Some Formal Methods Research at NUS

Jin Song Dong

Computer Science Department

National University of Singapore

July 2004

Some Research Topics in Our Group

- Synthesize Event-based Controllers for State-based Plant
- Integrated Specification based testing and monitoring
- Design and Varification of Autonomous Multi-agent Systems
- Formal Approaches to Context-Aware Computing
- Z and Timed Automata
  - ✓ Timed Patterns: TCOZ to Timed Automata (projection)
  - – Object-Z and Timed Automata (integration)
- Formal methods and Semantic Web
  - ✓ Apply Formal Methods to Semantic Web
  - – Semantic Web environment for linking various formalisms

Timed Patterns: TCOZ to Timed Automata

(joint work with P. Hao, J. Sun, SC. Qin and Y. Wang)

Object-Z and Timed CSP

- Object-Z
  - ✓ an excellent tool for modeling data states
  - × but difficult for modelling real-time concurrent systems
- Timed CSP
  - ✓ Good for specifying the timed process and communication
  - × Like CSP, cumbersome to capture the data state of a complex system
- Timed Communicating Object Z *(IEEE TSE 2000)*
  - * extension to timed failures semantics *(IFM'99)*
  - * UTP semantics *(FM'03)*

Related Work

* Z/OZ with CSP: Fischer, Smith, Derick, Suhl, Bolton, Davies, Woodcock ...

Timed Communicating Object Z (TCOZ)

TimedBuffer[X]

$items : \operatorname{seq} X$

$left, right :$ **chan**

INIT

$items = \langle\,\rangle$

Add

$\Delta(items);\ i? : X$

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

Remove

$\Delta(items)$

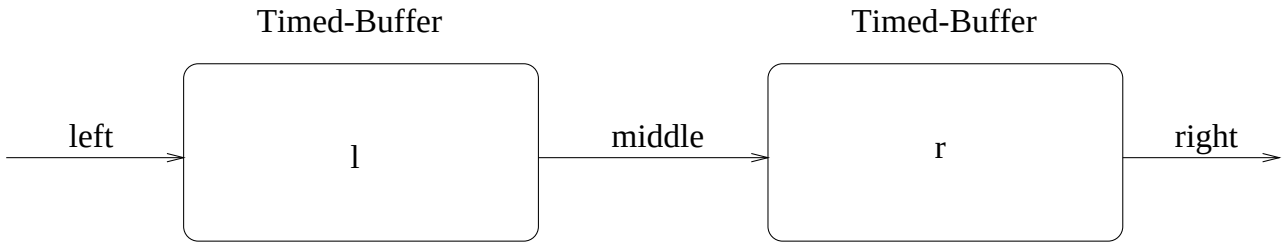
$items' = items \rhd \{last(items)\}$

$Join \mathrel{\widehat=} [i : X] \bullet left?i \rightarrow Add;$ DEADLINE tj

$Leave \mathrel{\widehat=} right!last(items) \rightarrow Remove;$ DEADLINE tl

$\text{MAIN} \mathrel{\widehat=} \mu\, T \bullet [items = \langle\,\rangle] \bullet Join;\ T \,\Box$
$[items \neq \langle\,\rangle] \bullet ((Join \,\Box\, Leave) \triangleright\{t_o\}\ Remove);\ T$

