LIFC

UFC

UML&FM'10

SysML to UML transformation for test generation purpose

November the 16th 2010

--

Jonathan Lasalle

Fabrice Bouquet – Bruno Legeard – Fabien Peureux

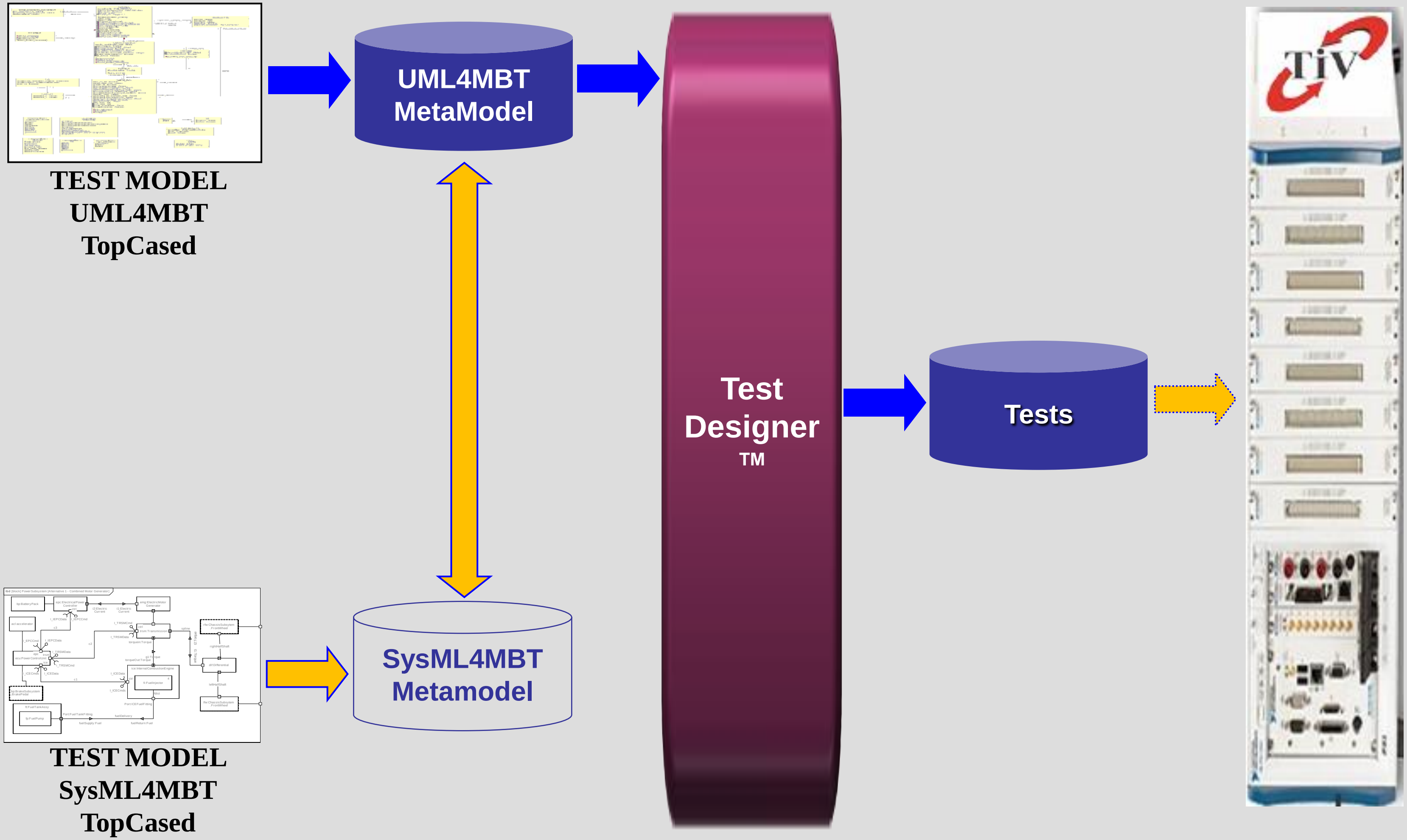
UFC UNIVERSITÉ DE FRANCHE-COMTÉ

LIFC

VETESS project

- VETESS: verification of vehicle embedded system by automatic test generation from specifications.
- Project labeled by the French competitiveness cluster “automotive of future” (2008/2010).
- Project members:
  - Smartesting: editor of tooled MBT solution (Test Designer).
  - Clemessy: testing bench provider (Test In View).
  - PSA Peugeot Citroën: car manufacturer.
  - LIFC: Model-Based Testing expertise (MBT).
  - MIPS: Model Driven Engineering expertise (MDE).

Toolchain



TEST MODEL
UML4MBT
TopCased

UML4MBT MetaModel

Test Designer TM

Tests

TEST MODEL
SysML4MBT
TopCased

SysML4MBT Metamodel

Outline

- UML4MBT
- SysML4MBT
- Transformation from SysML4MBT to UML4MBT
- Experimentations
- Conclusion & future works

UML4MBT [BGL+07]

- Class Diagram
  - Static view of the system.
  - Classes, associations, enumerations, class attributes and operations.
- Object Diagram
  - Concrete objects used to compute test cases.
  - Define the initial state of the system.
  - Must be an instanciation of the Class Diagram.

UML4MBT

- Dynamic view:
  - OCL expressions on pre/post condition of operations.
