Semantic Web and Formal Specifications

short version of ICSE'04 tutorial

DONG Jin Song

(www.comp.nus.edu.sg/∼dongjs)

Computer Science Department

National University of Singapore

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

June 2004

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)

Semantic Web is the main focus of WWW'04 May 18-22 2004 (at NYC).

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

- Seminar I: Introduction to Formal Specification and Semantic Web
  - Software Modeling Techniques: Z and Alloy
  - Semantic Web Languages: RDF, DAML+OIL, OWL
- Seminar II: Software Design Method/Tools for Semantic Web
  - Extracting DAML ontology from UML/Z models
  - Semantics of DAML+OIL in Z/Alloy
  - Combined Approach to Reasoning about Semantic Web

The Z Specification Language

- developed originally at Programming Research Group, Oxford University
- based on set theory and predicate logic
- J. Woodcock and J. Davies, Using Z: Specification, Refinement, and Proof. Prentice-Hall, 1996

Types

Z is strongly typed: every expression is given a type.

Any set can be used as a type.

The following are equivalent within set comprehension

$(x, y) : A \times B$
$x : A;\ y : B$
$x, y : A \qquad (\text{when } B = A)$

Notice that

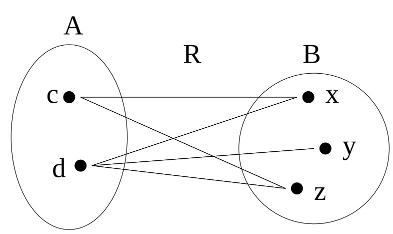
$\forall S : \mathbb{P}\, A \bullet \ldots \quad \text{not} \quad \forall S \subseteq A \bullet \ldots$

Relations

A relation R from A to B, denoted by

$R : A \leftrightarrow B,$

is a subset of $A \times B$.



R is the set $\{(c, x), (c, z), (d, x), (d, y), (d, z)\}$

Notation: the predicates

$(c, z) \in R \quad \text{and} \quad c \mapsto z \in R \quad \text{and} \quad c\, \underline{R}\, z$

are equivalent.

$\text{dom}\, R \quad \text{is the set} \quad \{a : A \mid \exists\, b : B \bullet a\, \underline{R}\, b\}$
$\text{ran}\, R \quad \text{is the set} \quad \{b : B \mid \exists\, a : A \bullet a\, \underline{R}\, b\}$

Domain and Range Restriction/Subtraction

Suppose $R : A \leftrightarrow B$ and $S \subseteq A$ and $T \subseteq B$; then

$$
\begin{array}{lll}
S \lhd R & \text{is the set} & \{(a,b) : R \mid a \in S\} \\
R \rhd T & \text{is the set} & \{(a,b) : R \mid b \in T\}
\end{array}
$$

$$
\begin{array}{lll}
S \ntriangleleft R & \text{is the set} & \{(a,b) : R \mid a \notin S\} \\
R \ntriangleright T & \text{is the set} & \{(a,b) : R \mid b \notin T\}
\end{array}
$$

e.g. if

$has\_sibling : People \leftrightarrow People$ then

$female \lhd has\_sibling$	is the relation	is_sister_of
$has\_sibling \rhd female$	is the relation	has_sister
$female \ntriangleleft has\_sibling$	is the relation	$is\_brother\_of$
$has\_sibling \ntriangleright female$	is the relation	$has\_brother$

Relational Image

Suppose $R : A \leftrightarrow B$ and $S \subseteq A$

$$R(\!|\ S\ |\!) = \{b : B \mid \exists\, a : S \bullet a\, \underline{R}\, b\}$$

$$R(\!|\ S\ |\!) \subseteq B$$

$$
\begin{aligned}
&divides(\!|\ \{8,9\}\ |\!) \\
&\quad = \{x : \mathbb{N} \mid \exists\, k : \mathbb{N} \bullet x = 8k \ \lor\ x = 9k\} \\
&\quad = \{\text{numbers divided by 8 or 9}\}
\end{aligned}
$$

