Semantic Web and Formal Methods

Jin Song DONG

Computer Science Department

National University of Singapore

(Joint work with Hai WANG, Yuan Fang LI and Jing SUN)

September 2003

Semantic Web

"The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications." – W3C (www.w3.org/2001/sw)

Formal Specification

"The use of notations and languages with a defined mathematical meaning enable specifications, that is statements of what the proposed system should do, to be expressed with precision and no ambiguity. " – FME (www.fmeurope.org/fm.html)

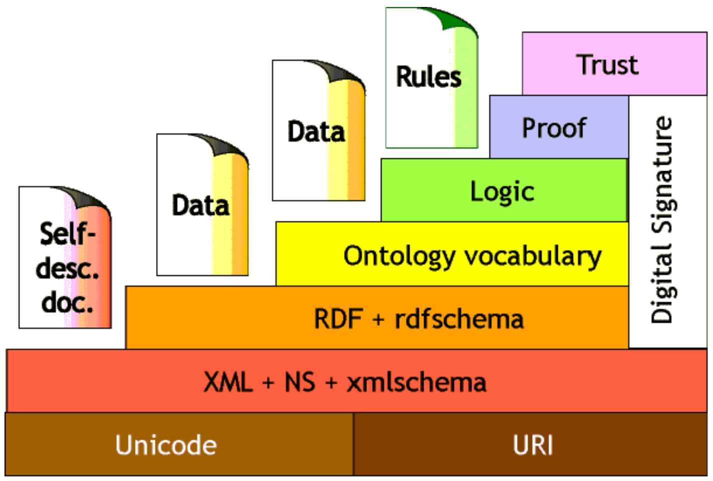
Overview

- Introduction to Semantic Web
  - RDF
  - DAML+OIL
  - OWL
- Semantic Web Environment for Formal Specifications
  - Formal Specification Languages and Their Integrations
  - Linking Different Formalisms through Semantic Web
  - Specification Comperhension via RDF Query
- Formal Methods for Semantic Web
  - Extracting DAML ontology from Z requirement models
  - Checking Web ontology in Z/EVES
  - Analysing and reasoning Web ontology in Alloy
- Conclusion and Further Work

Semantic Web

- Goals
  - Realizing the full potential of the Web
  - Making it possible for tools (agents) to effectively process information.
  - Ultimate goal - effective and efficient global information/knowledge exchange
- Building on proven ideas
  - Combines XML, RDF, hypertext and metadata approaches to linked information
  - Focuses on general principles of Web automation and data aggregation

Semantic Web Architectural Dependencies



www.w3c.org (by Tim Berners-Lee)

RDF, DAML+OIL and OWL

- Resource Description Framework (RDF) — 1999
  - An RDF document is a collection of assertions in *subject verb object* form for describing web resources
  - Provides interoperability between applications that exchange machine-understandable information on the Web
  - Use XML as a syntax, include XMLNS, and URIs
- DARPA Agent Markup Language (DAML+OIL) — 2001
  - Semantic markup language based on RDF, and
  - Extends RDF(S) with richer modelling primitives
  - DAML currently combines Ontology Interchange Language (OIL).
- OWL Web Ontology Language — 2003
  - Based on DAML+OIL
  - Three levels support: Lite, DL, Full

HTML and XML

- HTML

```
<H1> Semantic Web and Formal Methods</H1>
  <UL>
        <LI> Teacher: Jin Song Dong
        <LI> Students: s19908, s20015
        <LI> Requirements: discrete maths
  </UL>
```

- XML

```
<course>
    <title> Semantic Web and Formal Methods </title>
    <teacher> Jin Song Dong </teacher>
    <students> s19908, s20015 </students>
    <req> discrete maths </req>
</course>
```

Lack semantics in XML

- The XML is accepted as the emerging standard for data interchange on the Web. XML allows authors to create their own markup (e.g. `<course>`), which seems to carry some semantics.
- However, from a computational perspective tags like `<course>` carries as much semantics as a tag like `<H1>`. A computer simply does not know, what a course is and how the concept course is related to other concepts.
- XML may help humans predict what information might lie “between the tags” in the case of `<students> </students>`, but XML can only help.
- Only feasible for closed collaboration, e.g., agents in a small and stable community/intranet

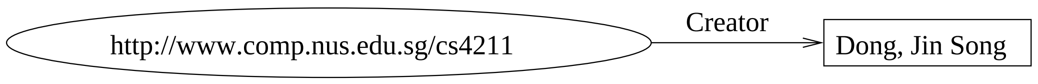
RDF Basics

- Resources — Things being described by RDF expressions. Resources are always named by URIs, e.g.
  - HTML Document
  - Specific XML element within the document source.
  - Collection of pages
- Properties — Specific aspect, characteristic, attribute or relation used to describe a resource, e.g. Creator, Title ...
- Statements —
  Resource (Subject) + Property (Predicate) + Property Value (Object)

