Vrije Universiteit Brussel

Inconsistency Management in Model-Driven Engineering

Ragnhild Van Der Straeten (SSEL)

Vrije Universiteit Brussel

Inconsistency Management in Model-Driven Engineering

- Model-driven Engineering
  - Primary assets are models
  - Model: views on the software system on a certain level of abstraction
- Inconsistency Management
  - Inconsistencies
    - Different models developed by different persons
    - Interdependencies between models poorly understood
    - Requirements unclear or ambiguous
    - Models are incomplete

How to manage inconsistencies?

Vrije Universiteit Brussel

Overview

- THE PAST
  - Classification of inconsistencies
  - Description Logics for detection and resolution
  - Graph Transformation for interdependencies
  - Tool Support

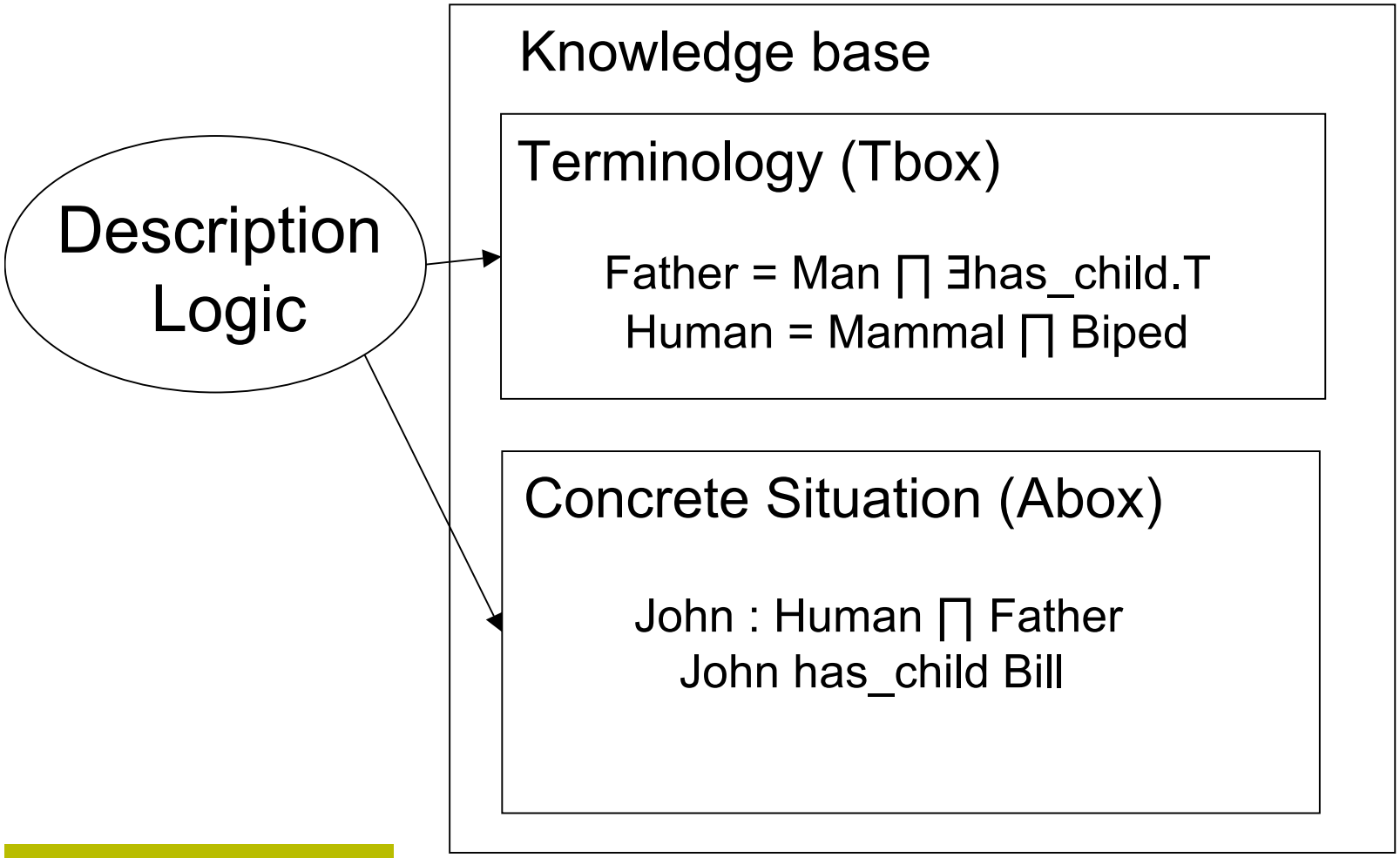
Vrije Universiteit Brussel

Classification of Inconsistencies

	Behavioural	Structural
Specification	Invocation interaction Observation interaction	Dangling type reference Inherited cyclic composition Connector specification missing
Specification - Instance	Specification incompatibility Specification behaviour incompatibility Invocation behaviour Observation behaviour	Instance specification missing
Instance	Invocation inheritance Observation inheritance Instance behaviour incompatibility	Disconnected model

