Software Engineering Techniques for Semantic Web

Jin Song DONG

(www.comp.nus.edu.sg/∼dongjs)

Computer Science Department

National University of Singapore

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

June 2005

Objectives

- To learn Software Modeling Techniques, i.e., Z, Alloy ...
- To learn Semantic Web Languages, i.e., RDF, OWL ...
- To study the Connections Between the Two Areas.

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)

Overview

- Introduction to Software Modeling Techniques
  - Z and Alloy
- Introduction to Semantic Web
  - RDF, DAML+OIL, OWL and ORL
- 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
- Conclusion and Further Work

The Z Specification Language

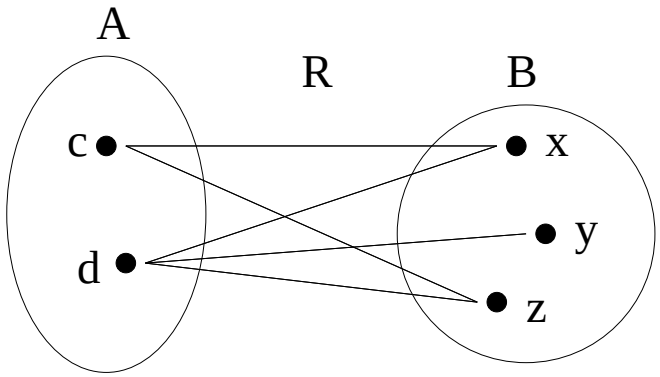
- developed originally at Programming Research Group, Oxford University
- based on set theory and predicate logic
- system described by introducing fixed sets and variables and specifying the relationships between them using predicates
- declarative, not procedural
- system state determined by values taken by variables subject to restrictions imposed by state invariant
- operations expressed by relationship between values of variables before, and values after, the operation
- variable declarations and related predicates encapsulated into schemas
- schema calculus facilitates the composition of complex specifications
- J. Woodcock and J. Davies, Using Z: Specification, Refinement, and Proof. Prentice-Hall, 1996

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.

$$\begin{array}{lll} \operatorname{dom} R & \text{is the set} & \{a : A \mid \exists\, b : B \bullet a \,\underline{R}\, b\} \\ \operatorname{ran} R & \text{is the set} & \{b : B \mid \exists\, a : A \bullet a \,\underline{R}\, b\} \end{array}$$

Domain and Range Restriction/Subtraction

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

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

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

e.g. if

$$has\_sibling : People \leftrightarrow People \quad \text{then}$$

$$\begin{array}{lll} female \lhd has\_sibling & \text{is the relation} & is\_sister\_of \\ has\_sibling \rhd female & \text{is the relation} & has\_sister \end{array}$$

$$\begin{array}{lll} female \lhd\!\!\!\!- has\_sibling & \text{is the relation} & is\_brother\_of \\ has\_sibling \rhd\!\!\!\!- female & \text{is the relation} & has\_brother \end{array}$$

Relational Image

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

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

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

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

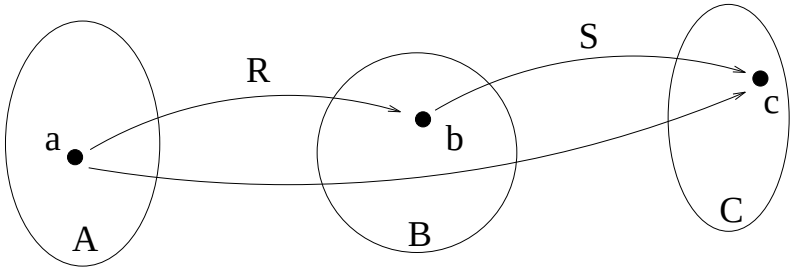
$$\leqslant (\!| \{7, 3, 21\} |\!) = \{x : \mathbb{N} \mid x \geqslant 3\}$$

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

Relational Composition

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

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



$R \fatsemi S \in A \leftrightarrow C$

e.g.

$$is\_parent\_of \fatsemi is\_parent\_of = is\_grandparent\_of$$

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

Functions

A (partial) function f from a set A to a set B, denoted by

$$f : A \nrightarrow B,$$

is a subset f of $A \times B$ with the property that for each $a \in A$ there is at most one $b \in B$ with $(a, b) \in f$. The function f is a *total* function, denoted

$$f : A \rightarrow B,$$

if and only if $\mathrm{dom}\, f$ is the set A.