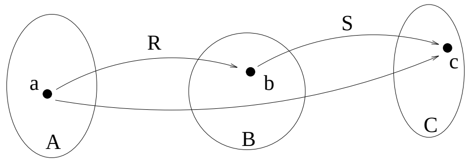
$$\leqslant (\!|\ \{7,3,21\}\ |\!) = \{x : \mathbb{N} \mid x \geqslant 3\}$$

$$has\_sibling(\!|\ male\ |\!) = \{\text{people who have a brother}\}$$

Relational Composition

Suppose $R : A \leftrightarrow B$ and $S : B \leftrightarrow C$

$$R \, ⨾ \, S = \{(a, c) : A \times C \mid \exists\, b : B \bullet a \underline{R}\, b \;\wedge\; b \underline{S}\, c\}$$



$R \, ⨾ \, S \in A \leftrightarrow C$

e.g.

$$is\_parent\_of \, ⨾ \, is\_parent\_of = is\_grandparent\_of$$

$$R^0 = \mathrm{id}[A], \quad R^1 = R, \quad R^2 = R \, ⨾ \, R, \quad R^3 = R \, ⨾ \, R \, ⨾ \, R, \ldots$$

Alloy Overview

Alloy (developed at MIT by D. Jackson's group) is a structural modelling language based on first-order logic (a subset of Z) and specifications organised in a tree of *modules*

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 parameterised 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.

Alloy Analyser (AA)

- Constraint solver with automated simulation & checking
- Transforms a problem into a (usually huge) boolean formula
- A *scope* (finite bound) must be given

Alloy Basics

`x` (a scalar), `{x}` (a singleton set containing a scalar), `(x)` (a tuple) and `{(x)}` (a relation) are all treated as the same as `{(x)}`. The relational composition (or join) and product:

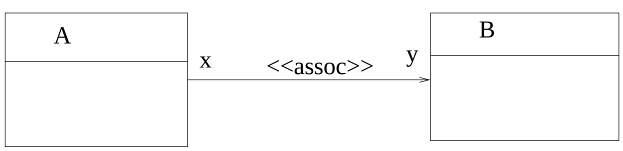
```
{(X1, .., Xm, S)}.{(S, Y1,.., Yn)} = {(X1, .., Xm, Y1, .., Yn)}
{(X1, .., Xm, S)} -> {(S, Y1,.., Yn)} = {(X1, .., Xm, S, S, Y1, .., Yn)}
```

Alloy Expression Examples

```
children = ~parents
ancestors = ^parents
descendants = ~ancestors
Man = Person - Woman
mother = parents & (Person->Woman)
father = parents & (Person->Man)
siblings = parents.~parents - iden [Person]
cousins = grandparents.~grandparents - siblings - iden [Person]
```


Alloy, UML and Z

Given the UML Class diagram



The corresponding Alloy expression:

`assoc: A x -> y B`

Given the Z expressions, the corresponding Alloy expressions:

in Z: $T_1 \rightarrow T_2$

`in Alloy:  T1 ->! T2`

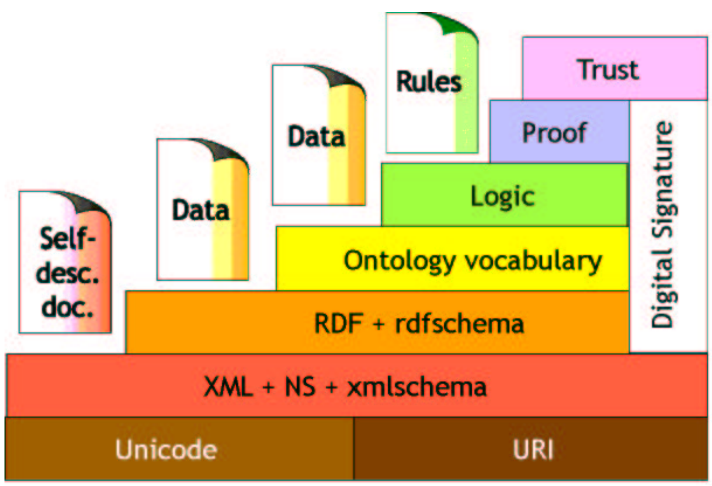
in Z: $T_1 \nrightarrow T_2$

`in Alloy:  T1 ->? T2`

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 combines Ontology Interchange Language (OIL).
- OWL Web Ontology Language — 2003 (become W3C rec)
  - Based on DAML+OIL
  - Three levels support: Lite, DL, Full

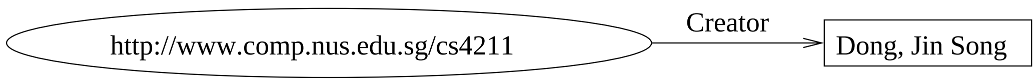
RDF Basics

- 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

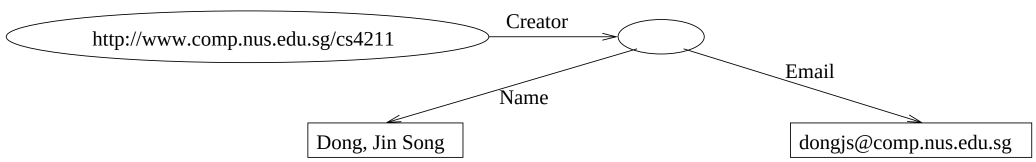
Dong, Jin Song is the creator of the web page
`http://www.comp.nus.edu.sg/cs4211`

- 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 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 2004

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>
```


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.

Recall Overview

Overview

- Seminar I: Introduction to Formal Specification and Semantic Web
  - Software Modeling Techniques: Z and Alloy
  - Semantic Web Languages: RDF, DAML+OIL, OWL
- ✓ Seminar II: Software Design Method/Tools for Semantic Web
  - Extracting DAML ontology from Z models
  - Semantics of DAML+OIL in Z/Alloy
  - Combined Approach to Reasoning about Semantic Web

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: software 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 Z conceptual domain 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

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

$S \in daml\_class,\ X \in daml\_objectproperty[S \rightarrow T_1],\ Y \in daml\_objectproperty[S \leftrightarrow T_2]$

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

```
<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)

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

$R \in daml\_objectproperty[B \leftrightarrow (\rightarrow, \nrightarrow)\, C]$

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

```
<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>
```


Ontology Tools: A Brief Survey

- RDF reasoner: Cwm, Triple
- **Fa**st **C**lassification of **T**erminologies (FaCT)
  - Supports consistency & subsumption reasoning (TBox)
  - Does not support instantiation reasoning (ABox)
- **R**enamed **A**Box and **C**oncept **E**xpression **R**easoner (RACER)
  - Supports TBox & ABox reasoning
  - Includes richer functionalities compared to FaCT
- FaCT & RACER are fully automated
- OilEd: graphical ontology editor that supports FaCT & RACER

Z/Alloy Semantics for DAML+OIL

Basic Concepts

- Resource

$[Resource]$

```
sig Resource {}
```

- Class & instances

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

```
disj sig Class extends Resource
  {instances: set Resource}
```

- Property & sub_val

$Property : \mathbb{P}\, Resource$

$Class \cap Property = \varnothing$

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

$sub\_val : Property \rightarrow (Resource \leftrightarrow Resource)$

Z/Alloy Semantics for DAML+OIL

Class Relationships

- subClassof & disjointWith

$\_subClassOf\_ : Class \leftrightarrow Class$
$\_disjointWith\_ : Class \leftrightarrow Class$

$\forall c_1, c_2 : Class \bullet$
$c_1 \;\underline{subClassOf}\; c_2 \Leftrightarrow instances(c_1) \in \mathbb{P}\, instances(c_2)$
$c_1 \;\underline{disjointWith}\; c_2 \Leftrightarrow instances(c_1) \cap instances(c_2) = \varnothing$

```
fun subClassOf(c1, c2: Class)
    {c2.instances in c1.instances}
fun disjointWith (c1, c2: Class)
    {no c1.instances & c2.instances}
```