Two Communicating Timed Buffers



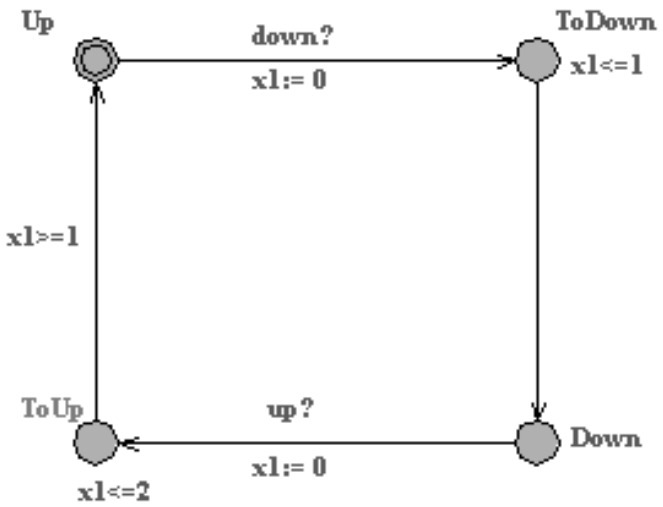
$TwoBuffers[X]$

$l : TimedBuffer[X][middle/right]$
$r : TimedBuffer[X][middle/left]$

$\textsc{Main} \mathrel{\widehat{=}} (l \,|[\, middle \,]|\, r \setminus middle)$

Timed Automata (TA)

Timed Automata are finite state machines with clocks. It can be used to model the behavior of real-time systems. Its definition provides a general way to annotate state-transition graphs with timing constraints using clock variables.



R. Alur and D.L.Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

TCOZ and Timed Automata — Two Ends of the Spectrum

- TCOZ is well suited for presenting more complete and coherent requirement models for complex systems. However, the challenge is how to check the TCOZ models with tool support, especially for analyzing timing properties.
- Timed Automata (TA) uses multiple clocks for designing real-time models and has well developed automatic tool support. One weakness of TA is the lack of high level composable graphical patterns to support the systematic design for complex systems.
- The investigation of the links between TCOZ and TA may benefit both techniques.
  - for TCOZ, TA's tool support can be reused to check timing properties.
  - for TA, a possible set of composable graphical patterns can be defined based on the semantics of the TCOZ constructs.

TA in Z

$$[State, Event, Clock]$$

$$\begin{aligned}\Phi ::= &(\_ \leq \_)\langle\!\langle Clock \times \mathbb{Z} \rangle\!\rangle \mid (\_ \geq \_)\langle\!\langle Clock \times \mathbb{Z} \rangle\!\rangle \mid \\ &(\_ < \_)\langle\!\langle Clock \times \mathbb{Z} \rangle\!\rangle \mid (\_ > \_)\langle\!\langle Clock \times \mathbb{Z} \rangle\!\rangle \mid \\ &(\_ \wedge \_)\langle\!\langle \Phi \times \Phi \rangle\!\rangle \mid true\end{aligned}$$

$$Label \mathrel{\widehat=} \mathbb{P}\, Event \times \mathbb{P}\, Clock \times \Phi$$

$$Transition \mathrel{\widehat=} State \times Label \times State$$

$\mathcal{S}_{TA}$

$$S : \mathbb{P}\, State; \quad i, e : State$$
$$I : State \pfun \Phi$$
$$T : \mathbb{P}\, Transition$$

$$i, e \in S \wedge \operatorname{dom} I = S$$
$$\forall\, s, s' : state;\ l : label \bullet (s, l, s') \in T \Rightarrow s, s' \in S$$

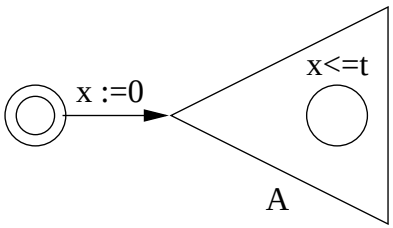
Timed Patterns: TCOZ to Timed Automata

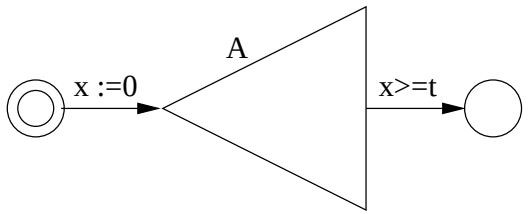
$$deadline : \mathcal{S}_{TA} \times \mathbb{T} \to \mathcal{S}_{TA}$$