RDF Statement Example 1

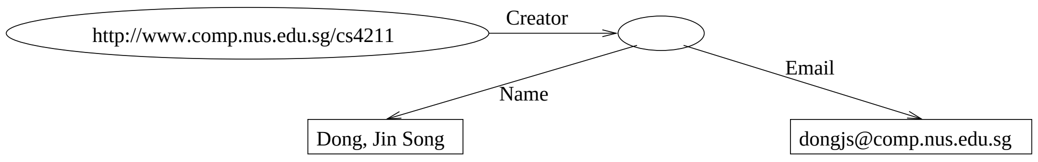
Dong, Jin Song is the creator of the web page
`http://www.comp.nus.edu.sg/cs4211`

- Subject (Resource) - `http://www.comp.nus.edu.sg/cs4211`
- Predicate (Property) - Creator
- Object (Literal) Dong, Jin Song



RDF Statement Example 2

Dong, Jin Song whose e-mail is `dongjs@comp.nus.edu.sg` is the creator of the web page `http://www.comp.nus.edu.sg/cs4211`



RDF in XML syntax

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">

     <rdf:Description about="http://www.comp.nus.edu.sg/cs4211">
         <dc:creator>Dong, Jin Song</dc:creator>
         <dc:title>Advanced Software Engineering</dc:title>
         <dc:date>2000-07-01</dc:date>
     </rdf:Description>

</rdf:RDF>
```

RDF Containers

- Bag - An unordered list of resources or literals
- Sequence - An ordered list of resources or literals
- Alternative - A list of resources or literals that represent alternatives for the value of a property

Container example: Sequence

Statement: The students of the course CS4211 in alphabetical order are Yuanfang Li, Jun Sun and Hai Wang .

```
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
         xmlns:s="http://www.schemas.org/Course/">
        <rdf:Description about=http://www.comp.nus.edu.sg/~cs4211>
           <s:students>
                 <rdf:Seq>
                 <rdf:li rdf:resource="http://www.comp.nus.edu.sg/~liyf"/>
                 <rdf:li rdf:resource="http://www.comp.nus.edu.sg/~sunj"/>
                 <rdf:li rdf:resource="http://www.comp.nus.edu.sg/~wangh"/>
                 </rdf:Seq>
           </s:students>
        </rdf:Description>
</rdf:RDF>
```

RDF Schema

- Basic vocabulary to describe RDF vocabularies, e.g.,
  `Class, subClassOf, Property, subPropertyOf, domain, range`
- Defines properties of the resources (e.g., title, author, subject, etc)
- Defines kinds of resources being described (books, Web pages, people, etc)
- XML Schema gives specific constraints on the structure of an XML document
  RDF Schema provides information about the interpretation of the RDF statements
- RDF schema uses XML syntax, but could theoretically use any other syntax

RDF Schema Example (Class)

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdfs:Class rdf:ID="Person">
        <rdfs:comment>Person Class</rdfs:comment>
        <rdfs:subClassOf
         rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Resource"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Student">
        <rdfs:comment>Student Class</rdfs:comment>
        <rdfs:subClassOf rdf:resource="#Person"/>
</rdfs:Class>
```

RDF Schema Example (Property)

```
<rdf:Property rdf:ID="teacher">
       <rdfs:comment>Teacher of a course</rdfs:comment>
       <rdfs:domain rdf:resource="#Course"/>
       <rdfs:range rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property rdf:ID="students">
       <rdfs:comment>List of Students in alphabetical order</rdfs:comment>
       <rdfs:domain rdf:resource="#Course"/>
       <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"/
</rdf:Property>
```

Why RDF(S) is not enough

- Only range/domain constraints on properties (need others)
- No properties of properties (unique, transitive, inverse, etc.)
- No equivalence, disjointness, etc.
- No necessary and sufficient conditions (for class membership)

DAML+OIL

- Europe: Ontology Inference Language (OIL) extends RDF Schema to a fully-fledged knowledge representation language.
- US: DARPA Agent Markup Language (DAML)
- Merged as DAML+OIL in 2001
  - logical expressions
  - data-typing
  - cardinality
  - quantifiers
- Becomes OWL — W3C standard in March 2003

DAML: Setting up the namespaces

```
<rdf:RDF
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd ="http://www.w3.org/2000/10/XMLSchema#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
>
```

