AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

# Towards Fault-based Generation of Test Cases for Dependable Embedded Software

4. Workshop Entwicklung zuverlässiger Software-Systeme
Stuttgart, 30.6.2011

Wolfgang Herzner, Rupert Schlick, AIT
Harald Brandl, Technical University Graz
Johannes Wiessalla, Ford Forschungszentrum Aachen

Tut es, was es soll?

Immer?

Tut es nicht, was es nicht soll?

Nie?

# Integrierter Entwicklungsprozess - *embedded*

# Testen

Zweck:

- Fehlerwirkungen nachweisen
- Qualität bestimmen
- Vertrauen erhöhen

Eigenschaften:

- Benötigt Domänenwissen
- Variabel in den Ergebnissen
- Aufwändig (50-80%)
- Testdurchführung am Echtzeitsysteme mitunter teuer

Wunsch:

- Wenig Testfälle (Testdurchführungsaufwand)
- Genug/ die richtigen Testfälle (Vertrauen)

# Mutation Testing

- Richard Lipton, 1971
- DeMillo/Lipton/Sayward, 1978
- Mutant: Stellvertreter für fehlerhaftes Artefakt
- Bewertung von Test Suiten
- Geringe industrielle Nutzung, rege akademische Nutzung

- Kleine Fehler, große „gehen mit“;
- „Direkteres“ Maß als gängige Abdeckungsmaße – Fehlerbezug
- Parametrisierbar über Fehlermodell
- Fehlermodell sprachabhängig

# Mutation Based Test Case Generation

- Testfall für bestimmte Qualität finden
- Konformanz Beziehung (Refinement)
- Behauptung:
  Mutiertes Artefakt („Implementierung“)
  ist konform zum
  Original-Artefakt („Spec“)
  (für alle Input-Output-Sequenzen)
- Gegenbeispiel = Testfall
- Model checking/Constraint solving
- Nutzbar auf allen „ausführbaren“ Artefakten:
  - Modell
  - Code
  - Executable

# Abdeckung

```
context TriangType(a: int, b: int, c: int): String
pre: a < (b+c)  and  b < (a+c)  and  c < (a+b)
post: if ((a = b) and (b = c))
      then result = "equilateral"
      else if ((a = b) or (a = c) or (b = c))
           then result = "isosceles"
           else result = "scalene"
      endif endif
```

- Testfälle

*a = 1, b = 1, c = 1, result = "equilateral"*
*a = 2, b = 2, c = 1, result = "isosceles"*
*a = 2, b = 3, c = 4, result = "scalene"*

- Erkennen nicht:

```
post: if ((a = a) and (b = c))
      then result = "equilateral"
      else if ((a = b) or (a = c) or (a = c))
           then result = "isosceles"
           else result = "scalene"
      endif endif
```

- Erkennt beides:

*a = 1, b = 2, c = 2, result = "isosceles"*

# Model-based GENeration of Tests for Embedded Systems

AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

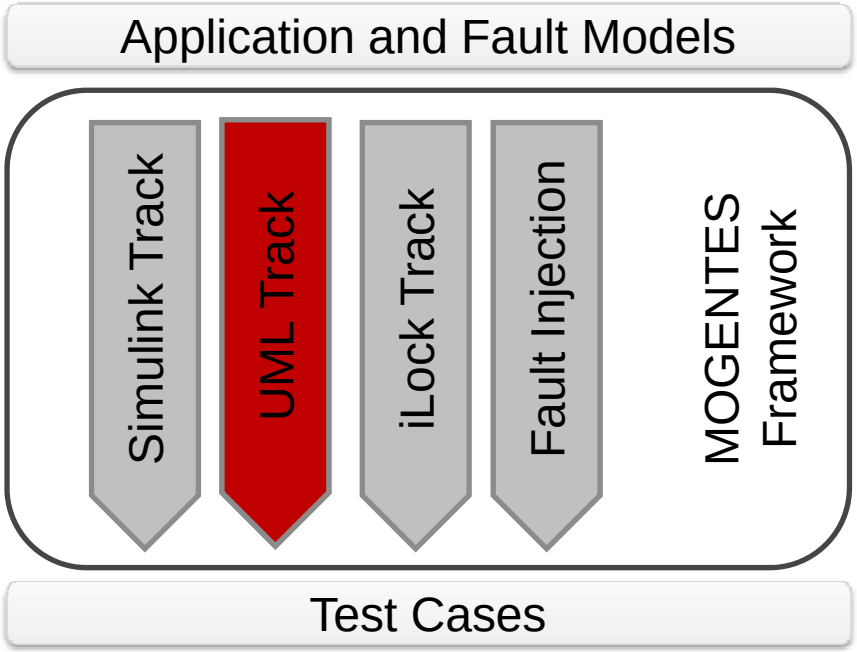
MŰEGYETEM 1782

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

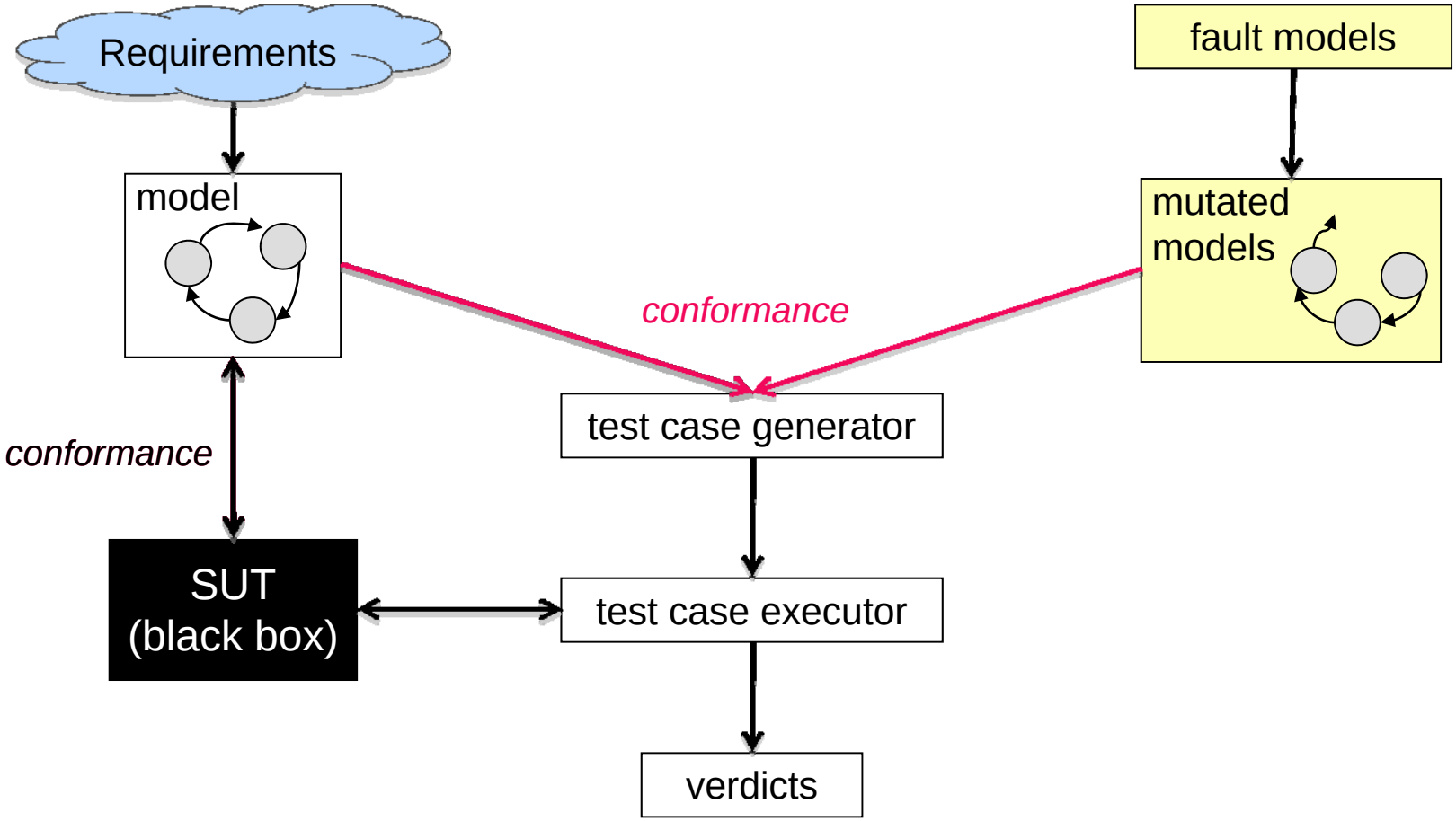
SP

PROVER TECHNOLOGY

TU Graz

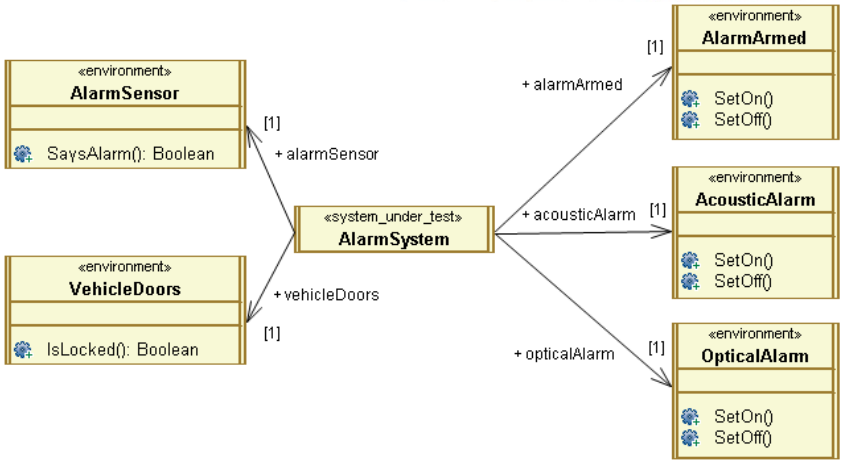


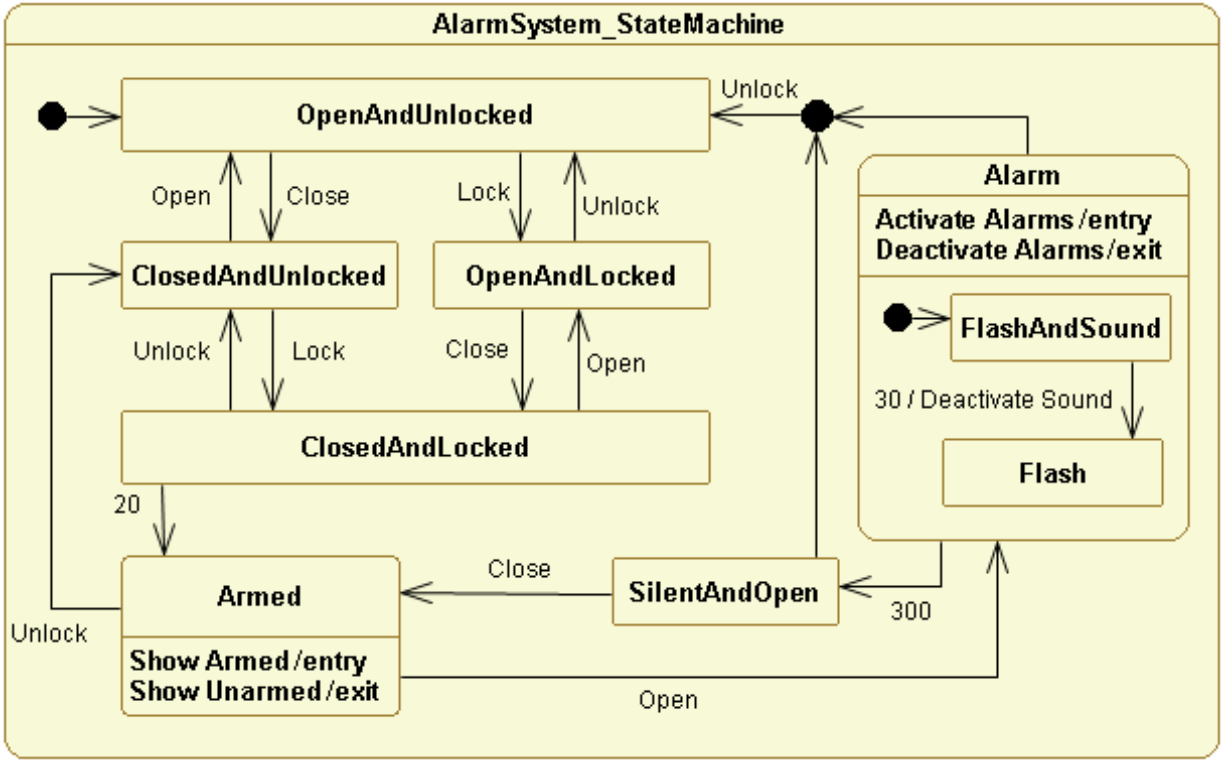

# Model-Based Mutation Testing



# UML Modelle

- Class diagrams (structure)
- State machines (behaviour)
- Stereotypes (testing interface, additional markup)



# Unterstützte UML Elemente

- state machine diagrams
- (active) Classes
- lightweight inheritance