Z/Alloy Semantics for DAML+OIL

Class & Property

- toClass

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

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

```
fun toClass (p:Property, c1:Class, c2:Class)
{all a1, a2: Resource | a1 in c1.instances <=>
    a2 in a1.(p.sub_val) => a2 in c2.instances}
```

  - Example: Anything that breathes by gill is a fish, including all those don't breathe at all!

$$Fish, Gill : Class$$
$$Breathe\_by : Property$$
$$(Fish, Breathe\_by)\ \underline{toClass}\ Gill$$

Z/Alloy Semantics for DAML+OIL

Property Relationships

- subPropertyOf

$$subPropertyOf : Property \leftrightarrow Property$$

$$\forall p_1, p_2 : Property \bullet p_1\ \underline{subPropertyOf}\ p_2 \Leftrightarrow sub\_val(p_1) \in \mathbb{P}\, sub\_val(p_2)$$

```
fun subPropertyOf (p1, p2:Property)
    {p1.sub_val in p2.sub_val}
```

Military Plan Ontology

- Developed by DSO Singapore, defining concepts in military domain: `military.daml`
- Instance ontologies generated from plain text by IE engine
- Contains sets of
  - Military operations & tasks
  - Military units
  - Geographic locations
  - Time points

Transformation

- DAML+OIL to Z
  - Developed a Java tool for automatic transformation
  - Supports both plan & instance ontologies
  - A number of enhancements made
    * Z predicates marked by *labels* as (rewrite or assumption) rules
    * Time points modeled as natural numbers $\mathbb{N}$
    * Domain-specific theorems are added
    * Supports Unique Name Assumption
    * Additional predicates added to facilitate proof
- DAML+OIL to Alloy
  - More straightforward
  - Using an XSLT stylesheet

Transformation: Example

- DAML+OIL:

```
<daml:Class rdf:about="http://www.dso.org.sg/
   PlanOntology#MilitaryTask">
  <rdfs:label>MilitaryTask</rdfs:label>
  <rdfs:subClassOf>
    <daml:Class rdf:about="http://www.dso.org.sg/
      PlanOntology#MilitaryProcess"/>
  </rdfs:subClassOf>
</daml:Class>
```

- Z:

$$MilitaryTask : Class$$

$$\langle\langle \text{grule} \quad \text{MilitaryTask\_subClassOf\_MilitaryProcess} \rangle\rangle$$
$$(MilitaryTask, MilitaryProcess) \in subClassOf$$

- Alloy:

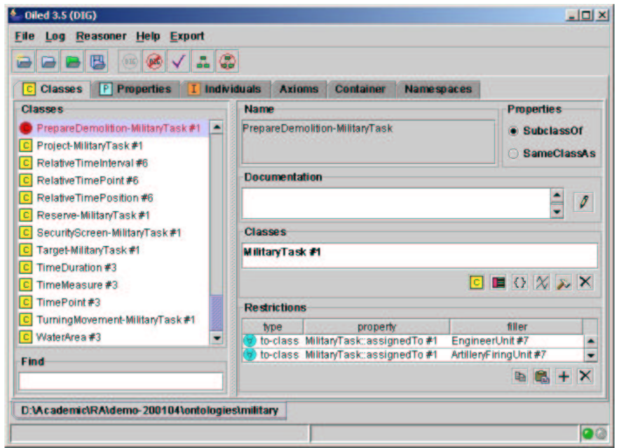
```
static disj sig MilitaryTask extends Class {}
fact{subClass(MilitaryProcess, MilitaryTask)}
```


The Combined Approach

1. Transforms ontology to Z & type-check using Z/EVES
   - Semi-automated
2. Use RACER & OilEd to check for ontological inconsistencies
3. If inconsistencies found, use AA to pinpoint them
   - Iterate steps 2 & 3 until RACER finds no inconsistency
4. If an instance ontology, use Z/EVES to check for properties inexpressible in DAML+OIL & Alloy
   - Interactive...