DAML: Define Classes

```
<rdfs:Class rdf:ID="Animal"> <rdfs:label>Animal</rdfs:label> </rdfs:Class>
<rdfs:Class rdf:ID="Male">
   <rdfs:subClassOf rdf:resource="#Animal"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Female">
   <rdfs:subClassOf rdf:resource="#Animal"/>
   <daml:disjointWith rdf:resource="#Male"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Man">
   <rdfs:subClassOf rdf:resource="#Person"/>
   <rdfs:subClassOf rdf:resource="#Male"/> </rdfs:Class>
```

DAML: Define Properties

```
<rdf:Property rdf:ID="hasParent">
   <rdfs:domain rdf:resource="#Animal"/>
   <rdfs:range rdf:resource="#Animal"/>
</rdf:Property>
<rdf:Property rdf:ID="hasFather">
   <rdfs:subPropertyOf rdf:resource="#hasParent"/>
   <rdfs:range rdf:resource="#Male"/>
</rdf:Property>
```

DAML: Define Restrictions

```
<rdfs:Class rdf:ID="Person"> <rdfs:subClassOf rdf:resource="#Animal"/>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#hasParent"/>
      <daml:toClass rdf:resource="#Person"/>
    </daml:Restriction> </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="#hasFather"/>
    </daml:Restriction> </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:maxcardinality="1">
      <daml:onProperty rdf:resource="#hasSpouse"/>
    </daml:Restriction> </rdfs:subClassOf>
</rdfs:Class>
```

DAML: UniqueProperty and Transitive

```
<daml:UniqueProperty rdf:ID="hasMother">
   <rdfs:subPropertyOf rdf:resource="#hasParent"/>
   <rdfs:range rdf:resource="#Female"/>
</daml:UniqueProperty>

<daml:TransitiveProperty rdf:ID="hasAncestor">
   <rdfs:label>hasAncestor</rdfs:label>
</daml:TransitiveProperty>
```

DAML: oneOf

```
<rdf:Property rdf:ID="hasHeight">
  <rdfs:range rdf:resource="#Height"/>
</rdf:Property>

<rdfs:Class rdf:ID="Height">
  <daml:oneOf rdf:parseType="daml:collection">
    <Height rdf:ID="short"/>
    <Height rdf:ID="medium"/>
    <Height rdf:ID="tall"/>
  </daml:oneOf>
</rdfs:Class>
```

DAML: hasValue and intersectionOf

```
<rdfs:Class rdf:ID="TallThing">
  <daml:sameClassAs>
    <daml:Restriction>
       <daml:onProperty rdf:resource="#hasHeight"/>
       <daml:hasValue rdf:resource="#tall"/>
    </daml:Restriction>
  </daml:sameClassAs>
</rdfs:Class>
<rdfs:Class rdf:ID="TallMan">
  <daml:intersectionOf rdf:parseType="daml:collection">
     <rdfs:Class rdf:about="#TallThing"/>
     <rdfs:Class rdf:about="#Man"/>
  </daml:intersectionOf>
</rdfs:Class>
```

DAML: instances

```
<Person rdf:ID="Adam">
   <rdfs:label>Adam</rdfs:label>
   <rdfs:comment>Adam is a person.</rdfs:comment>
   <hasHeight rdf:resource=#medium/>
</Person>
```

OWL: The three sublanguages

- *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1.
- *OWL DL* supports those users who want the maximum expressiveness while retaining computational completeness and decidability. OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class).
- *OWL Full* is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right.

OWL: Changes from DAML+OIL

- With respect to the three sublanguages, the DAML+OIL semantics is closests to the OWL DL semantics.
- The namespace was changed to `http://www.w3.org/2002/07/owl`
- Cyclic subclasses are now allowed
- multiple `rdfs:domain` and `rdfs:range` properties are handled as intersection
- Various properties and classes were renamed, e.g., `daml:UniqueProperty` is replaced by `owl:FunctionalProperty`
- ... `http://www.w3.org/TR/owl-ref/`

Recall: Overview

- Introduction to Semantic Web
  - RDF
  - DAML+OIL
  - OWL
- ✓ Semantic Web Environment for Formal Specifications
  - Formal Specification Languages and Their Integrations
  - Linking Different Formalisms through Semantic Web
  - Specification Comperhension via RDF Query
- Formal Methods for Semantic Web
  - Extracting DAML ontology from Z requirement models
  - Checking Web ontology in Z/EVES
  - Analysing and reasoning Web ontology in Alloy
- Conclusion and Further Work

Formal Specification Languages and Their Integrations

- Many formal specification techniques exist for modeling different aspects of software systems, however,
- it is difficult to find a single notation that can model all functionalities of a complex system.
- E.g., B/VDM/Z are designed for modeling system data/states, while CSP/π-calculus are designed for modeling system behaviour/interactions.
- Various formal notations are often extended and combined for modeling large and complex systems. In recent years, *integrated formal method* (IFM) has been a popular research topic, i.e.
  IFM'99 (York), IFM'00 (Dagstuhl), IFM'02 (Turku), IFM'04 (Kent).
- Due to different motivations, there are possible different semantic links between two formalisms, which can lead to different integrations between the two.