  - One Statemachine Diagram (annotated with OCL constraint).

🚫
- ➡parallel states
- ➡historic states
- ➡fork and join states

- Several restrictions on OCL.

SysML4MBT

- Block Definition Diagram (BDD)
  - Static view of the system and its environment.
  - Blocks, associations, compositions enumerations, blocks attributes and operations, PORTS and SIGNALS.
- Internal Block Diagram (IBD)
  - Interconnection between blocks.
  - Represent electrical or mechanical communications.

SysML4MBT

- Dynamic view:
  - OCL expressions on pre/post conditions of operations.
  - One or more Statemachine Diagram(s) (annotated with OCL constraints).
    - parallel states
    - historic states
    - fork and join states
  - Triggers: signal reception.

SysML4MBT

- OCL
  - Same restrictions than in UML4MBT.
  - Addition of OCL ^ operator (signal sending).
- Requirements Diagram
  - Represents system requirements.
  - Links requirements with model elements that satisfy them.

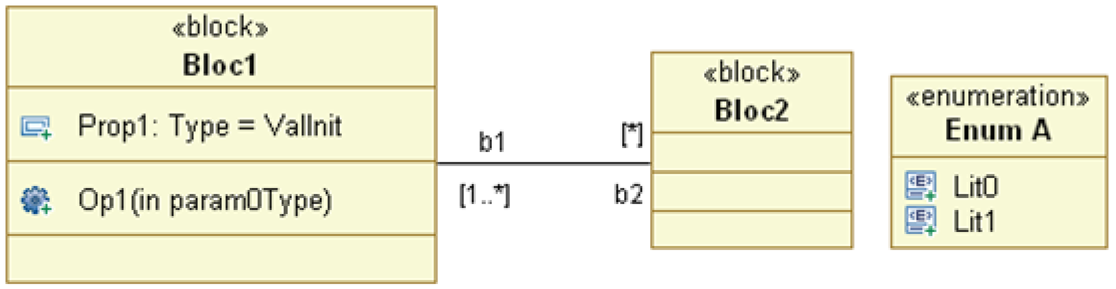
SysML4MBT to UML4MBT transformation – Algorithm

- Algorithm outline:
  1. Transformation of BDD & IBD to Class Diagram.
  2. Rewriting of Requirement Diagram.
  3. Translation of signal sends/receives.
  4. Transformation of fork/join states to parallel states.
  5. Rewriting of each composite, historic and parallel states by hierarchical stage.
  6. Merging of parallel Statemachines.
  7. Building of the Object Diagram.