The predicates

$$(a, b) \in f \quad \text{and} \quad f(a) = b$$

are equivalent.

Sequences

A sequence s of elements from a set A, denoted

$$s : \operatorname{seq} A,$$

is a function $s : \mathbb{N} \pfun A$ where $\operatorname{dom} s = 1 \upto n$ for some natural number n. For example,

$\langle b, a, c, b \rangle$ denotes the sequence (function) $\{1 \mapsto b, 2 \mapsto a, 3 \mapsto c, 4 \mapsto b\}$

The empty sequence is denoted by $\langle\ \rangle$.

The set of all sequences of elements from A is denoted $\operatorname{seq} A$ and is defined to be

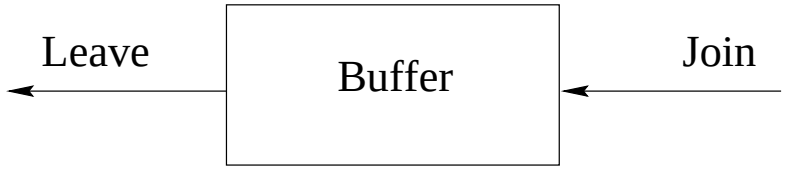
$$\operatorname{seq} A == \{s : \mathbb{N} \pfun A \mid \exists n : \mathbb{N} \bullet \operatorname{dom} s = 1 \upto n\}$$

We define $\operatorname{seq}_1 A$ to be the set of all non-empty sequences, i.e.

$$\operatorname{seq}_1 A == \operatorname{seq} A - \{\langle\ \rangle\}$$

Notice that: $\langle a, b, a \rangle \neq \langle a, a, b \rangle \neq \langle a, b \rangle$

Z Schemas: A Message Buffer Example



- A number of messages are transmitted from one location to another.
- Because of other traffic on the line each message for transmission is placed in a buffer which outputs the message when the line is free.
- This buffer may contain several messages at any time, but there is a fixed upper limit on the number of messages the buffer may contain.
- The buffer operates on a first in/first out (FIFO) principle.

Formal Specification

The State Schema

$[MSG]$ (The exact nature of these messages is not important)

is the set of all possible messages that could ever be transmitted.

$max : \mathbb{N}$ (The actual value of max is not important)

is the constant maximum number of messages that can be held in the buffer at any one time.

Buffer

$items : \operatorname{seq} MSG$ — declaration

$\#items \leqslant max$ — predicate

e.g. suppose $MSG = \{m_1, m_2, m_3\}$ and $max = 4$

Then $items = \langle m_1, m_2 \rangle$ is an instance, but $items = \langle m_3, m_1, m_1, m_2, m_2 \rangle$ is not

Schema Inclusion and Operation/Initial Schemas

$\Delta Buffer$

$Buffer$

$Buffer'$

$\Delta Buffer$

$items, items' : \operatorname{seq} MSG$

$\#items \leqslant max \land \#items' \leqslant max$

Join

$\Delta Buffer$

$msg? : MSG$

$\#items < max$

$items' = items \frown \langle msg? \rangle$

Leave

$\Delta Buffer$

$msg! : MSG$

$items \neq \varnothing$

$items = \langle msg! \rangle \frown items'$

$Buffer_{\text{INIT}}$

$Buffer$

$items = \langle\ \rangle$

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 Logical Operators

```
!F // negation: not F
F && G // conjunction: F and G
F || G // disjunction: F or G
F => G // implication: F implies G; same as !F || G
F <=> G // biimplication: F when G; same as F =>G && G => F
F => G,H // if F then G else H; same as F => G && !F => H
```

Quantifiers

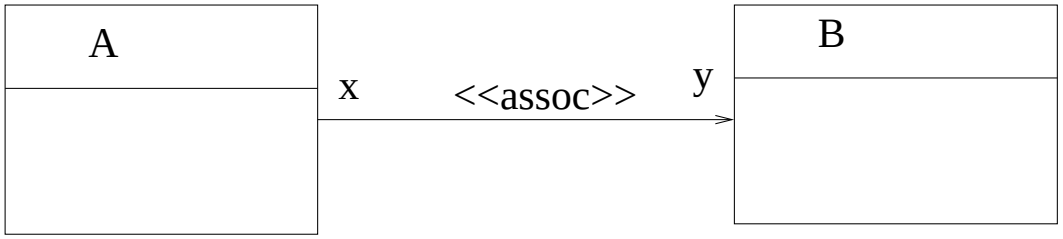
```
all x: e | F
some x: e | F
no x: e | F
sole x: e | F
one x: e | F
one x:e, y:f | F
all disj x,y: e | F
```