Integrated Formal Methods

- Unlike UML, an industrial effort for standardising diagrammatic notations, a single dominating integrated formal method may not exist in the near future. The reason may be partially due to the fact that
  - there are many different well established individual schools,
  - the open nature of the research community, i.e. FME, which is different from the industrial 'globalisation' community, i.e. OMG.
- Regardless of whether there will be or there should be an ultimate integrated formal method (like UML), *diversity* seems to be the current reality for formal methods and their integrations. Such a diversity may have an advantage, that is, different formal methods and their combinations may be suitable for different kinds of complex systems modeling.
- Challenge: to develop environment/tools for extending and combining various formal specification techniques — *Semantic Web*

Semantic Web environment for Z

```
<rdf:RDF
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:daml = "http://www.daml.org/2001/03/daml+oil#"
  xmlns:z = "http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#">
<!-- ... -->
  <rdfs:Class rdf:ID="Schemadef">
    <rdfs:label>Schemadef</rdfs:label> </rdfs:Class>
  <rdfs:Class rdf:ID="Schemadefbox">
    <rdfs:label>Schemadefbox</rdfs:label>
      <rdfs:subClassOf rdf:resource="#Schemadef"/>
      <rdfs:subClassOf>
        <daml:Restriction daml:cardinalityQ="1">
          <daml:onProperty rdf:resource="#name"/></daml:Restriction></rdfs:subClassOf>
      <rdfs:subClassOf>
        <daml:Restriction daml:minCardinality="0">
          <daml:onProperty rdf:resource="#delta"/>
          <daml:toClass rdf:resource="#Schemadef"/></daml:Restriction></rdfs:subClassOf>
<!-- ... -->
```

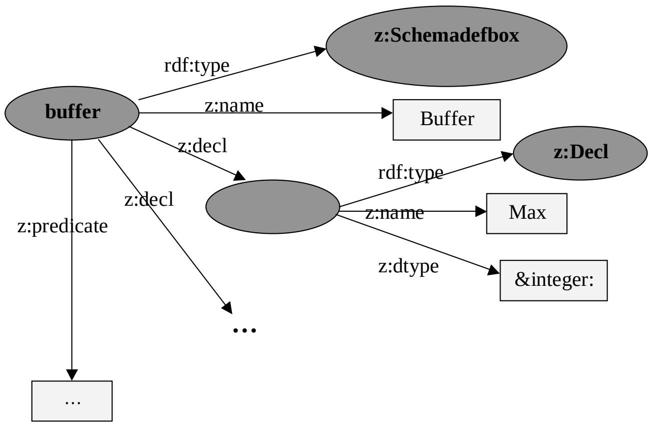
Example: Semantic Web environment for Z

Buffer

$Max : \mathbb{Z}$

$items : \mathrm{seq}\, MSG$

$\#items \leq Max$



Z Semantic Web environments can be easily extended for Object-Z. Similarly, other formalisms, i.e. CSP and TCSP, can be also constructed. Linking different formalisms is an interesting issue.

Semantics Links

- Various modeling methods can be used in an effective combination for designing complex systems if the semantic links between those methods can be clearly established and defined.
  - e.g. ‘mapping sets of Z operations into CSP actions’ A. Hall [FME’02]
- Given two sets of formalisms, say state-based ones and event-based ones, it’s not too surprising to see that different possible integrations are more than the cross-product of the two sets. This is simply because the different semantic links between the two formalisms lead to different integrations.
- Furthermore, the semantic links can be directional and bi-directional.

Object-Z ⌢ CSP : *class* $\Longrightarrow$ *process* (Smith and Derrick, FME'97)

Buffer

$max : \mathbb{N};\quad items : \mathrm{seq}\, MSG$

$\#items \leq max$

INIT

$items = \langle\,\rangle$

Join

$\Delta(items)$

$i? : MSG$

$\#items < max$

$items' = \langle i?\rangle ^\frown items$

Leave

$\Delta(items)$

$i! : MSG$

$\#items \neq 0$

$items = items' ^\frown \langle i!\rangle$

$Buffer_1 \mathrel{\widehat=} Buffer[Transfer/Leave]$

$Buffer_2 \mathrel{\widehat=} Buffer[Transfer/Join]$

$TwoLinkedBuffers \mathrel{\widehat=} Buffer_1 \mathrel{|[}\, Transfer \,]|\, Buffer_2$