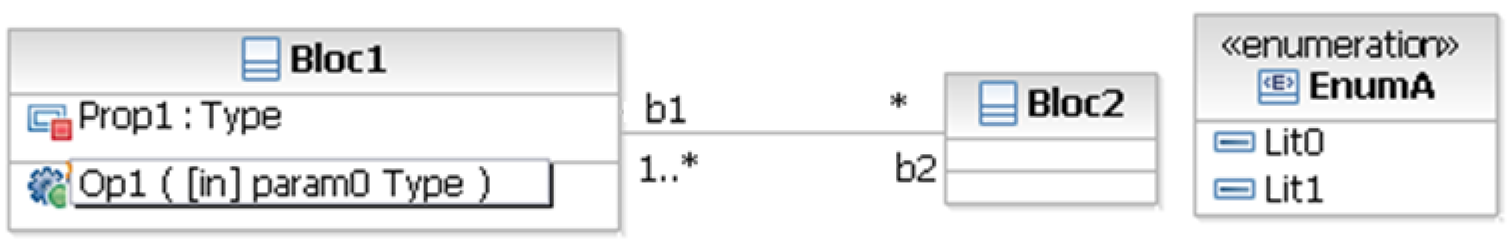
SysML4MBT to UML4MBT transformation – BDD

- SysML BDD to UML Class Diagram
  - No changes: blocks (classes), associations, operations, attributes and enumerations on both.

4MBT

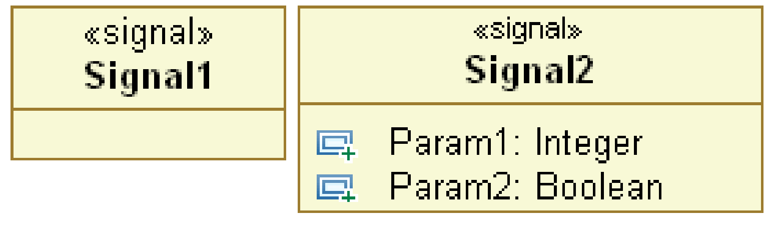


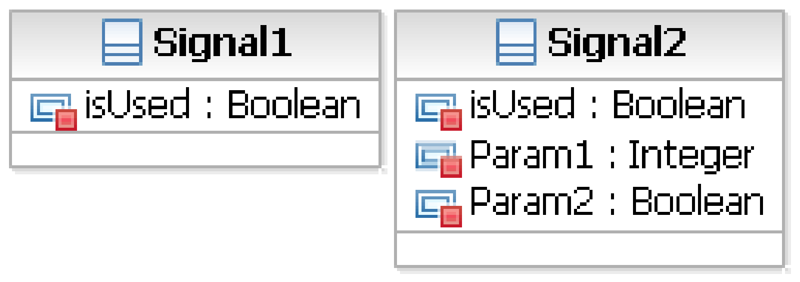
4MBT



SysML4MBT to UML4MBT transformation – Signals

- SysML IBD to UML Class Diagram
  - Signals:
    - ➡ Used on IBD.
    - ➡ Defined on BDD => UML classes.
    - ➡ Add a new attribute isUsed.