$$\begin{aligned}
&\forall A : \mathcal{S}_{TA};\ t : \mathbb{T} \bullet \\
&\quad \exists x : Clock;\ i_0 : State \bullet \\
&\quad\ deadline(A, t) = (\!| \\
&\quad\ S \mathrel{\widehat=} A.S \cup \{i_0\}, \\
&\quad\ i \mathrel{\widehat=} i_0, e \mathrel{\widehat=} A.e, \\
&\quad\ I \mathrel{\widehat=} \{s : A.S \bullet (s, x <= t \land A.I(s))\}, \\
&\quad\ T \mathrel{\widehat=} A.T \cup \{(i, (\tau, \{x\}, true), A.i)\} \ |\!)
\end{aligned}$$

$$waituntil : \mathcal{S}_{TA} \times \mathbb{T} \to \mathcal{S}_{TA}$$

$$\begin{aligned}
&\forall A : \mathcal{S}_{TA};\ t : \mathbb{T} \bullet \\
&\quad \exists x : Clock;\ i_0, e_0 : State \bullet \\
&\quad\ waituntil(A, t) = (\!| \\
&\quad\ S \mathrel{\widehat=} A.S \cup \{i_0, e_0\}, \\
&\quad\ i \mathrel{\widehat=} i_0, e \mathrel{\widehat=} e_0, I \mathrel{\widehat=} A.I, \\
&\quad\ T \mathrel{\widehat=} A.T \cup \{(A.e, (\tau, \varnothing, x \geq t), e), \\
&\qquad\ (i, (\tau, \{x\}, true), A.i)\} \ |\!)
\end{aligned}$$



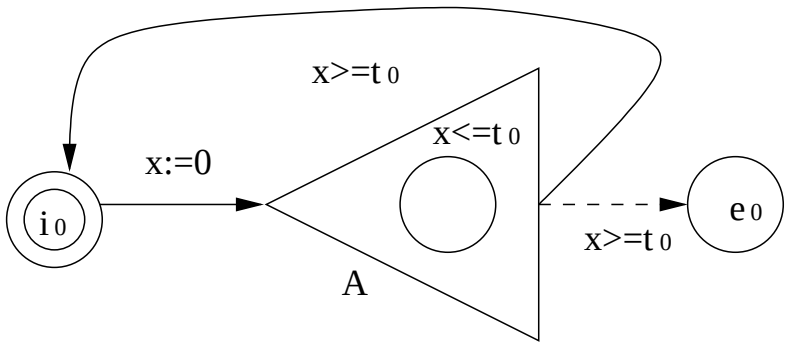


Other patterns: *recursion*, *seqcom*, *timeout*, *interrupt* ...

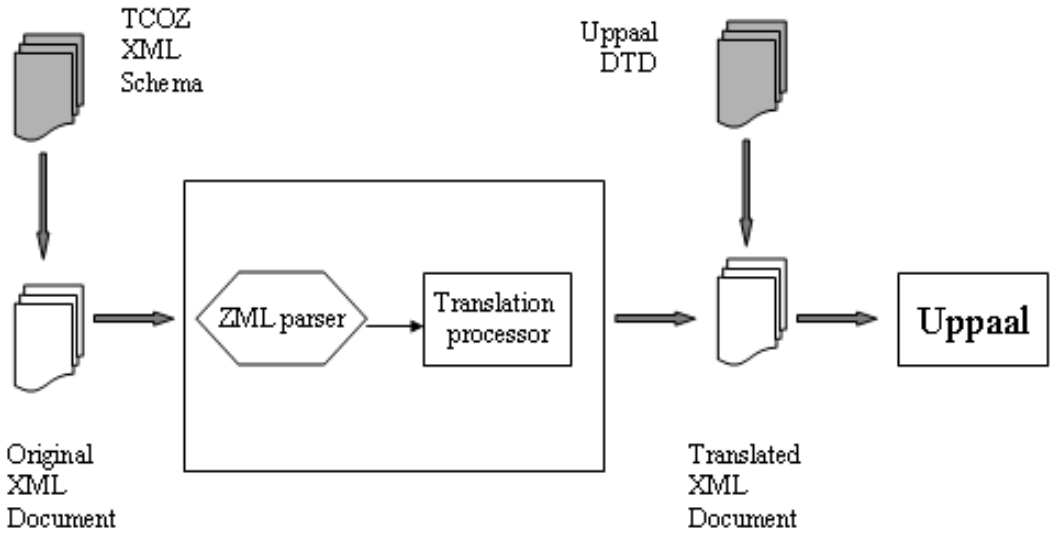
Composing Timed Patterns

$$periodic\_repeat : \mathcal{S}_{TA} \times \mathbb{T} \rightarrow \mathcal{S}_{TA}$$