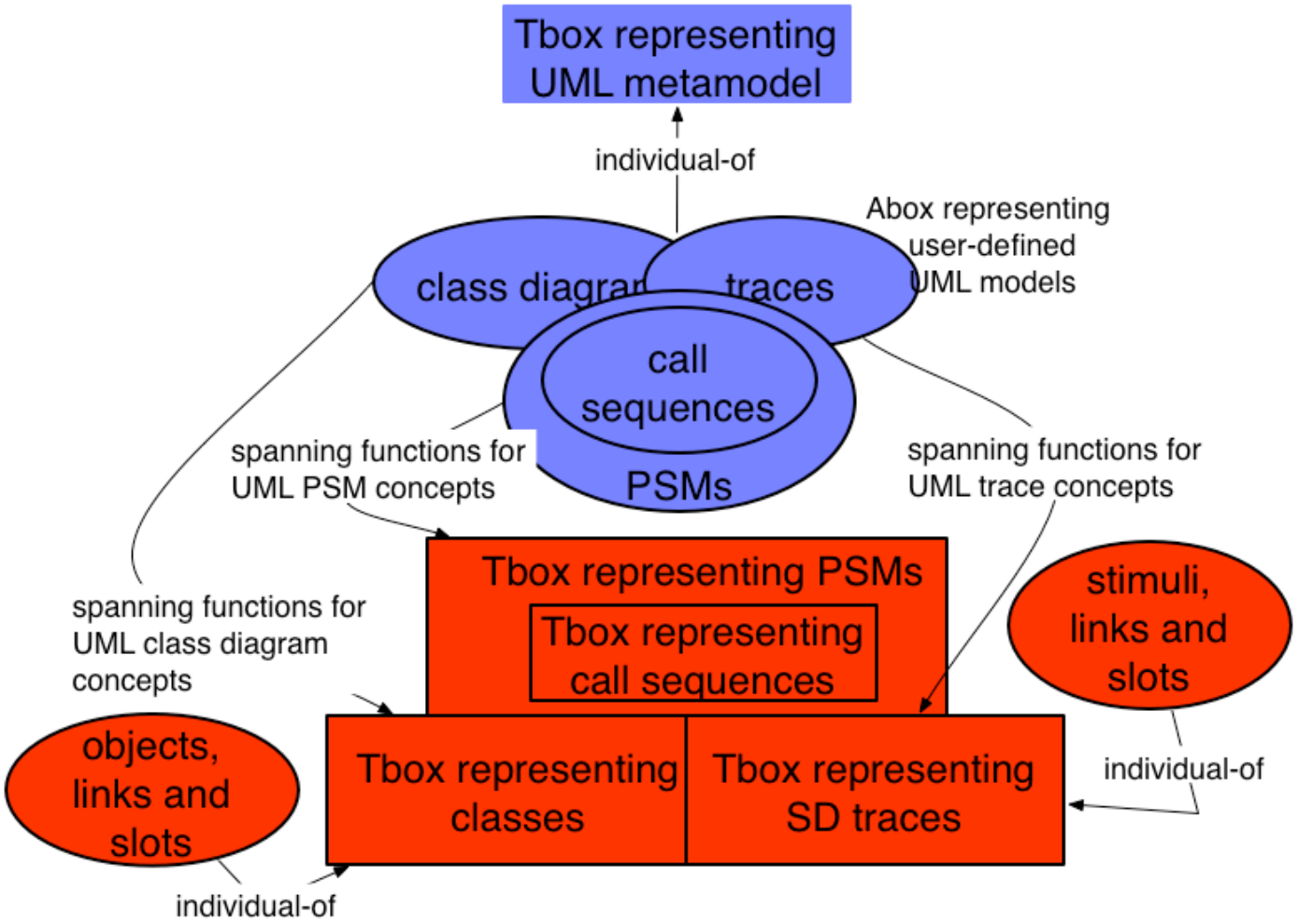
Vrije Universiteit Brussel

Description Logics



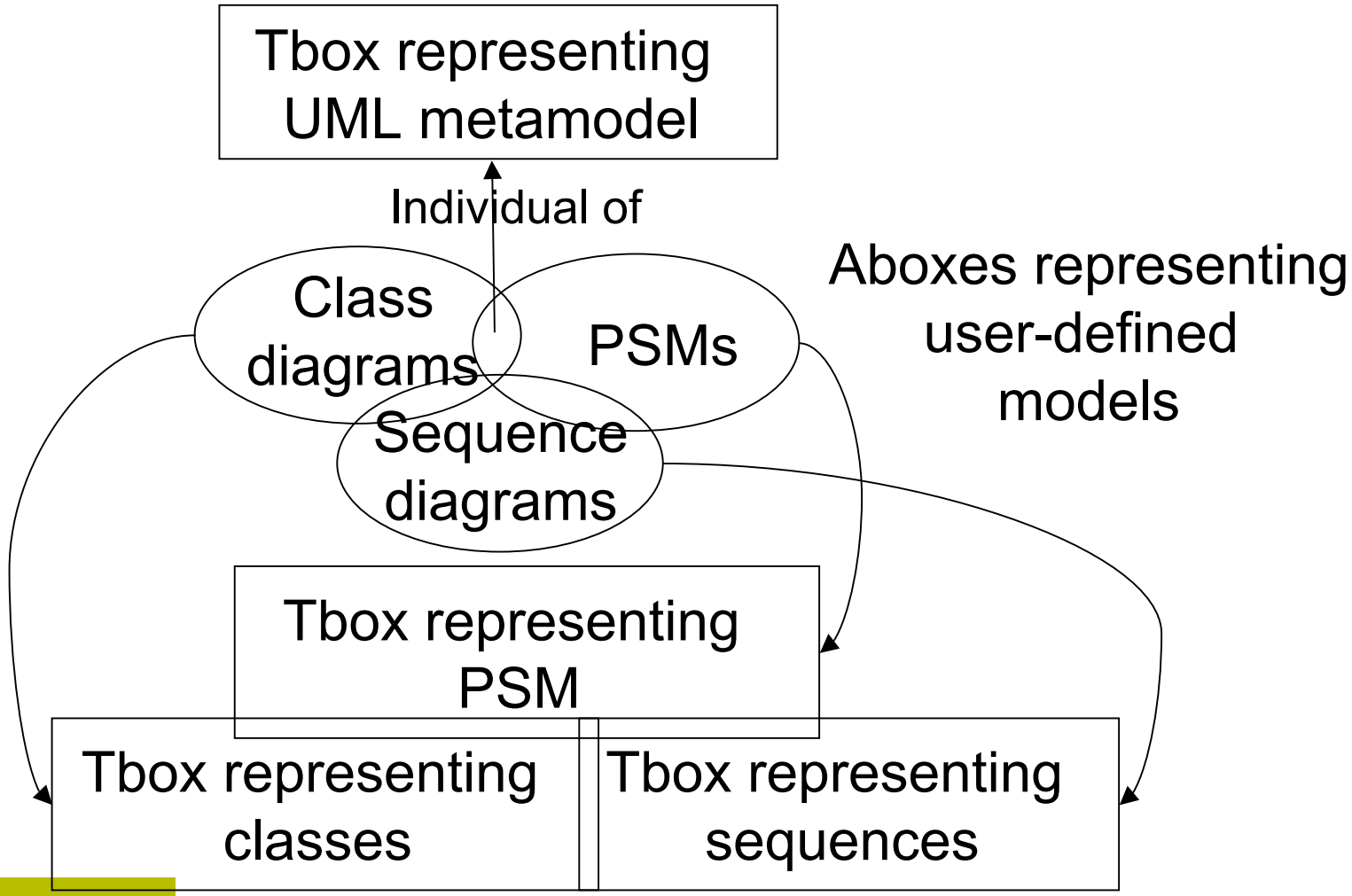
Vrije Universiteit Brussel

UML Models as DL KBs

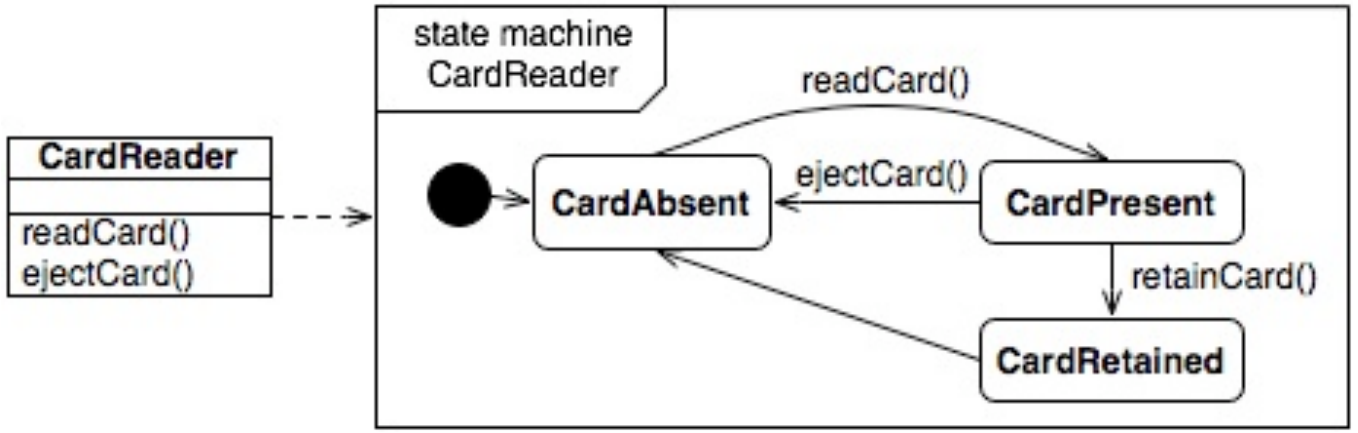


Vrije Universiteit Brussel

UML Models as DL Knowledge Bases



Detection of Inconsistencies



$$
\begin{aligned}
&ans(psm, cl, op) \\
&\quad \leftarrow \quad protocolstatemachine(psm) \wedge contextbehavior(psm, cl) \wedge \\
&\qquad regions(psm, r) \wedge transitions(r, t) \wedge protocoltransition(t) \wedge \\
&\qquad referredoperation(t, op) \wedge \ (\backslash(has\_known\_successor(ownedoperation, op)) \ \vee \\
&\qquad ((not(ownedoperation(cl, op))) \wedge (not(general(superc, c))) \wedge ownedoperation(superc, op)))
\end{aligned}
$$

Vrije Universiteit Brussel

Resolution of Inconsistencies

- Rule-based Inconsistency Resolution Approach

Condition:

check inconsistency

information about state of the model

user interaction

Conclusion

resolution actions: add/delete instances, fillers

Vrije Universiteit Brussel

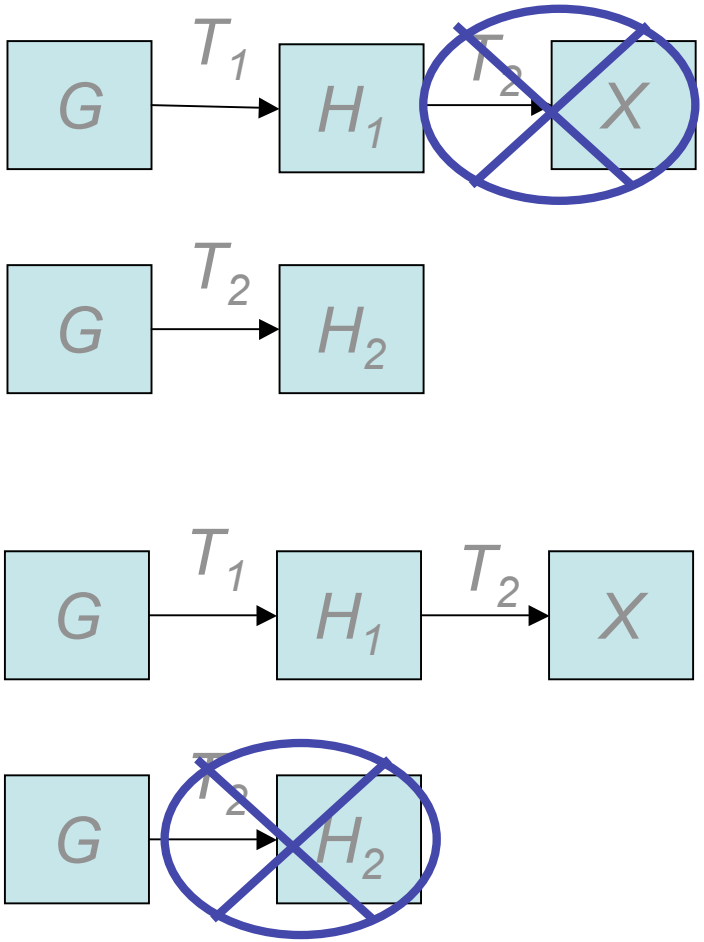
Inconsistency Resolution: Iterative & incremental

- Induced Inconsistencies
  - Resolution of one inconsistency may induce other inconsistencies as a side effect
- Conflicting Resolutions
  - Some resolutions may not be jointly applicable
- Resolution Cycles
  - Certain sequences of resolutions may reintroduce a particular inconsistency occurrence

Vrije Universiteit Brussel

Graph Transformations: Critical Pair Analysis

- T_1 and T_2 form a *critical pair* if
  - they can both be applied to the same initial graph G
  - applying T_1 prohibits application of T_2
- T_2 *sequentially depends* on T_1 if
  - T_1 can be applied to G but T_2 cannot
  - applying T_1 triggers application of T_2



Vrije Universiteit Brussel

Tool Support

- Poseidon plug-in
  - but:
    - Translation UML meta -> DL manually
    - UML 1.x
- Eclipse plug-in (was planned for 2006/Q4, 2007/Q1)
  - Usage of Ecore, EMF, UML2 editor(?), AGG Eclipse plug-in,
  - Automatic translation UML meta -> DL

Overview

- THE PAST
  - Classification of inconsistencies,
  - Description Logics for detection and resolution,
  - Graph Transformation for interdependencies,
  - Tool Support.
- THE FUTURE
  - General inconsistency management framework
    - Classification, comparison of formalisms,
    - Different modeling languages,
    - Bridging the gap between theory and practice.

WP4

Vrije Universiteit Brussel

Classification, comparison and integration of formalisms

- *Our research*
  - DL <--> Graph Transformations
- *Goal*
  - integrate variety of formal approaches
  - classify, compare and combine different formalisms for inconsistency management
- *Collaboration*
  - partners working in the area of inconsistency management or studying certain formalisms.
  - compare, integrate formalisms.

Vrije Universiteit Brussel

Different Modeling Languages

- *Our research*
  - Until now: UML class, sequence and state machine diagrams
- *Goal*
  - Incorporate other modeling languages, e.g., workflow languages and their AO extensions.
  - Which kinds of inconsistencies?
  - Formalism for detection and resolution?
  - Integrate, compare, … with other formalisms.
- *Collaboration*
  - Defining semantics of these MLs.
  - Defining right formalism.

Vrije Universiteit Brussel

Bridging the gap between theory and practice

- *Goal*
  - Based on comparison of formalisms:
    - Improve existing approaches
  - Use, apply formal techniques of other partners from other WPs
- *Research*
  - Integrate these approaches in our framework and provide feedback to the partners developing/investigating these formalisms.
- *Collaboration*
  - Partners studying certain formalisms -> provide these formalisms for inconsistency management.