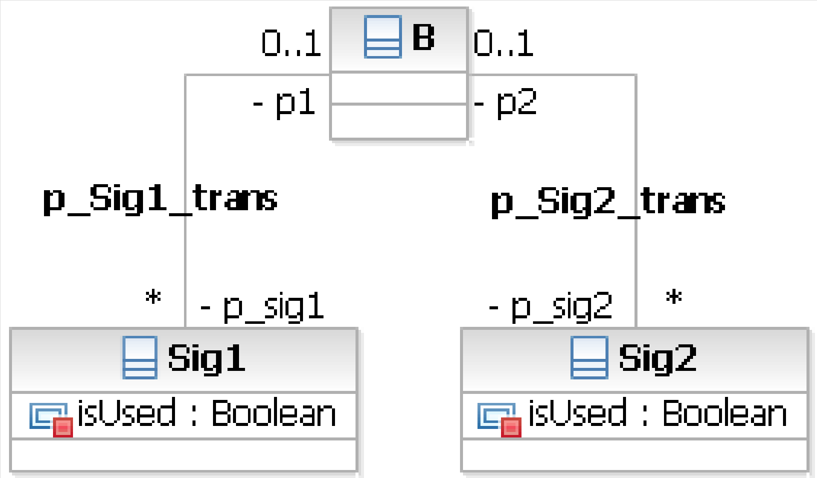




SysML4MBT to UML4MBT transformation – Ports

- SysML IBD to UML Class Diagram
  - Ports:
    - ➡Used on IBD.
    - ➡Defined on BDD.
    - ➡Which signal is pending on which port.

Block B
Port p which can receive Sig1 & Sig2



SysML4MBT to UML4MBT transformation – Requirements

- SysML Requirement Diagram to OCL
  - OCL for MBT: expression of requirements: */**@REQ: description of the requirement*/*
  - For each requirement of diagram
    - ➡ If satisfied by a transition: OCL added to effect.
    - ➡ If satisfied by an operation: OCL added to post condition.
    - ➡ If satisfied by an onEntry/onExit expression: OCL added to onEntry/onExit effect.

SysML4MBT to UML4MBT transformation – Dynamic view

- Statemachine Diagrams
  - Shared concepts:
    - initial states,
    - final states,
    - standard states,
    - composite states,
    - transitions without signal reception,
    - OCL expressions without the circumflex operator.

SysML4MBT to UML4MBT transformation – Signal sending

- Signal sending: ^ OCL operator
  - Block.Port ^ Signal(parameters).
  - Useful information: a new signal is pending in the corresponding port.
  - Instantiation of associations:

```
Block.Port^Sig(Val1,Val2)
```

```
let s = Sig.allInstances()->any(isUsed = false) in
s:Param1 = Val1 and s:Param2 = Val2 and
s:isUsed = true and
Block.Port_Sig->includes(s)
```

SysML4MBT to UML4MBT transformation – Signal receiving

- Signal receiving (trigger on transitions)
  - The corresponding signal is pending.
  - After crossing the transition, the signal was read.
  - Check of link and deletion.

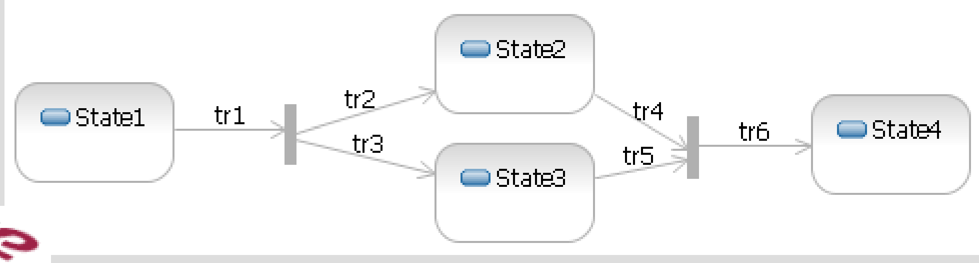
A trigger defining the reception of “Sig” on “Port” hosted by “Block”.

