace

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  ...
</rdf:RDF>
```

Within the RDF element we can define triples

- Co-exists well with the rest of XML – if you don’t understand RDF you can omit everything within this element
- Readable enough for people to build by hand

In addition to what follows you may want to check out Manola and Miller, RDF Primer. W3C draft technical note, 2002.
A simple fact in RDF

```xml
<?xml version="1.0"?>
  xmlns:ff=http://www.fame.ie/ontologies/fame-faults#
  xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  <rdf:Description rdf:about="http://www.fame.ie/ontologies/fame-faults#fault1004">
    <ff:additionalInfo>OK panic now</ff:additionalInfo>
  </rdf:Description>
</rdf:RDF>
```

The RDF namespace

The namespace for the predicates we want to use

The RDF namespace

A standard RDF element meaning 'the subject of the triples contained in this element'

A triple whose predicate is taken from the `ff` namespace and whose object is the given text

**Subject:**  http://www.fame.ie/ontologies/fame-faults#fault1004

**Predicate:**  http://www.fame.ie/ontologies/fame-faults#additionalInfo

**Object:**  OK panic now

See how the predicate gets created from the namespace?
A cluster of facts

Given a common subject we can build a cluster of facts using nested predicate elements

```xml
<?xml version="1.0"?>
 xmlns:ff="http://www.fame.ie/ontologies/fame-faults#"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://www.fame.ie/ontologies/fame-faults#fault1004">
    <ff:additionalInfo>OK panic now</ff:additionalInfo>
    <ff:dateStamp>2009-05-21T19:42:59Z</ff:dateStamp>
    <ff:priority>maximum</ff:priority>
  </rdf:Description>
</rdf:RDF>
```

Each of these gives rise to a triple with the same subject (inherited from the containing Description element)
An alternative syntax

There’s an alternative syntax that compresses all this information into a single element

- Good, because a browser will typically ignore the elements it doesn’t understand, and this way there’s no spurious element content to mess up the display
- Bad, because it’s a bit too compact and less flexible than the larger form

```xml
<?xml version="1.0"?>
xmlns:ff="http://www.fame.ie/ontologies/fame-faults#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://www.fame.ie/ontologies/fame-faults#fault1004"
ff:additionalInfo="OK panic no"
ff:dateStamp="2009-05-21T19:42:59Z"
ff:priority="maximum">
  </rdf:Description>
</rdf:RDF>
```

Same predicates, but as attributes rather than an nested elements
Suppose we want to talk about the associated Network Element 01 and capture its information too

- Not information about Fault 1004, so shouldn’t be held as child elements of the Description element

The solution is to define another cluster of facts and relate this *cluster* to the fault1004 node
Structured values – 2

These predicates are 'about' the subject that immediately contains them.
Alternative Representation

The **about** attribute is used for subjects -- the left-hand side of a triple and the **resource** attribute is used for objects (targets) -- the right-hand side of the triple - a sort of pointer.

```
<rdf:Description rdf:about="http://www.fame.ie/ontologies/fame-faults#fault1004">
  <ff:additionalInfo>OK panic now</ff:additionalInfo>
  <ff:dateStamp>2009-05-21T19:42:59Z</ff:dateStamp>
  <ff:priority> maximum</ff:priority>

  <ff:fromNetworkElement rdf:resource="#ne01"></ff:fromNetworkElement>
</rdf:Description>

<rdf:Description rdf:about="http://www.fame.ie/ontologies/fame-faults#ne01">
  <ff:iD>01</ff:iD>
  <ff:manufacturer>Cisco</ff:manufacturer>
  <ff:serviceType>router</ff:serviceType>
</rdf:Description>
```
Synthetic structured values

Sometimes the structured value doesn’t have an obvious identity to put in an about attribute

If this is the case, RDF will synthesise one

```xml
<s:member>
  <rdf:Description>
    <s:surname>O’Sullivan</s:surname>
    <s:firstname>Declan</s:firstname>
  ...
  </rdf:RDF>
</s:member>
```

Description doesn’t have an about attribute, so RDF synthesises a unique one for use in the graph
# Relationship to Relational Data?

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Author</th>
<th>Medium</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>As You Like It</td>
<td>Shakespeare</td>
<td>Play</td>
<td>1599</td>
</tr>
<tr>
<td>2</td>
<td>Hamlet</td>
<td>Shakespeare</td>
<td>Play</td>
<td>1604</td>
</tr>
<tr>
<td>3</td>
<td>Hero and Leander</td>
<td>Marlowe</td>
<td>Poem</td>
<td>1593</td>
</tr>
</tbody>
</table>

![RDF Diagram](image-url)
Summary – basic RDF

RDF is a web standard that lets us build knowledge graphs using XML files

- Use URLs for subjects, predicates, objects with namespaces to ensure uniqueness
- Graph built from triples, but using a nested notation
- Multiple facts can be specified with minimal repetition
- Fairly straightforward for humans to write by hand