Object-Z ⌢ CSP : *operation* $\Longleftrightarrow$ *process* (Mahony and Dong, ICSE'98)

TBuffer

Buffer

$left, right : \mathbf{chan}$ [input and output channels]

$\mathrm{MAIN} \mathrel{\widehat=} \mu\, Q \bullet ([i : MSG] \bullet left?i \rightarrow Join \;\Box$

$\quad [\#items \neq 0] \bullet right!last(items) \rightarrow Leave);\; Q$

Two Communicating Buffers

$TwoLinkedBuffers_a$

$l : TBuffer[middle/right]$

$r : TBuffer[middle/left]$

$\mathrm{MAIN} \mathrel{\widehat=} l \mathrel{|[}\, middle \,]|\, r$

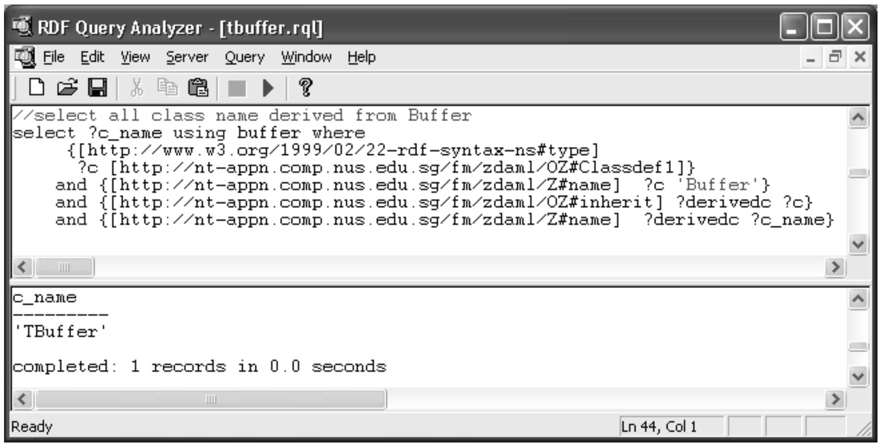
Semantic Web for linking Object-Z and CSP

class $\Longrightarrow$ *process*

```
<daml:Ontology rdf:about="">
    <daml:imports rdf:resource="http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ"/>
    <daml:imports rdf:resource="http://nt-appn.comp.nus.edu.sg/fm/zdaml/CSP"/>
</daml:Ontology>
<rdfs:Class rdf:about="oz:Classdef">
    <rdfs:subClassOf rdf:resource="csp:Pro"/> </rdfs:Class>
```

operation $\Longleftrightarrow$ *process*

```
<daml:ObjectProperty rdf:ID="MAIN">
    <rdfs:range rdf:resource="csp:Process"/>
    <rdfs:domain rdf:resource="#Classdef"/> </daml:ObjectProperty>
<rdfs:Class rdf:about="oz:OP">
    <daml:sameClassAs rdf:resource="csp:Process"/> </rdfs:Class>
```


RDF Query Analyzer - [tbuffer.rql]
File Edit View Server Query Window Help
Ready
Ln 44, Col 1

```
//select all class name derived from Buffer
select ?c_name using buffer where
     {[http://www.w3.org/1999/02/22-rdf-syntax-ns#type]
      ?c [http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#Classdef1]}
    and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name]  ?c 'Buffer'}
    and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#inherit] ?derivedc ?c}
    and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name]  ?derivedc ?c_name}
```

```
c_name
---------
'TBuffer'

completed: 1 records in 0.0 seconds
```

Find all the sub-classes of the *Buffer*

Specification Comprehension

Recall: Overview

- Introduction to Semantic Web
  - RDF
  - DAML+OIL
  - OWL
- Semantic Web Environment for Formal Specifications
  - Formal Specification Languages and Their Integrations
  - Linking Different Formalisms through Semantic Web
  - Specification Comperhension via RDF Query
- ✓ Formal Methods for Semantic Web
  - Extracting DAML ontology from Z requirement models
  - Checking Web ontology in Z/EVES
  - Analysing and reasoning Web ontology in Alloy
- Conclusion and Further Work

Problems in designing Semantic Web ontology/services

- Semantic Web languages are not expressive enough for designing Semantic Web complex ontology properties and service/agents.