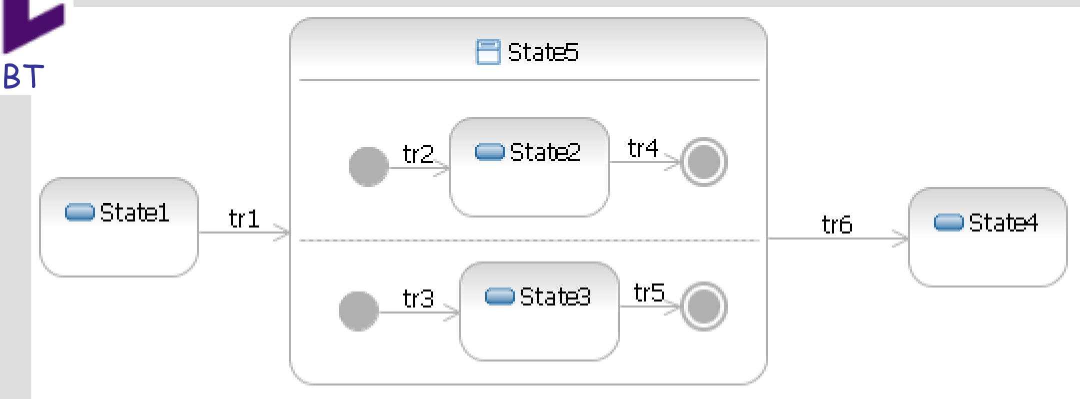
Add of guard: *[Block.Port_Sig -> notEmpty()]*

Add of effect: *let s = Block.Port_Sig.allInstances()->any(true) in s:isUsed = false and Block.Port_Sig->excludes(s)*

SysML4MBT to UML4MBT transformation – Fork & join states

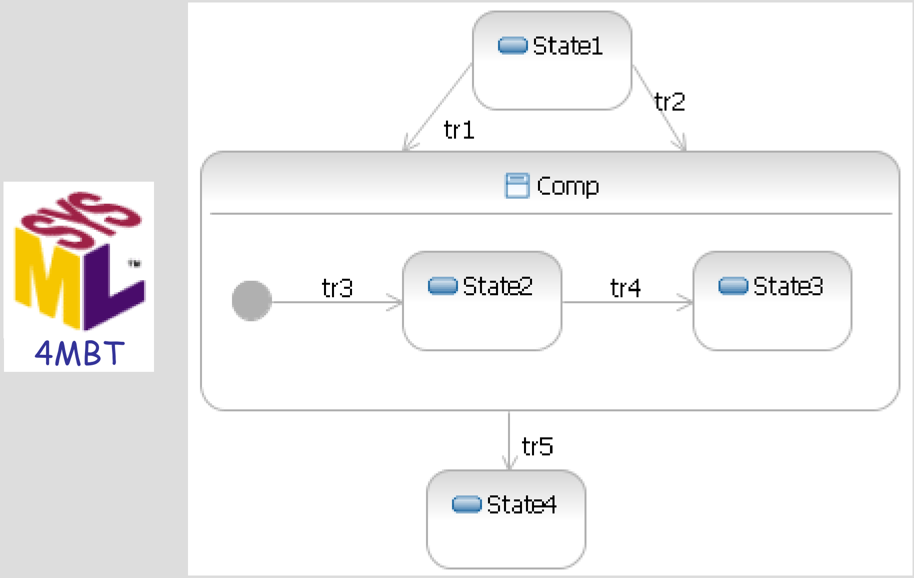
- Fork & join states
  - Rewriting to parallel states.