$$\forall A, A_0 : \mathcal{S}_{TA};\ t_0 : \mathbb{T};\ e_0 : State \bullet$$
$$e_0 = A_0.e \land A_0 = waituntil(deadline(A, t_0), t_0) \Rightarrow$$
$$periodic\_repeat(A, t_0) = recursion(waituntil(deadline(A, t_0), t_0), e_0)$$



TCOZ to TA Tool Uppaal



Railroad Crossing Example in TCOZ

$GateS ::= ToUp \mid Up \mid ToDn \mid Down$

$LightS ::= Green \mid Red$

$TrainS ::= ToIn \mid In \mid Out$

$ControlS ::= TrIn \mid TrOut \mid GtClose \mid GtOpen$

Gate

$status : GateS$

$up, down : \textbf{chan}$

Init

$status = Up$

Lower

$\Delta(status)$

$status' = ToDn$

Down

$\Delta(status)$

$status' = Down$

Raise

$\Delta(status)$

$status' = ToUp$

Up

$\Delta(status)$

$status' = Up$

$Open \mathrel{\widehat=} up \rightarrow (Raise \bullet \textsc{WaitUntil}\ 1;\ Up) \bullet \textsc{Deadline}\ 2$

$Close \mathrel{\widehat=} down \rightarrow Lower \bullet \textsc{Deadline}\ 1;\ Down$

$\textsc{Main} \mathrel{\widehat=} \mu\, G \bullet Close;\ Open;\ G$

Railroad Crossing Example in TCOZ (continue)

Train

$status : TrainS$

$t_{in}, in, out :$ **chan**

Init

$status = Out$

$Approach, Pass, Exit \mathrel{\widehat=} \ldots$

$Train_{in} \mathrel{\widehat=} t_{in} \rightarrow in \rightarrow (Approach \bullet \textsc{WaitUntil}\ 3;\ Pass) \bullet \textsc{Deadline}\ 5$

$Train_{out} \mathrel{\widehat=} out \rightarrow Exit$

$\textsc{Main} \mathrel{\widehat=} \mu\, T \bullet Train_{in};\ Train_{out};\ T$

RCSystem

$t : Train;\ g : Gate;\ l : Light;\ c : Controller$

$\textsc{Main} \mathrel{\widehat=} \| (t \xrightarrow{in,out} c \xrightarrow{up,down} g;\ l \xrightarrow{setlight} c)$

Controller

$status : ControlS$

$up, down :$ **chan**

$in, out, setlight :$ **chan**

Init

$ControlS = TrOut$

$Arrive, Leave, GateClose, GateOpen \mathrel{\widehat=} \ldots$

$Train_{in} \mathrel{\widehat=} in \rightarrow Arrive \bullet \textsc{WaitUntil}\ 1 \bullet \textsc{Deadline}\ 1$

$Train_{out} \mathrel{\widehat=} out \rightarrow Leave \bullet \textsc{Deadline}\ 1$

$Close \mathrel{\widehat=} setlight!Red \rightarrow down \rightarrow GateClose$

$Open \mathrel{\widehat=} up \rightarrow setlight!Green \rightarrow GateOpen$

$\textsc{Main} \mathrel{\widehat=} \mu\, C \bullet Train_{in};\ Close;\ Train_{out};\ Open;\ C$