Examples

```
// no polygamy
all p: Person | sole p.spouse
// a married person is his or her spouse's spouse
all p: Person | some p.spouse => p.spouse.spouse = p
// no incest
no p: Person | some (p.spouse.parents & p.parents)
// a person's siblings are those persons with the same parents
all p: Person | p.siblings = {q: Person | q.parents = p.parents} - p
// everybody has one mother
all p: Person | one p.parents & Woman
// somebody is everybodys ancestor
some x: Person | all p: Person | x in p.*parent
```

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
```

Module, Sig, Fact, Fun and Assert (example)

```
module CeilingsAndFloors
sig Platform {}
sig Man {ceiling, floor: Platform}
fact {all m: Man | some n: Man | Above (n,m)}
fun Above (m, n: Man) {m.floor = n.ceiling}
assert BelowToo {all m: Man | some n: Man | Above (m,n)}
run Above for 2
check BelowToo for 2
```

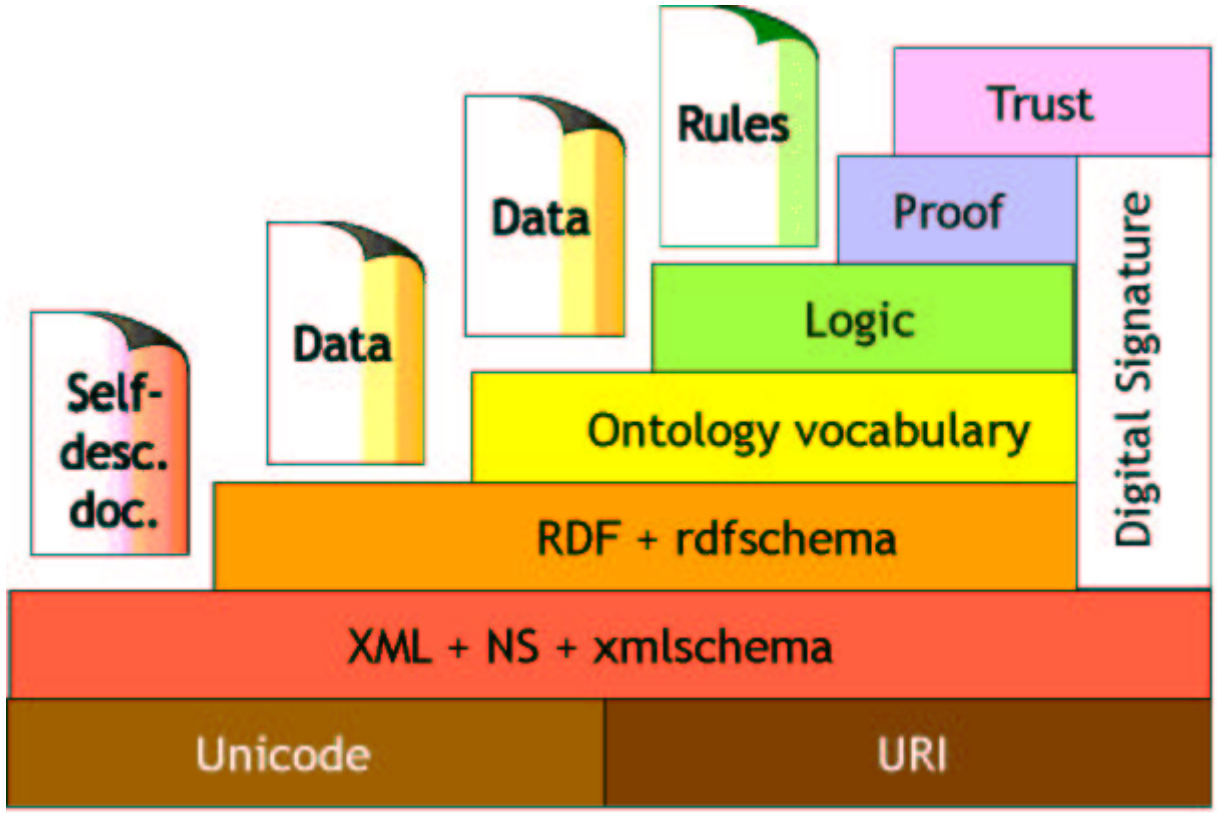
Recall Overview

- Introduction to Software Modeling Techniques
  - Z and Alloy
- ✓ Introduction to Semantic Web
  - RDF, DAML+OIL, OWL and ORL
- 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
- 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 combines Ontology Interchange Language (OIL).
- OWL Web Ontology Language — 2003 (become W3C rec)
  - 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

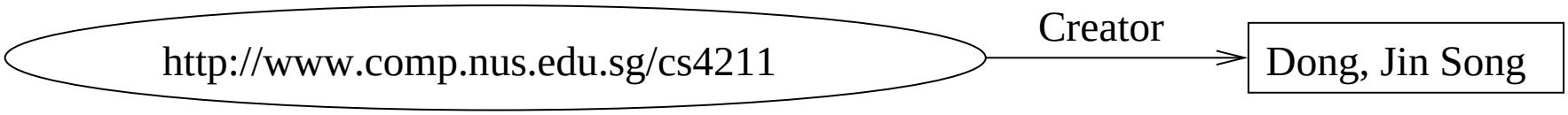
RDF Basics

- 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

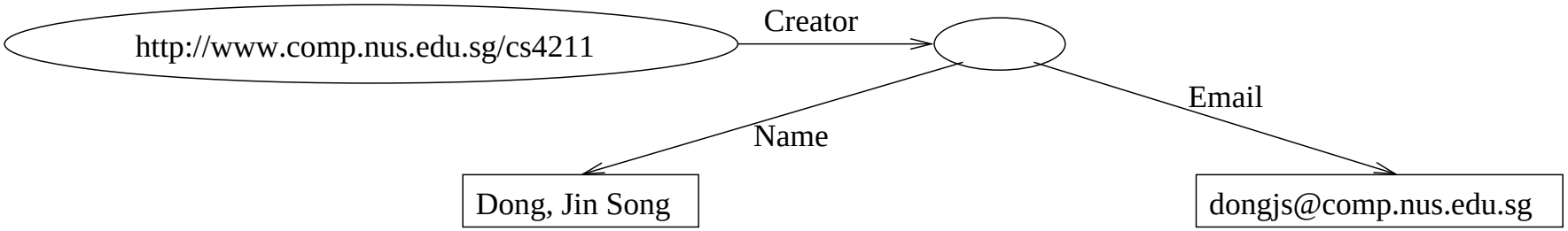
Dong, Jin Song is the creator of the web page
`http://www.comp.nus.edu.sg/cs4211`

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

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

RDF Schema Example (Property)