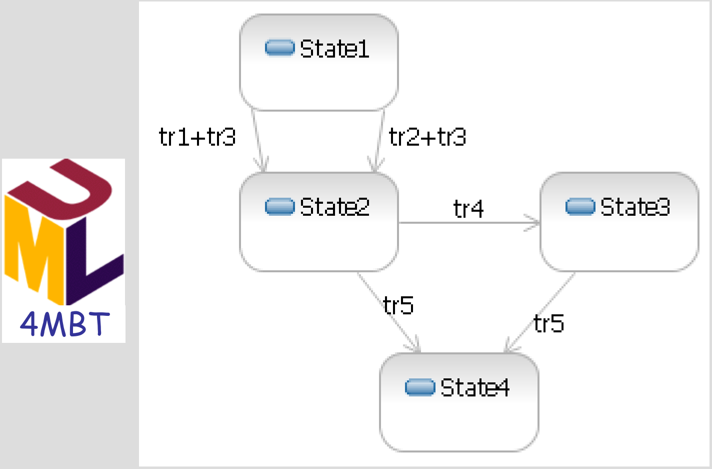




SysML4MBT to UML4MBT transformation – Composite states

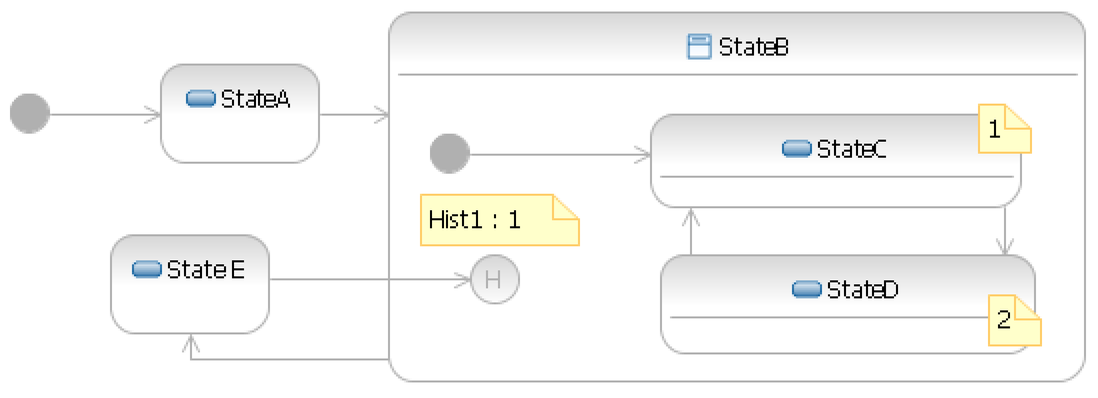
- Composite states
  - Must not contain parallel or historic states.