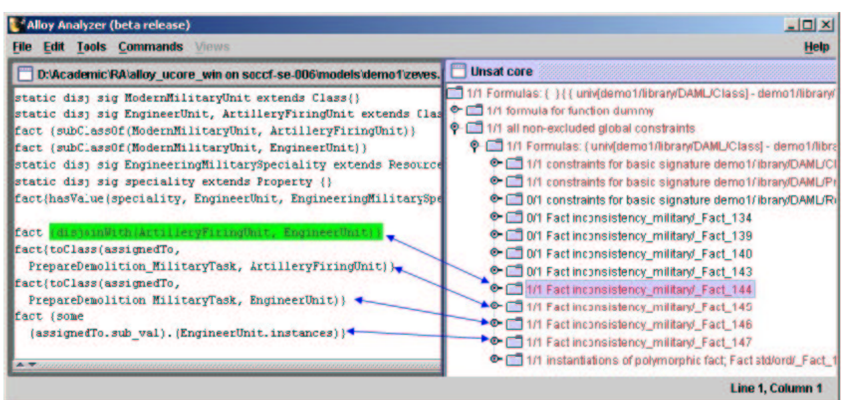
Standard SW Reasoning

- Step 1: Z/EVES finds no type errors in (transformed) `military.daml`
- Step 2: RACER complains about an inconsistent class, PrepareDemolition-MilitaryTask on the left



Standard SW Reasoning (continued)

- However, RACER cannot tell where the inconsistency is
- Step 3: Extract fragment of ontology according to OilEd
- AA finds the inconsistency, and it gives the possible cause in red color



More Advanced Reasoning

- Applied to instance ontology **planA.daml**: 954 RDF statements, 195 subjects
- Ontology fragment:

```
<rdf:Description rdf:about='G. SMILAX'>
  <rdf:type rdf:resource='http://www.dso.org.sg/PlanOntology#AxisOfAdvance'/>
</rdf:Description>
<rdf:Description rdf:about='InfantryBattalion_aa5'>
  <rdf:type rdf:resource='http://www.dso.org.sg/PlanOntology#InfantryBattalion'/>
</rdf:Description>
```

$G\_SMILAX : Resource$

⟨⟨grule G_SMILAX_type⟩⟩
$G\_SMILAX \in instances(AxisOfAdvance)$

$InfantryBattalion\_aa5 : Resource$

⟨⟨grule InfantryBattalion_aa5_type⟩⟩
$InfantryBattalion\_aa5 \in instances(InfantryBattalion)$

- 28 type errors discovered by Z/EVES: mostly caused by re-definition
- No ontological errors found by RACER

More Advanced Reasoning (continued)

- Use domain-specific theorems to systematically test the consistency of the ontology
- E.g., *"no military task should be the sub task of itself and its start time should be less than or equal to its end time"*.
- Once a goal cannot be proved: negate the theorem and prove
- 14 ***hidden errors*** found by Z/EVES in step 4
  - 2: military task's start time greater than end time
  - 4: military task doesn't have end time defined
  - 3: military unit assigned to different tasks simultaneously
  - 5: military tasks with more than one start or end time point

Local Consistency

- *"No military task should be the sub task of itself and its start time should be less than or equal to its end time"* – local consistency of military tasks

theorem MilitaryTaskTimeSubTaskTest1
$\forall x : instances(MilitaryTask) \bullet$
$start(x) < end(x) \land x \notin (sub\_val(subTaskOf))(\!|\ \{x\}\ |\!)$

- Systematically test all instances of `MilitaryTask`
- Example: the remaining goal of one inconsistent example:`ECA_P3_P3_S1`