```
<rdf:Property rdf:ID="teacher">
       <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: 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: 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>
   <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/`

Beyond OWL: Ontology Rule Language (ORL)

- Decidability vs Expressiveness
- OWL is weak in express composite properties
- ORL extends OWL DL with a form of rules while maintaining compatibility with OWLs existing syntax and semantics.
- I. Horrocks and P. F. Patel-Schneider, A Proposal for an OWL Rules Language, ACM WWW'04, NY, May 2004

ORL Example

```
<owlx:Rule>
  <owlx:antecedent>
    <owlx:individualPropertyAtom  owlx:property="hasParent">
      <owlx:Variable owlx:name="x1" />
      <owlx:Variable owlx:name="x2" />
    </owlx:individualPropertyAtom>
    <owlx:individualPropertyAtom  owlx:property="hasBrother">
      <owlx:Variable owlx:name="x2" />
      <owlx:Variable owlx:name="x3" />
    </owlx:individualPropertyAtom>
  </owlx:antecedent>
  <owlx:consequent>
    <owlx:individualPropertyAtom  owlx:property="hasUncle">
      <owlx:Variable owlx:name="x1" />
      <owlx:Variable owlx:name="x3" />
    </owlx:individualPropertyAtom>
  </owlx:consequent>
</owlx:Rule>
```

Recall Overview

- Introduction to Software Modeling Techniques
  - Z and Alloy
- Introduction to Semantic Web
  - RDF, DAML+OIL, OWL and ORL