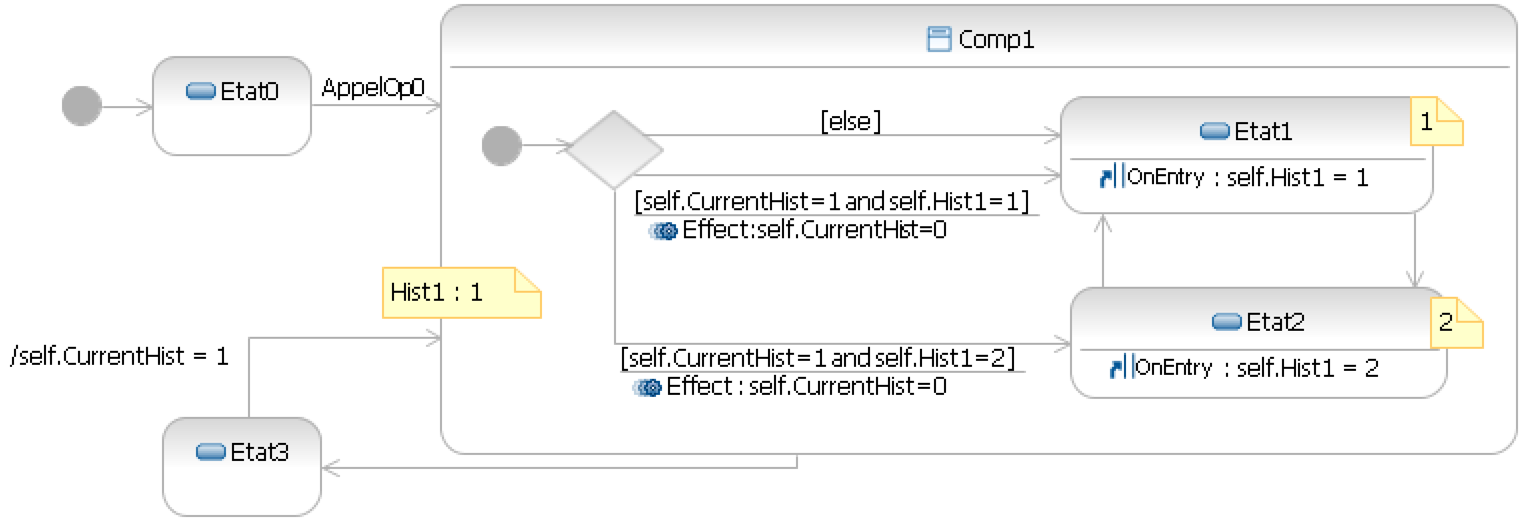




SysML4MBT to UML4MBT transformation – Historic states

- Historic states




Etat0
AppelOp0
Comp1
[else]
Etat1
OnEntry : self.Hist1 = 1
[self.CurrentHist=1 and self.Hist1=1]
Effect:self.CurrentHist=0
Hist1 : 1
/self.CurrentHist = 1
[self.CurrentHist=1 and self.Hist1=2]
Effect : self.CurrentHist=0
Etat2
OnEntry : self.Hist1 = 2
Etat3
1
2

SysML4MBT to UML4MBT transformation – Parallelism

- Parallel states
  - Same steps to merge parallel states and parallel Statemachines.
- Parallelism of Statemachine Diagram
  - Multiple Statemachines in SysML4MBT.
  - Single Statemachine in UML4MBT.

  => Merging of Statemachines

SysML4MBT to UML4MBT transformation – Parallelism

- Following steps:
  1. Translation of all complex states (fork, join, composite, parallel and historic states).
  2. Cartesian product of all Statemachines:
     1. Draw transition only if a path to reach start state exists.
     2. Informations of pending signals are stored on states.
     3. Transitions triggered by signal receiving:
        drawn only if the signal is pending on the root state.

SysML4MBT to UML4MBT transformation – Object Diagram

- UML Object Diagram
  - Each class => one instance.
  - Associations => instantiated using the minimum number of links (lower multiplicity).
  - Classes representing signals:
    instantiated according how many times it can be pending at the same time.

Case studies

- Lighting
  - Front lighting system of a car.
  - Light on and light off independently headlights and highlights with a control lever.
- Steering
  - Representation of the steering column of a car.
  - Reaction of the steering column in regard of road.
- Wiper
  - Specification of the front wiper system of a car.
  - The modeled functionalities are slow speed drying up, high speed drying up, intermittently speed drying up and cleaning with drying up.

Results

		Lightings	Steering	Wiper
SYSML4MBT	Blocks	6	9	15
	Connectors	4	10	18
	Statemachines	5	6	12
	States	(2,2,2,2,4)	(2,2,2,2,2,2)	(1,1,1,1,1,2, 17,10,2,2,2,2)
	Transitions	(3,3,3,3,9)	(3,3,3,3,2,8)	(3,4,3,5,2,4, 53,17,3,3,3,3)
UML4MBT	Classes	10	16	29
	Objects	15	20	57
	States	64	18	2526
	Transitions	256	123	31873

Conclusion & future works

- Rewriting rules to translate SysML4MBT models into UML4MBT models.
- Made it possible to generate test cases from SysML4MBT models with Test Designer.
- Problem about Scalability.
  - Improving rewriting rules.
  - Increasing UML4MBT expressiveness (native support of parallelism).
- About testing: Increasing model coverage with new test generation strategies.

The end

Any questions?

- [BGL+07] F. Bouquet, C. Grandpierre, B. Legeard, F. Peureux, N. Vacelet, and M. Utting. A sub-set of precise UML for model-based testing. In Proceedings of the 3rd International Workshop on Advances in Model Based Testing (A-MOST'07), pages 95{104, London, UK, July 2007. ACM Press.