Require a systematic design process with expressive high level modeling techniques

Solution: formal specifications

Some DAML constructs in Abstract Form

Abstract DAML constructs	**Description**
$daml\_class$	classes
$daml\_subclass[C]$	subclasses of C
$daml\_objectproperty[D \leftrightarrow R]$	relation properties with domain D, range R
$daml\_objectproperty[D \rightarrow R]$	function properties with domain D, range R
$daml\_subproperty[P]$	sub properties of P
$instanceof[C]$	instances of the DAML class C

Extracting DAML ontology from the Z model

Z can be used to model web-based ontology at various levels. The conceptual domain Z models can be transformed to DAML+OIL ontology via XSLT technology.

Given type transformation

$$\frac{[T]}{T \in daml\_class}$$

e.g.

$[Author]$

```
<daml:class rdf:ID="author">
    <rdfs:label>Author</rdfs:label> </daml:Class>
```

Z schema transformation

$$\begin{array}{|l} \hline S \\ X : T_1; \quad Y : \mathbb{P}\, T_2 \\ \hline \end{array} \qquad T_1, T_2 \in \mathit{daml\_class}$$

$$S \in \mathit{daml\_class},\ X \in \mathit{daml\_objectproperty}[S \to T_1], Y \in \mathit{daml\_objectproperty}[S \leftrightarrow T_2]$$

$$\begin{array}{|l} \hline \mathit{Paper} \\ \mathit{title} : \mathit{Title}; \quad \mathit{authors} : \mathbb{P}\, \mathit{Author} \\ \hline \end{array}$$

```
<daml:class rdf:ID="paper"> <rdfs:label>Paper</rdfs:label> </daml:Class>
<daml:ObjectProperty rdf:ID="paper_title"> <rdf:type rdf:resource="
    http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <rdf:domain rdf:resource="#paper"/>
  <rdf:range rdf:resource="#title"/> </daml:ObjectProperty>
<daml:ObjectProperty rdf:ID="paper_authors">
  <rdf:domain rdf:resource="#paper"/>
  <rdf:range rdf:resource="#author"/> </daml:ObjectProperty>
```

Z axiomatic definition transformation (relation/functions)

$$\begin{array}{|l} R : B \leftrightarrow (\to, \nrightarrow)\, C \\ \hline \ldots \end{array} \qquad B, C \in \mathit{daml\_class}$$

$$R \in \mathit{daml\_objectproperty}[B \leftrightarrow (\to, \nrightarrow)\, C]$$

$$\begin{array}{|l} \mathit{reference} : \mathit{Paper} \leftrightarrow \mathit{Paper} \end{array}$$

```
<daml:ObjectProperty rdf:ID="paper_reference">
  <rdfs:domain rdf:resource="#paper"/>
  <rdfs:range rdf:resource="#paper"/>
</daml:ObjectProperty>
```

Z axiomatic definition transformation (subset)

$$\frac{\begin{array}{|l} M : \mathbb{P}\, N \\ \hline \ldots \end{array} \qquad N \in \mathit{daml\_class}}{M \in \mathit{daml\_subclass}[N]}$$

$$\begin{array}{|l} \mathit{Biannual} : \mathbb{P}\, \mathit{ConfSeries} \\ \hline \ldots \end{array}$$

```
<daml:class rdf:ID="biannual">
  <rdfs:subClassOf rdf:resource="#confseries"/>
</daml:class>
```

Improve the ontology quality through Z tools

Z/EVES tool is an interactive system for composing, checking, and analyzing Z specifications. It supports the analysis of Z specifications in several ways: syntax and type checking, schema expansion, precondition calculation, domain checking, and general theorem proving. Some ontology related flaws in Z model can be detected and removed with the assistance of Z/EVES so that the transformed DAML ontology from checked Z model will have better quality.

Alternatively, one can develop reverse transformation tools from DAML ontology to the formal specifications then to use formal specification tools to detect domain and logical errors that the current DAML reasoner is not able to detect.

Ecoding DAML semantics in Z/EVES

Basics

$[\mathit{Resource}]$

$$\mathit{Class} : \mathbb{P}\,\mathit{Resource}$$

$$\begin{array}{|l}\mathit{Property} : \mathbb{P}\,\mathit{Resource} \\ \hline \mathit{Class} \cap \mathit{Property} = \{\}\end{array}$$

$$\mathit{instances} : \mathit{Class} \rightarrow \mathbb{P}\,\mathit{Resource}$$

$$\mathit{sub\_val} : \mathit{Property} \rightarrow (\mathit{Resource} \leftrightarrow \mathit{Resource})$$