✓ 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

- 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: 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

$$\frac{S \;\; X : T_1;\quad Y : \mathbb{P}\, T_2}{T_1, T_2 \in daml\_class}$$

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

$$\frac{Paper \;\; title : Title;\quad authors : \mathbb{P}\, Author}{}$$

```
<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)

$$\frac{R : B \leftrightarrow (\rightarrow, \twoheadrightarrow) C \qquad B, C \in daml\_class}{\ldots}$$

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

$$reference : Paper \leftrightarrow Paper$$

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

Z axiomatic definition transformation (subset)

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

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

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

Exercise: Convert Z spec to DAML

$$[Students, Code, Title]$$

$$\begin{array}{|l}
\hline
Course \\
\hline
code : Code \\
title : Title \\
\hline
\end{array}
\quad
\begin{array}{|l}
GraduateCourse : \mathbb{P}\, Course \\
enrolment : Students \leftrightarrow Course
\end{array}$$

Convert the Z spec to DAML:

```
<daml:class rdf:ID="student"> <rdfs:label>Student</rdfs:label> </daml:Class>
<daml:class rdf:ID="code"> <rdfs:label>Code</rdfs:label> </daml:Class>
<daml:class rdf:ID="title"> <rdfs:label>Title</rdfs:label> </daml:Class>

<daml:class rdf:ID="course"> <rdfs:label>Course</rdfs:label> </daml:Class>
<daml:ObjectProperty rdf:ID="course_code">
<rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <rdf:domain rdf:resource="#course"/> <rdf:range rdf:resource="#code"/>
</daml:ObjectProperty>
<daml:ObjectProperty rdf:ID="course_title">
<rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <rdf:domain rdf:resource="#course"/> <rdf:range rdf:resource="#coursetitle"/>
</daml:ObjectProperty>

<daml:class rdf:ID="graduatecourse">
  <rdfs:subClassOf rdf:resource="#course"/> </daml:class>
<daml:ObjectProperty rdf:ID="enrolment">
  <rdfs:domain rdf:resource="#student"/> <rdfs:range rdf:resource="#course"/>
</daml:ObjectProperty>
```

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.

Checking Military Plan Ontology Experience

- Singapore DSO has developed an IE engine which has been used to generate ontologies (in DAML) from military formation and plan (in natural language).
- A military ontology is made up of the following four main ingredient sets.
  - military operations and tasks, which define the logic order, type , and phases of a military campaign.
  - military units, which are the participants of the military operations and tasks,
  - geographic locations, where such operations take place and
  - 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 Military Case Study Statistics in Z/EVES

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 beyond DAML	2

Recall Overview

- Introduction to Software Modeling Techniques
  - Z and Alloy
- Introduction to Semantic Web
  - RDF, DAML+OIL, OWL and ORL
- 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
- Conclusion and Further Work

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

$$Property : \mathbb{P}\, Resource$$
$$\overline{Class \cap Property = \varnothing}$$

```
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

$$
\begin{array}{|l}
subClassOf : Class \leftrightarrow Class \\
disjointWith : Class \leftrightarrow Class \\
\hline
\forall\, c_1, c_2 : Class \bullet \\
c_1 \ \underline{subClassOf}\ c_2 \Leftrightarrow instances(c_1) \in \mathbb{P}\, instances(c_2) \\
c_1 \ \underline{disjointWith}\ c_2 \Leftrightarrow instances(c_1) \cap instances(c_2) = \varnothing
\end{array}
$$

```
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

$$\begin{array}{|l} toClass : (Class \times Property) \leftrightarrow Class \\ \hline \forall\, c_1, c_2 : Class;\ p : Property \bullet (c_1, p)\ \underline{toClass}\ c_2 \Leftrightarrow \\ (\forall\, a_1, a_2 : Resource \bullet a_1 \in instances(c_1) \Leftrightarrow \\ \qquad ((a_1, a_2) \in sub\_val(p) \Rightarrow a_2 \in instances(c_2))) \end{array}$$

```
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!

$$\begin{array}{|l} Fish, Gill : Class \\ Breathe\_by : Property \\ \hline (Fish, Breathe\_by)\ \underline{toClass}\ Gill \end{array}$$

Z/Alloy Semantics for DAML+OIL

Class & Property (continued)

- hasValue

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

$$\forall\, c : Class;\ p : Property;\ r : Resource \bullet$$
$$(c, p)\ \underline{hasValue}\ r \Leftrightarrow$$
$$(\forall\, a : instances(c) \bullet (a, r) \in sub\_val(p))$$

```
fun hasValue (p:Property, c:Class, r:Resource)
    {all a:Resource |
    a in c.instances => a.(p.sub_val) = r}
```