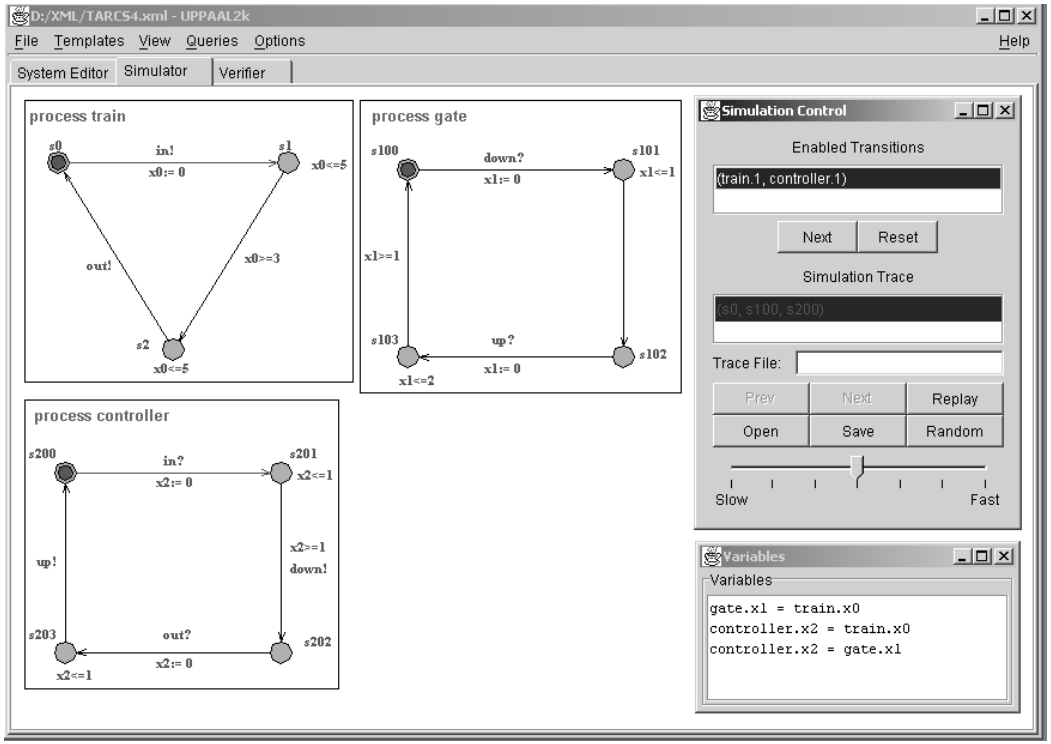
Light

$\ldots$

Model Checking with Uppaal

$$RCSystem \bullet \\ \Box(g.status = ToDn \rightarrow \\ \Diamond_{\leq 10}\ g.status = Up)$$

$$RCSystem \bullet \\ t.status = In \Rightarrow \\ g.status = Down$$



summary and next topic

- Timed Patterns: TCOZ to TA (projection)
  - for TCOZ, TA's tool support can be reused to check timing properties.
    for TA, composable high level patterns can be formulated as a library.
  - more: ICFEM'04 paper
- Semantic Web and Formal Methods (joint work with YF. Li and H. Wang)
  - ✓ Extracting web ontologies systematically from Z specifications
  - ✓ Checking Semantic Web Using Software Tools
  - Semantic Web environment for linking various formalisms

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).