Class Relationships

$$\begin{array}{|l}\mathit{subClassOf} : \mathit{Class} \leftrightarrow \mathit{Class} \\ \mathit{disjointWith} : \mathit{Class} \leftrightarrow \mathit{Class} \\ \mathit{sameClassAs} : \mathit{Class} \leftrightarrow \mathit{Class} \\ \hline \forall c_1, c_2 : \mathit{Class} \bullet \\ \quad c_1 \ \underline{\mathit{subClassOf}}\ c_2 \Leftrightarrow \mathit{instances}(c_1) \subseteq \mathit{instances}(c_2) \\ \quad c_1 \ \underline{\mathit{disjointWith}}\ c_2 \Leftrightarrow \mathit{instances}(c_1) \cap \mathit{instances}(c_2) = \{\} \\ \quad c_1 \ \underline{\mathit{sameClassAs}}\ c_2 \Leftrightarrow \mathit{instances}(c_1) = \mathit{instances}(c_2)\end{array}$$

$$\begin{array}{|l}\mathit{unionOf} : \mathrm{seq}\,\mathit{Class} \leftrightarrow \mathit{Class} \\ \hline \forall\, cl : \mathrm{seq}\,\mathit{Class};\ c : \mathit{Class} \bullet cl\ \underline{\mathit{unionOf}}\ c \Leftrightarrow \\ \quad \mathit{instances}(c) = \bigcup\{x : \mathit{Class} \mid x \in \mathrm{ran}\,cl \bullet \mathit{instances}(x)\}\end{array}$$

$$\begin{array}{|l}\mathit{disjointUnionOf} : \mathrm{seq}\,\mathit{Class} \leftrightarrow \mathit{Class} \\ \hline \forall\, cl : \mathrm{seq}\,\mathit{Class};\ c_1 : \mathit{Class} \bullet cl\ \underline{\mathit{disjointUnionOf}}\ c \Leftrightarrow \\ \quad cl\ \underline{\mathit{unionOf}}\ c \wedge \forall\, x, y : \mathrm{ran}\,cl \bullet x\ \underline{\mathit{disjointWith}}\ y\end{array}$$

Class and Property

$$toClass : (Class \times Property) \leftrightarrow Class$$

$$\forall c_1, c_2 : Class;\ p : Property \bullet (c_1, p)\ \underline{toClass}\ c_2 \Leftrightarrow$$
$$\quad \forall a_1, a_2 : Resource \bullet a_1 \in instances(c_1) \Leftrightarrow ((a_1, a_2) \in sub\_val(p) \Rightarrow a_2 \in instances(c_2))$$

$$hasValue : (Class \times Property) \leftrightarrow Resource$$

$$\forall c : Class;\ p : Property;\ r : Resource \bullet$$
$$\quad (c, p)\ \underline{hasValue}\ r \Leftrightarrow \{r\} = sub\_val(p)(\!|\ instances(c)\ |\!)$$

$$hasClass : (Class \times Property) \leftrightarrow Class$$

$$\forall c_1 : Class;\ p : Property;\ c_2 : Class \bullet (c_1, p)\ \underline{hasClass}\ c_2 \Leftrightarrow$$
$$\quad \forall x : instance(c_1) \bullet sub\_val(p)(\!|\ \{x\}\ |\!) \cap instances(c_2) \neq \varnothing$$

$$cardinality : (Class \times Property) \nrightarrow \mathbb{N}$$

$$\forall c : Class;\ p : Property;\ n : \mathbb{N} \bullet$$
$$\quad ((c, p), n) \in cardinality \Leftrightarrow \forall x : instance(c) \bullet \#(sub\_val(p)(\!|\ \{x\}\ |\!)) = n$$

Property Relationships

$$subPropertyOf : Property \leftrightarrow Property$$

$$\forall p1, p2 : Property \bullet p1\ \underline{subPropertyOf}\ p2 \Leftrightarrow sub\_val(p1) \subseteq sub\_val(p2)$$

$$samePropertyAs : Property \leftrightarrow Property$$

$$\forall p1, p2 : Property \bullet p1\ \underline{samePropertyAs}\ p2 \Leftrightarrow sub\_val(p1) = sub\_val(p2)$$

$$inverseOf : Property \leftrightarrow Property$$

$$\forall p1, p2 : Property \bullet p1\ \underline{inverseOf}\ p2 \Leftrightarrow sub\_val(p1) = (sub\_val(p2))^{\sim}$$

$$UniqueProperty : \mathbb{P}\ Property$$

$$\forall p : Property;\ x, y, z : Resource \bullet p \in UniqueProperty \Leftrightarrow$$
$$\quad ((x, y) \in sub\_val(p) \land (x, z) \in sub\_val(p) \Rightarrow\ y = z)$$