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}
```

Import Mechanism & Proof Support for Z/EVES

- Import mechanism
  - Z definitions are put into a *section* `daml2z`
  - Alloy definitions are put into a *module* `DAML`
  - Other transformed ontologies have these definitions as parents
- Proof support for Z/EVES
  - Definitions alone are not adequate
  - Trivial proof goals should be automated
  - A *section* `DAML2ZRules` of *rewrite*, *assumption* & *forward* rules are constructed

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 $\mathbb{N}$
    - Domai n-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:

$$\frac{MilitaryTask : Class}{\langle\!\langle \text{grule} \;\; \text{MilitaryTask\_subClassOf\_MilitaryProcess} \rangle\!\rangle \\ (MilitaryTask, MilitaryProcess) \in subClassOf}$$

- Alloy:

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

Recall Overview

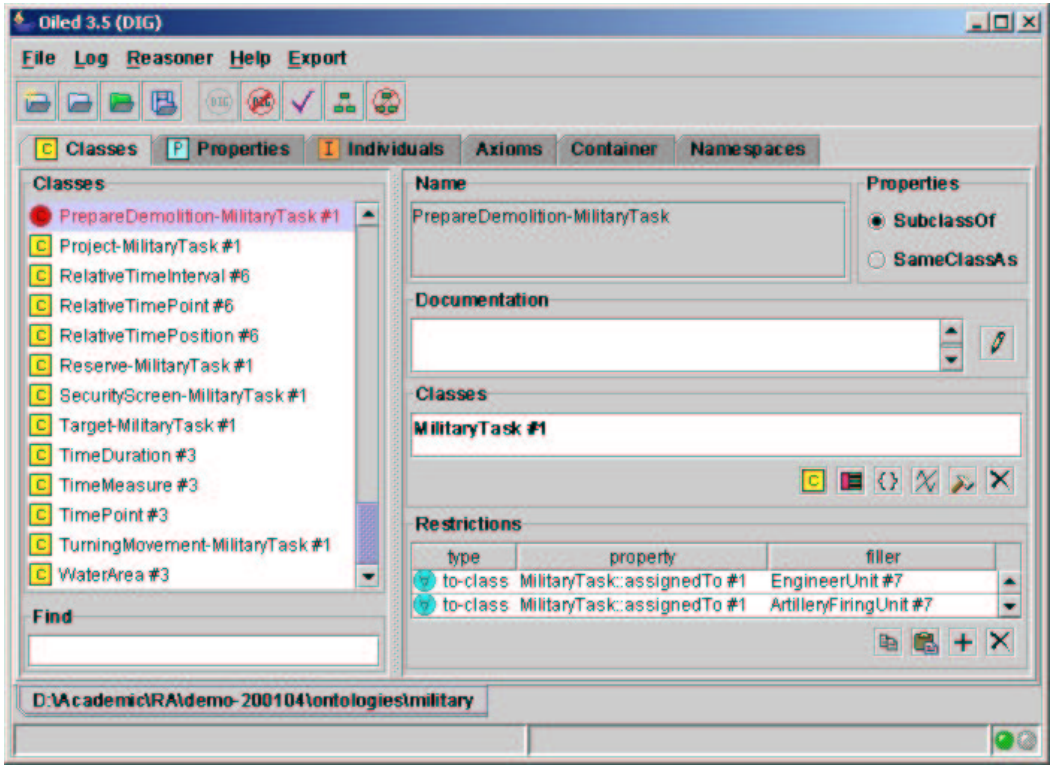
- Introduction to Software Modeling Techniques
  - Z and Alloy
- Introduction to Semantic Web
  - RDF, DAML+OIL, OWL and ORL
- Software Design Method/Tools for Semantic Web
  - Extracting DAML ontology from Z models
  - Semantics of DAML+OIL in Z/Alloy
  - ✓ **The Combined Approach**
- Conclusion and Further Work

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...

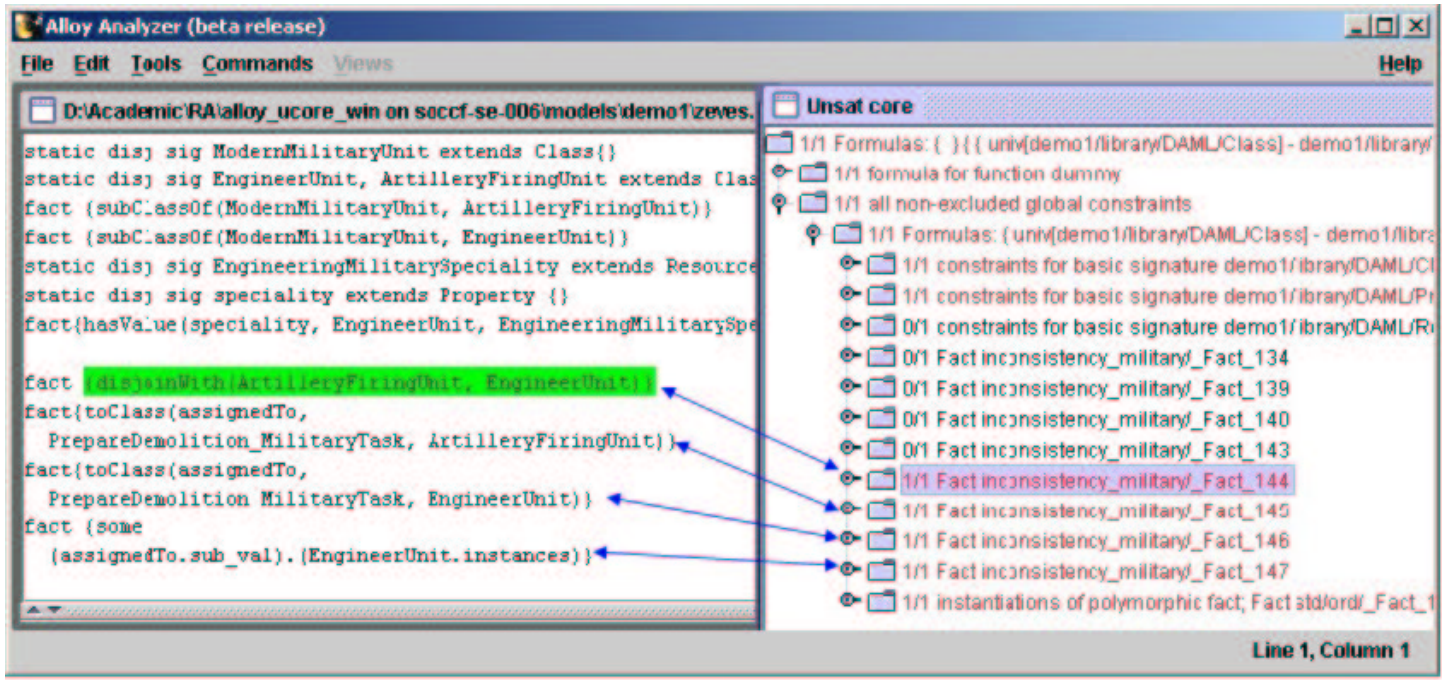
Standard SW Reasoning

- 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>
```