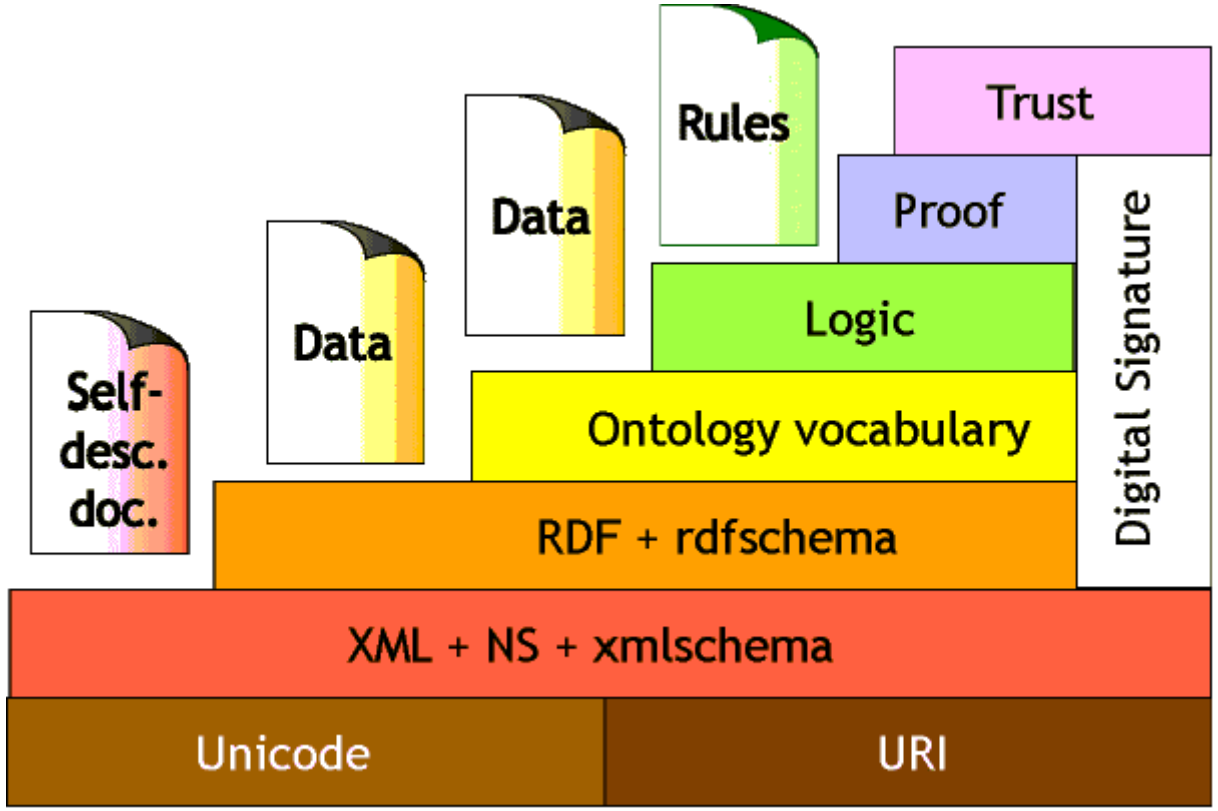
Formal Specification

"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)

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

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: Schema Example

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

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

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

```
<daml:class rdf:ID="paper"> <rdfs:label>Paper</rdfs:label> </daml:Class>
<daml:CbjectProperty 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:CbjectProperty>
<daml:CbjectProperty rdf:ID="paper_authors">
  <rdf:domain rdf:resource="#paper"/>
  <rdf:range rdf:resource="#author"/> </daml:CbjectProperty>
```

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>
```

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 but have weakness, i.e., if inconsistencies found, they cannot show why. They cannot verify complex ontology related properties.

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...

Z/Alloy Semantics for DAML+OIL

Example: 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: Example

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