$\neg\ x = ECA\_P3\_P3\_S1$

- Apparent contradiction: negate the theorem & prove again

theorem negatedMilitaryTaskTimeSubTaskTest1
$\exists x : instances(MilitaryTask) \bullet$
$\neg\ (start(x) < end(x) \land x \notin (sub\_val\ (subTaskOf))(\!|\ \{x\}\ |\!))$

Temporal Relationships Retween Tasks

- *"Sub tasks's duration must be within its super tasks' durations"*

theorem subTaskOfTimingTest2
$\forall x : instances(MilitaryTask) \bullet$
$\forall y : \mathbb{P}(instances(MilitaryTask)) \mid$
$y = (sub\_val(subTaskOf))(\!|\ \{x\}\ |\!) \bullet$
$\forall z : y \bullet start(z) \leq start(x) \land end(z) \geq end(x)$

- y is the set of super tasks of x, z is any member of y
- Local consistency ensured by the previous theorem, hence $start(z) \leq start(x) \land end(z) \geq end(x)$ is sufficient

Military tasks & units

- *"No military task is to be assigned to 2 different tasks at the same time"*

$$\begin{array}{l}\textbf{theorem}\ \text{MilitaryUnitTest}\\ \quad \forall x : instances(ModernMilitaryUnit) \bullet \forall y, z : instances(MilitaryTask) \mid\\ \qquad x \in (sub\_val(assignedTo))(\!\mid \{y\} \mid\!) \wedge x \in (sub\_val(assignedTo))(\!\mid \{z\} \mid\!) \bullet\\ \qquad\quad end(y) \leq start(z) \vee end(z) \leq start(y)\end{array}$$

- Since local consistency has been ensured for each military task, predicate $end(y) \leq start(z) \vee end(z) \leq start(y)$ is sufficient
- Example: the remaining goal for military tasks `ECA_P3_P5_S1` & `ECA_P3_P5_S3` and military unit `CHF_1`

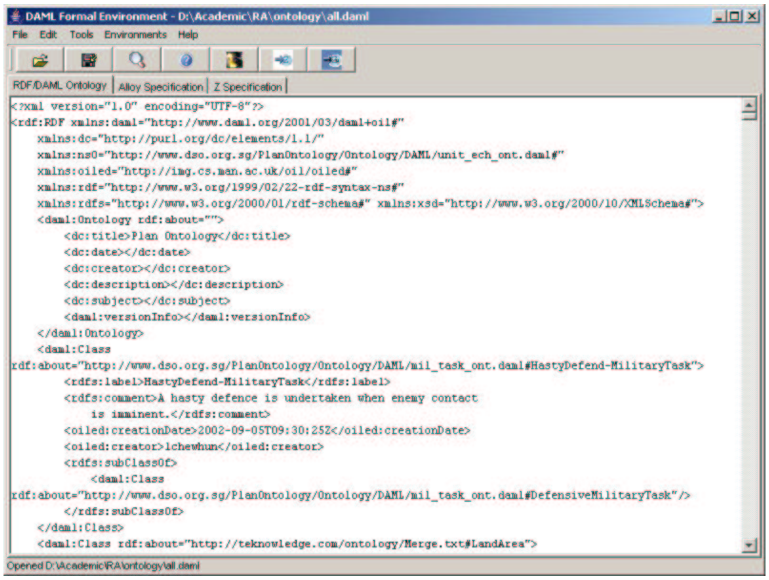
$$\begin{array}{l}z = ECA\_P3\_P5\_S1 \wedge y = ECA\_P3\_P5\_S3\\ \Rightarrow \neg\ x = CHF\_1\end{array}$$

- An obvious contradiction, negate the theorem & prove again
- 3 such errors were found

Summary of the Combined Approach

- The combination of SW & SE reasoning tools effectively checks ontology-related properties
- Results of the synergy
  - Automatedly find ontological inconsistencies using RACER
  - Isolate & find the source of the inconsistencies using Alloy Analyser
  - Interactively checks for more complex properties (inexpressible in DAML+OIL) using Z/EVES
- Application to the second military-domain case study revealed 1 ontological inconsistency & 14 *hidden* errors