$$\frac{G\_SMILAX : Resource}{\langle\!\langle \text{grule G\_SMILAX\_type} \rangle\!\rangle \quad G\_SMILAX \in instances(AxisOfAdvance)}$$

$$\frac{InfantryBattalion\_aa5 : Resource}{\langle\!\langle \text{grule InfantryBattalion\_aa5\_type} \rangle\!\rangle \quad InfantryBattalion\_aa5 \in instances(InfantryBattalion)}$$

- 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

Temporal Relationships Retween Tasks

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

$$
\begin{array}{l}
\textbf{theorem}\ \text{subTaskOfTimingTest2} \\
\quad \forall\, x : \mathit{instances}(\mathit{MilitaryTask}) \bullet \\
\quad\quad \forall\, y : \mathbb{P}(\mathit{instances}(\mathit{MilitaryTask})) \mid \\
\quad\quad\quad y = (\mathit{sub\_val}(\mathit{subTaskOf}))(\!|\, \{x\} \,|\!) \bullet \\
\quad\quad\quad \forall\, z : y \bullet \mathit{start}(z) \leq \mathit{start}(x) \land \mathit{end}(z) \geq \mathit{end}(x)
\end{array}
$$

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

Military tasks & units

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

  theorem MilitaryUnitTest
  $\forall\, x : instances(ModernMilitaryUnit) \bullet \forall\, y, z : instances(MilitaryTask) \mid$
  $x \in (sub\_val(assignedTo))(\!|\, \{y\} \,|\!) \land x \in (sub\_val(assignedTo))(\!|\, \{z\} \,|\!) \bullet$
  $end(y) \leq start(z) \lor end(z) \leq start(y)$