```
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}
```

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
- `planA.daml`: 954 RDF statements and 195 subjects

More Advanced Reasoning: Beyond SW

Example: Military tasks & units

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

  theorem MilitaryUnitTest

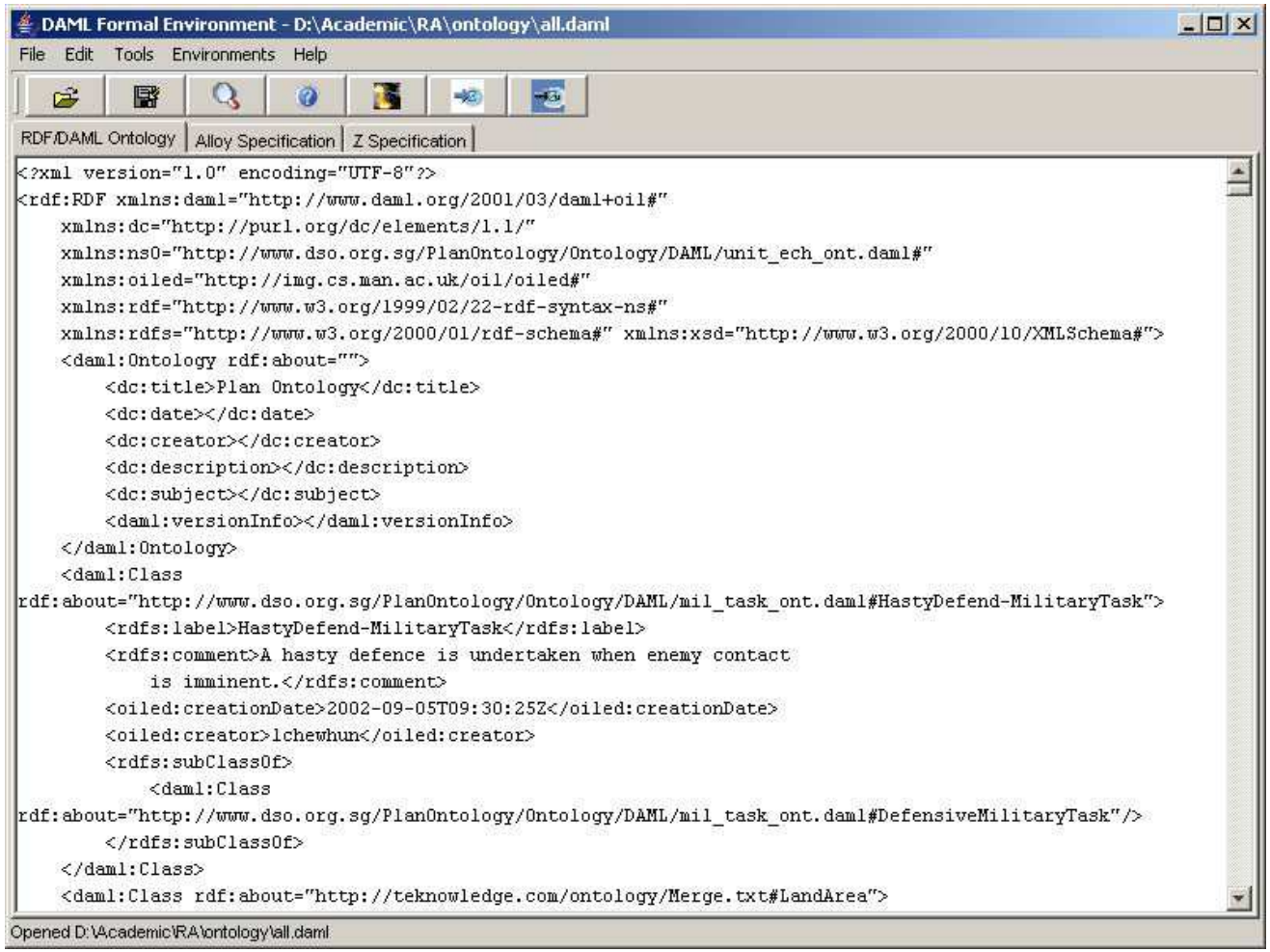
  $$\forall\, x : instances(ModernMilitaryUnit) \bullet \forall\, y, z : instances(MilitaryTask) \mid$$
  $$x \in (sub\_val(assignedTo))(\!|\ \{y\}\ |\!) \wedge x \in (sub\_val(assignedTo))(\!|\ \{z\}\ |\!) \bullet$$
  $$end(y) \leq start(z) \vee end(z) \leq start(y)$$

- 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`

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

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

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
  - ✓ Extracting web ontologies systematically from Z specifications (ICFEM'02)
  - ✓ Checking Semantic Web Using Software Tools (FME'03, ICSE'04, WWW'04)
  - – Semantic Web environment for linking various formalisms (FME'02)

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