Imported in Z/EVES

```
\begin{zsection}{daml2z}{toolkit}

\begin{zed}
[Resource]
\end{zed}

\begin{axdef}
Class: \power Resource
\end{axdef}

\begin{axdef}
Property: \power Resource
\where
Class \cap Property = \{\}
\end{axdef}
⋮
```

```
\begin{axdef}
subClassOf: Class \rel Class
\where
\forall c_1, c_2: Class @
c_1 \inrel{subClassOf} c_2 \iff\\
  instances(c_1) \subseteq instances(c_2)
\end{axdef}
```

Checking Military Plan Ontology Experience

- Singapore DSO has developed an information extraction engine which has been used to generate military plan ontologies (in DAML) from military formation and plan (in natural language).
- A military ontology is made up of the following four main ingedients.
  - A set of military operations and tasks, which define the logic order, type , and phases of a military campaign.
  - A set of military units, which are the participants of the military operations and tasks,
  - A set of geographic locations, where such operations take place and
  - A set of time points for constraining the timing of such operations.
- We have developed an auto transformation tool that takes DAML document and produces Z specifications, then we use Z/EVES tool to check the type errors and ontology consistency issues.
- Checking beyond web ontology (e.g. one military unit assigned two different tasks at the same time period)

A Real Military Case Study Statistics

Items	Numbers
Resources	138
Operations, tasks, phases	56
Units	47
Geographic areas	35
Statements (in RDF)	592
Transformed Axiomatic Defs (in Z)	138
Transformed Predicates (in Z)	410
Type errors	22
DAML related ontology errors	0
errors beyound DAML	2

Checking Semantic Web through Alloy

Alloy overview

Alloy (developed at MIT by D. Jackson's group) is a structural modelling language based on first-order logic.

Signature: A signature (`sig`) paragraph introduces a basic type and a collection of relation (called field) in it along with the types of the fields and constraints on their value. A signature may inherit fields and constraints from another signature.

Function: A function (`fun`) captures behaviour constraints. It is a parameterized formula that can be "applied" elsewhere,

Fact: Fact (`fact`) constrains the relations and objects. A `fact` is a formula that takes no arguments and need not to be invoked explicitly; it is always true.

Assertion: An assertion (`assert`) specifies an intended property. It is a formula whose correctness needs to be checked, assuming the facts in the model.

DAML semantic encoding in Alloy

```
module DAMLOIL
...
```

The semantic models for DAML (for DAML+OIL) are encoded in the module `DAMLOIL`.

- Resource

  ```
  sig Resource {}
  ```

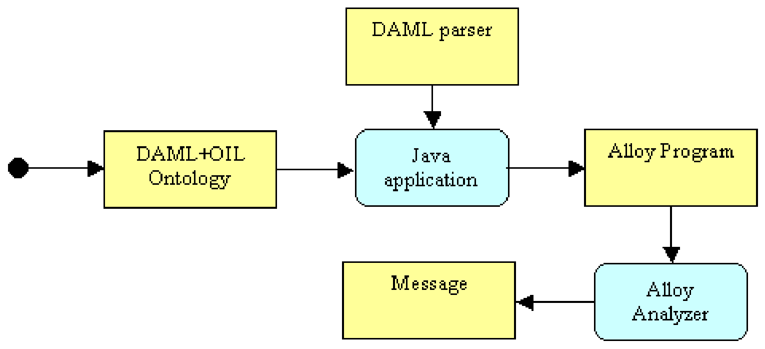
- Property

  ```
  disj sig Property extends Resource
      {sub_val: Resource -> Resource}
  ```

- subClassOf

  ```
  fun subClassOf(csup, csub:  Class)
      {csub.instances in csup.instances}
  ```

DAML to Alloy Transformation



DAML to Alloy Transformation

- DAML class transformation

$$\frac{C \in daml\_class}{static\ disj\ sig\ C\ extends\ Class\{\}}$$

  A daml_class `C` will be transformed into a scalar `C`, constrained to be an elements of the signature `Class`.

- DAML property transformation

$$\frac{P \in daml\_property}{static\ disj\ sig\ P\ extends\ Property\{\}}$$

  A daml_property `p` will be transformed into a scalar `P`, constrained to be an elements of the signature `Property`.

... (further detail see a technical paper presentation at FM'03)

Summary and Future Work

- Semantic Web
  - ✓ good support for automation, collaboration, extension and integration
  - × less expressive and no systematic design process for web ontology/agents
- Formal Specifications
  - ✓ expressive, diverse and can be combined effectively
  - × weak in linking various methods for collaborative design
- Approaches
  - ✓ Semantic Web environment for linking various formalisms (FME'02)
  - ✓ Extracting web ontologies systematically from Z specifications (ICFEM'02)
  - ✓ Checking Semantic Web through Alloy (FM'03)
- Vision:
  - (?) lightweight meta integrator
  - (?) formal specification languages as semantic web languages