Tool Environment for the Combined Approach (on going)



Conclusion

- Semantic Web
  - ✓ good support for automation, collaboration, extension and integration
  - × less expressive and no systematic design process for web ontology/agents
- Software 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 Using Software Tools (FME'03, ICSE'04, WWW'04)

Possible Future Research

- Software Engineering for Semantic Web:
  - Software specification languages (like Z) as Semantic Web languages
  - Web Services (OWL-S) Specifications
  - Model behaviors of intelligent Semantic Web agents using Z, process algebra or integrated formal methods
- Semantic Web for Software Engineering
  - Meta integrating environment for software modeling
  - Intelligent Software Engineering Environment

Recent Publications

The research on Formal methods and Semantic Web has been investigated in [8, 9, 7, 12, 6, 4, 5]. The research on UML and Semantic Web has been investigated in [3, 11, 1, 2, 10].

References

[1] K. Baclawski, M. Kokar, P. Kogut, L. Hart, J. Smith, W. Holmes, J. Letkowski, and M. Aronson. Extending UML to Support Ontology Engineering for the Semantic Web. In M. Gogolla and C. Kobryn, editors, *UML'01*, Lect. Notes in Comput. Sci. Springer-Verlag, 2001.

[2] S. Cranefield. Networked knowledge representation and exchange using uml and rdf. *Journal of Digital Information*, 1(8), 2001.

[3] S. Cranefield and M. Purvis. Uml as an ontology modeling language. In *IJCAI Workshop of Intelligent Information Integration*, 1999.

[4] J. S. Dong, C. H. Lee, H. B. Lee, Y. F. Li, and H. Wang. A Combined Approach to Checking Web Ontology. In *The 13th International World Wide Web Conference (WWW'04), refereed papers track*. ACM Press, May 2004.

[5] J. S. Dong, C. H .Lee, Y. F. Li, and H. Wang. Verifying DAML+OIL and Beyond in Z/EVES. In *The 26th International Conference on Software Engineering (ICSE'04)*. IEEE Press, May 2004.

[6] J. S. Dong, J. Sun, and H. Wang. Semantic Web for Extending and Linking Formalisms. In L.-H. Eriksson and P. A. Lindsay, editors, *Proceedings of Formal Methods Europe: FME'02*, pages 587–606, Copenhagen, Denmark, July 2002. LNCS, Springer-Verlag.

[7] J. S. Dong, J. Sun, and H. Wang. Z Approach to Semantic Web. In C. George and H. Miao, editors, *International Conference on Formal Engineering Methods (ICFEM'02)*, pages 156–167. LNCS, Springer-Verlag, October 2002.

[8] J. S. Dong, J. Sun, and H. Wang. Checking and Reasoning about Semantic Web through Alloy. In *Proceedings of 12th Internation Symposium on Formal Methods Europe: FM'03*, pages 796–813, Pisa, Italy, September 2003. LNCS, Springer-Verlag.

[9] J. S. Dong, J. Sun, H. Wang, C. H. Lee, and H. B. Lee. Analysing Semantic Web: A Military Case Study. In *The 15th International Conference on Software Engineering and Knowledge Engineering (SEKE'03)*, San Francisco, USA, June 2003.

[10] K. Falkovych, M. Sabou, and H. Stuckenschmidt. Uml for the semantic web: Transformation-based approaches, 2003. in B. Omelayenko and M. Klein: Knowledge Transformation for the Semantic Web, IOS Press.

[11] P. Kogut, S. Cranefield, L. Hart, M. Dutra, K. Baclawski, M. Kokar, and J. Smith. UML for Ontology Development. *Knowledge Engineering Review*, 17, 2002.

[12] Hai Wang. *Semantic Web and Formal Design Methods.* PhD thesis, National University of Sinagpore, 2004. (to appear).