  (no polymorphism / late-binding)
- OCL constraints (guards)
- orthogonal regions
- nested regions
- signals (incl. broadcast)
- time trigger
- substates
- junctions
- change triggers

# Mutationsoperatoren

- Init values, constants, OCL literals, AGSL literals:
  - Constant Value Variation Step / Factor
  - Enum Value Exchange
  - Set Constant Value to Zero/Max
- States:
  - Remove State Entry/Exit Action
  - Exchange State Entry/Exit Action Method
- Transitions:
  - Remove Transition
  - Exchange Transition Source/Target
  - Exchange Transition Trigger
- Triggers
  - Exchange Trigger Signal
  - Minimally Change Time Trigger Duration
  - Substantially Change Time Trigger Duration
  - Minimize/Maximize Time Trigger Durartion
  - Modify Change Trigger Expression
  - Exchange Call Trigger Method
- Guards:
  - Fix Guard Value
  - Invert Guard
  - Modify Guard Expression
- Effect:
  - Remove Effect
  - Exchange Effect Method
  - Modify Effect Body
- OCL:
  - Fix Expression
  - Negate Expression
  - Fix Subexpression
  - Negate Subexpression
  - Modify Boolean Operator
  - Modify Relational Operator
  - Modify Arithmetic Operator
  - Modify Set Operator
  - Modify Quantifier
- AGSL:
  - Remove Statement
  - Reorder Statement
  - Fix Parameter/Property
  - Modify Parameter/Property
  - Modify Operator
  - Fix Operand
  - Modify Operand
  - Fix Result