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

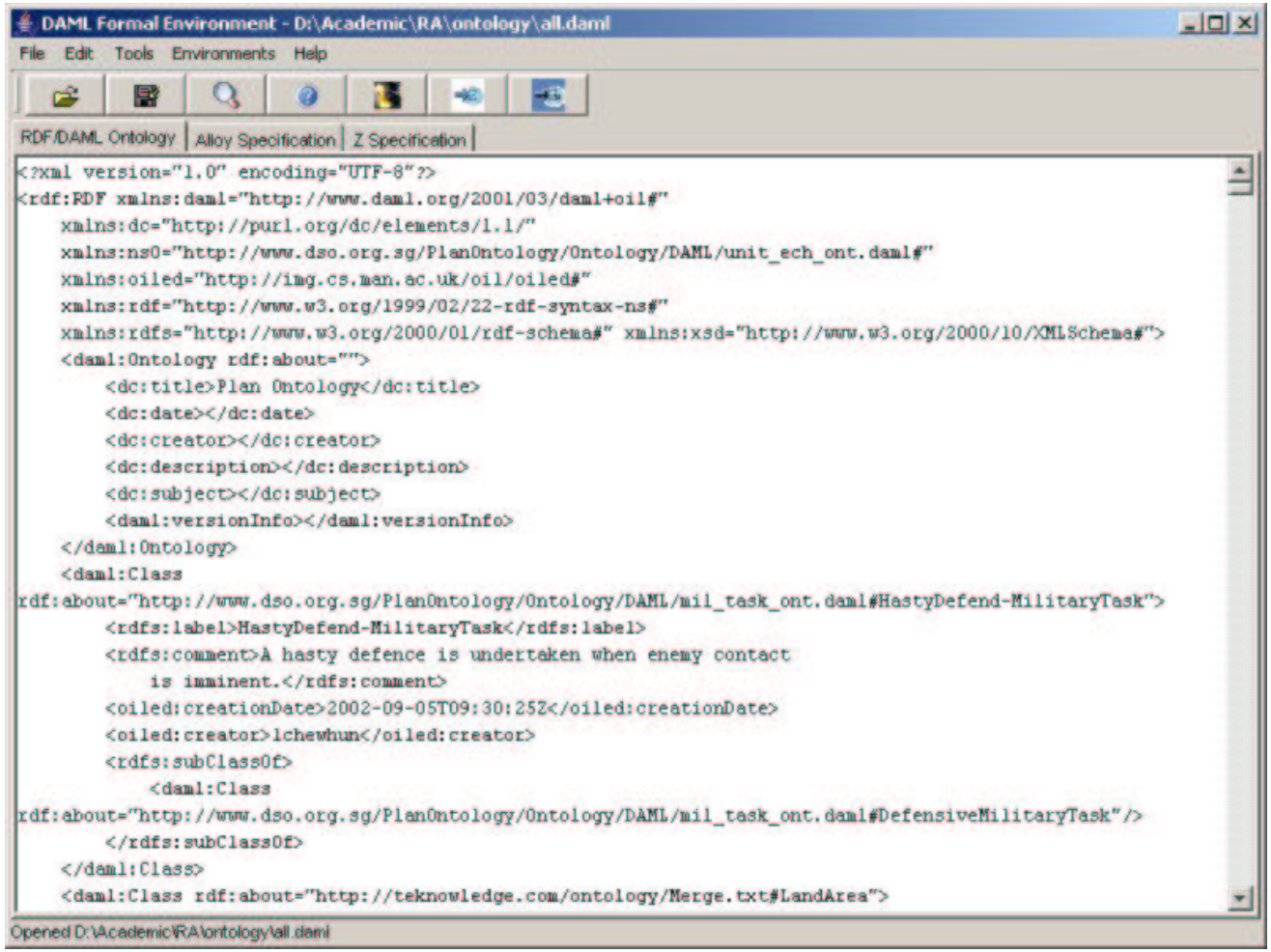
  $z = ECA\_P3\_P5\_S1 \land y = ECA\_P3\_P5\_S3$
  $\Rightarrow \neg\, x = CHF\_1$

- 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)



Tutorial 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 [5, 6, 4, 7, 3, 1, 2].

References

[1] 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.

[2] 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)*. ACM/IEEE Press, May 2004.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] Hai Wang. *Semantic Web and Formal Design Methods*. PhD thesis, National University of Sinagpore, 2004.