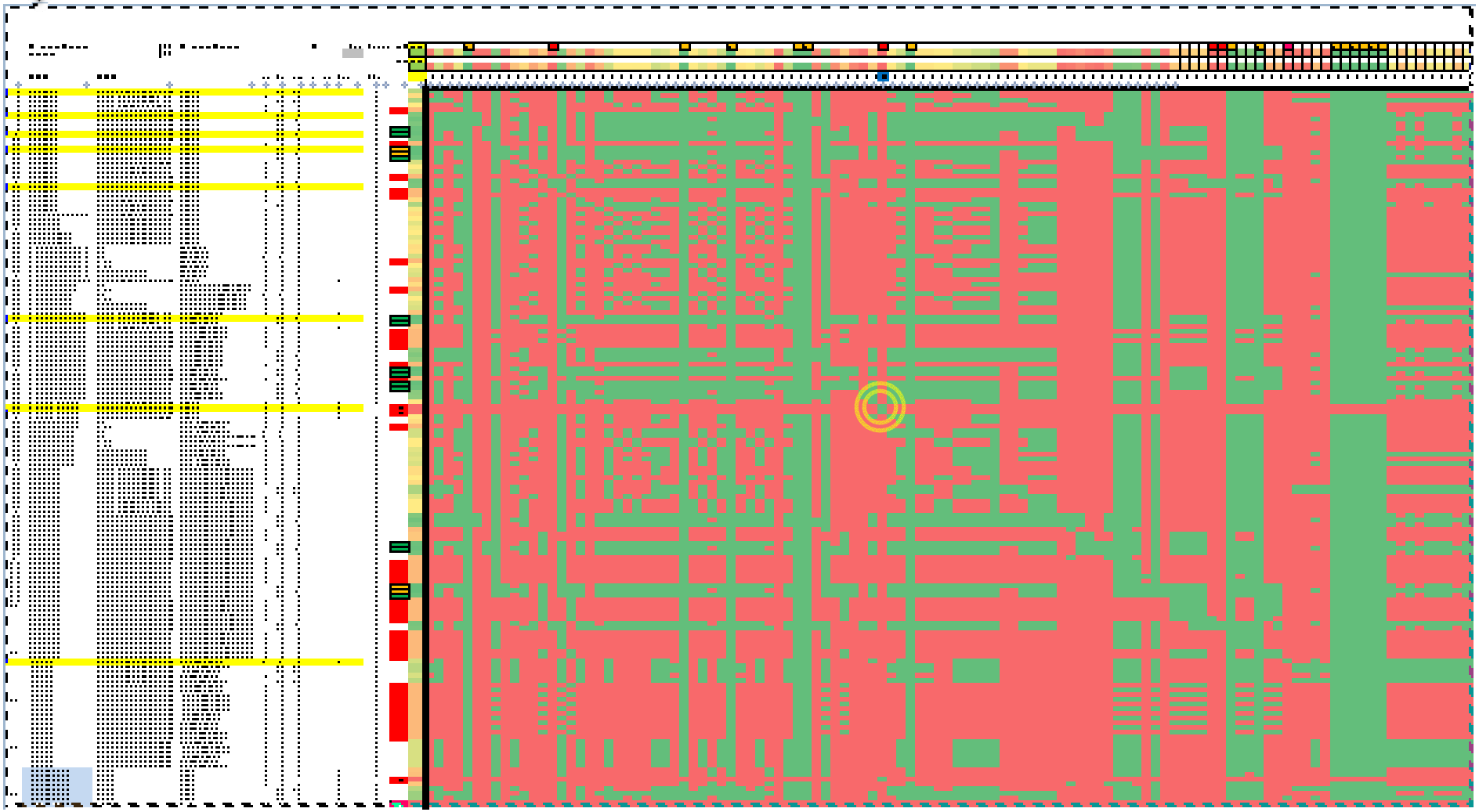
# Mutants and Test Cases

- 15 applicable mutation operators,
- 20 locations
- 111 mutants
- 152 generated test cases (minimal search depth)

| | **Manual Set** | **„Minimal“ generated set** |
|---|---|---|
| Source | 8 use cases | 8 mutants |
| I/O Steps | 70 | 137 |
| Accumulated time | 470s | 800s |
| Found mutants | 83 / 75% | 111 / 100% |

# Coverage overview

↓ Test cases → Mutants

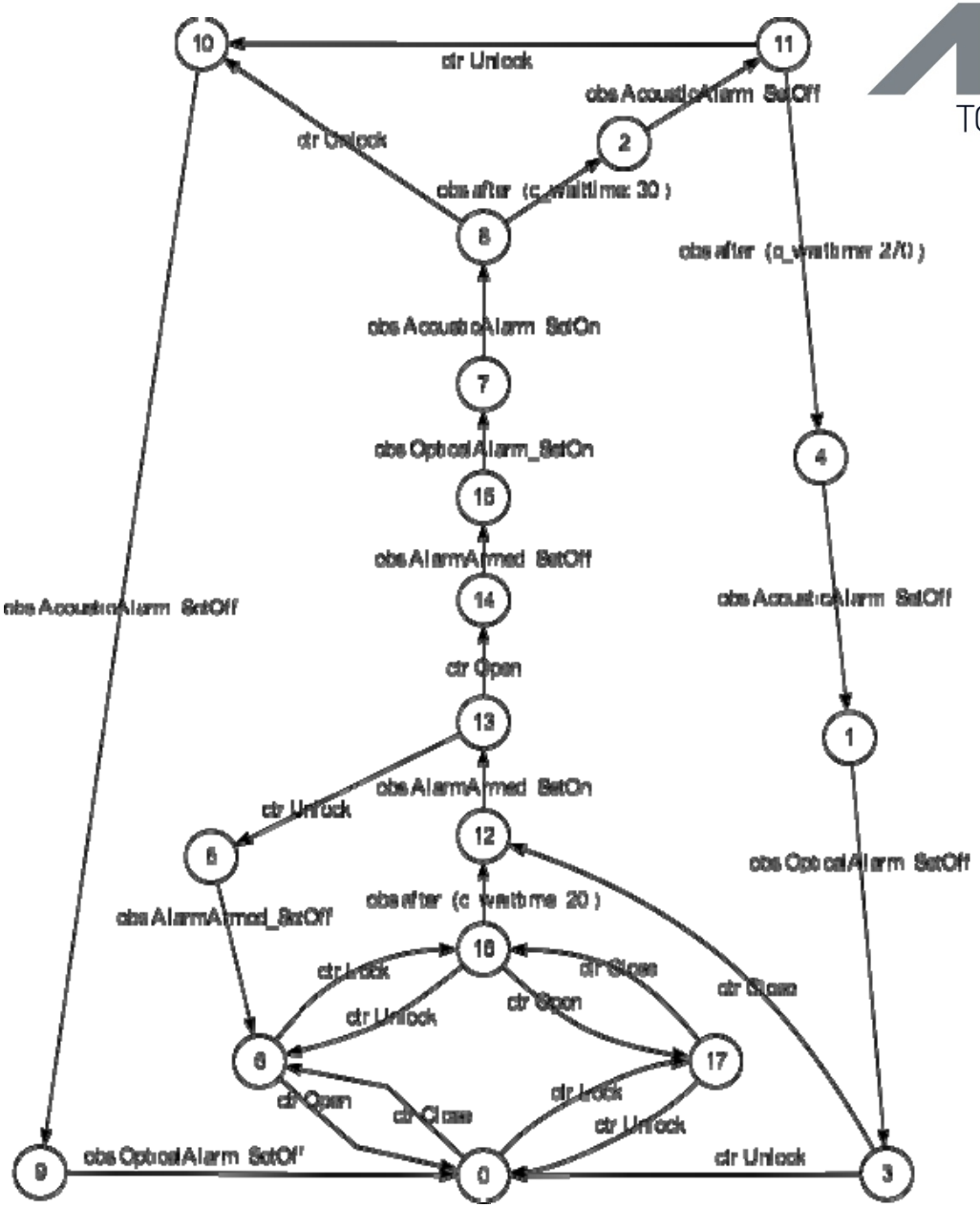


# Offene Themen / Nächste Schritte

- Skalierbarkeit
  - Symbolisch lösen
  - Model checking optimieren
  - Model splitting
- Effiziente Testfallsuiten
  - Auswahl Mutationsoperatoren
  - Erreichbarkeitsanalyse
  - Reihung Mutanten
- Äquivalente Mutanten
  - Heuristiken

# Backup

# Labeled Transition Systems (LTS)



# Input Output Conformance (ioco) (1)

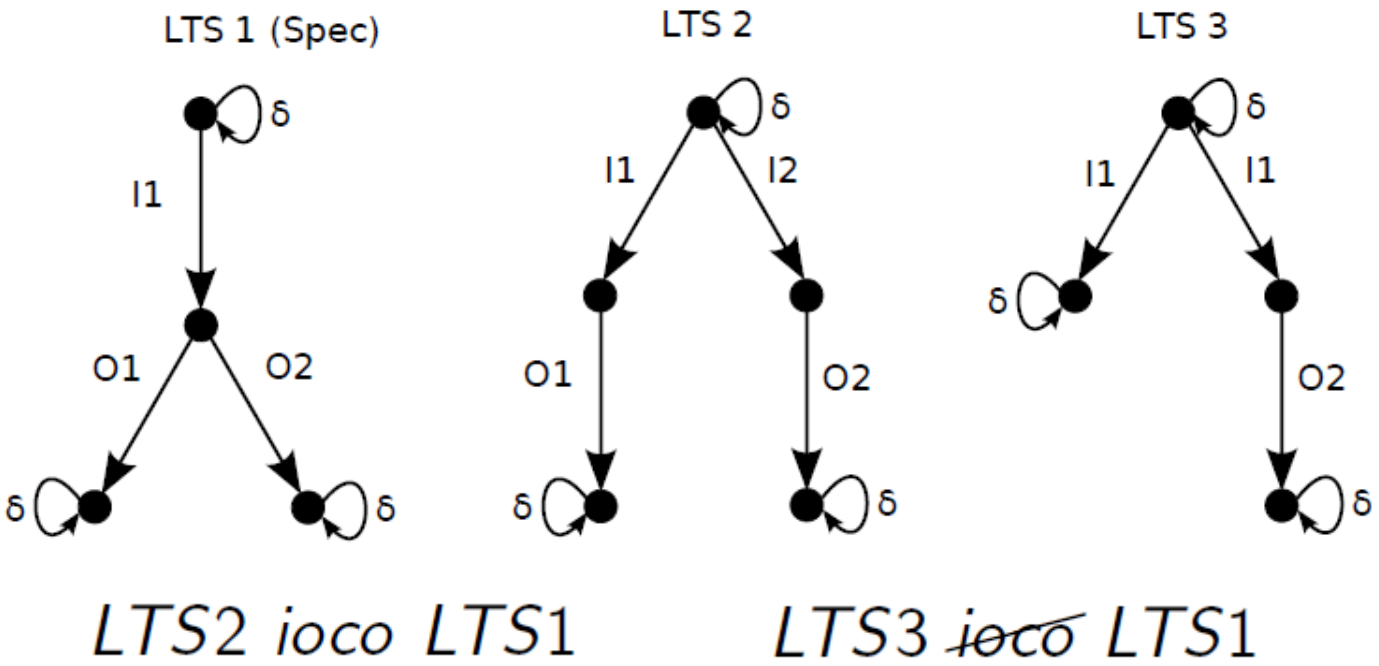
- Actions are partitioned into input $L_I$, output $L_U$, and internal events $\tau$
- $\delta$ denotes the observation that no output has occurred
- Implementations are input-enabled, i.e. an input cannot be blocked

$$i \textbf{ ioco } s =_{df} \forall \sigma \in Straces(s) \bullet$$
$$out(i \text{ after } \sigma) \subseteq out(s \text{ after } \sigma)$$

where Straces are defined over the alphabet $L_I \cup L_U \cup \{\delta\}$

# Input Output Conformance (2)

$$i \textbf{ ioco } s =_{df} \forall \sigma \in Straces(s) \bullet$$
$$out(i \text{ after } \sigma) \subseteq out(s \text{ after } \sigma)$$




*LTS2 ioco LTS1*  *LTS3 ~~ioco~~ LTS1*

# Figures

- Duration:
  - 36 (+3) months
    (Jan. 2008 – Dec. 2010)
- Costs:
  - 4,4 M€ Total
  - 3,1 M€ Funding
- Efforts:
  - 400,5 PM RTD
  - 48,5 Demo
  - 12,0 Mgmt
- Coordination:
  - AIT

Homepage

www.mogentes.eu

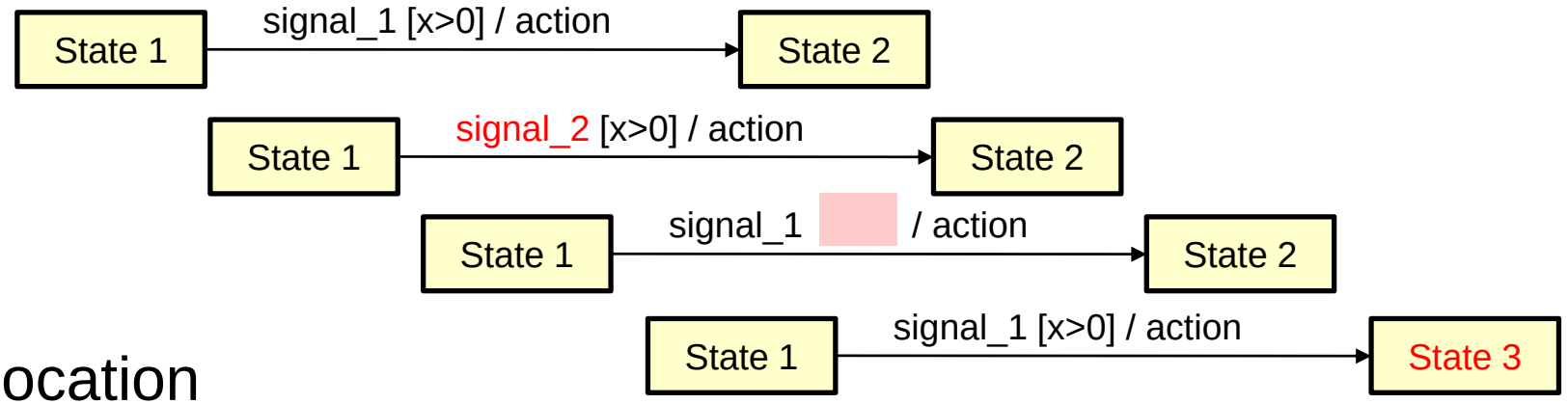
# Partner

- Universities
  - Budapest University of Technology and Economics (HU)
  - ETH Zurich (CH) / Stanford University (UK)
  - Graz University of Technology (AT)
- Research Organisations
  - AIT – Austrian Institute of Technology
  - SP Technical Research Institute of Sweden
- Industrial Demonstrators
  - Ford Forschungszentrum Aachen (DE)
  - Prolan Irányítástechnikai ZRT (HU)
  - Re:Lab S.R.L. (IT)
  - Thales Rail Signalling Solutions GmbH (AT)
- Tool Developer
  - Prover Technology AB (SE)

# Anatomy of a Mutation

Example: UML state diagram

- Mutation operators, e.g.:




- Location
  - affected model element
- Parameter
  - E.g. the new operator if an expression operator is changed

# Abstract Test Case

Example:

| | |
|---|---|
| 1. Ctrl : Alarm sensor off | initialisation |
| 2. Ctrl : Lock the vehicle | … |
| 3. after(20) | … |
| 4. Obsv : AlarmArmed SetOn | observed signal |
| 5. Ctrl : Alarm trigger | alarm event |
| 6. Obsv : AlarmArmed SetOff | expected reactions |
| 7. Obsv : OpticalAlarm SetOn | … |
| 8. Obsv : AcousticAlarm SetOn | … |
| 9. after(30) | after 30 seconds |
| 10. Obsv : AcousticAlarm SetOff | acoustic alarm switches off |
| 11. Ctrl : Unlock | unlock the car |
| 12. Obsv : OpticalAlarm SetOff | optical alarm switches off |