UML in Action

A Two-Layered Interpretation for Testing

Bernhard K. Aichernig

Joint work with
Harald Brandl, Elisabeth Jöbstl, Willibald Krenn

Institute for Software Technology
Graz University of Technology
Austria

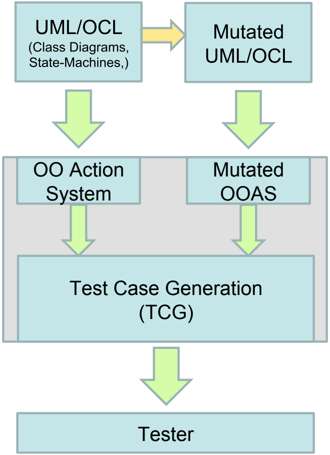
UML&FM, Nov 16, 2010

Research Context

MOGENTES

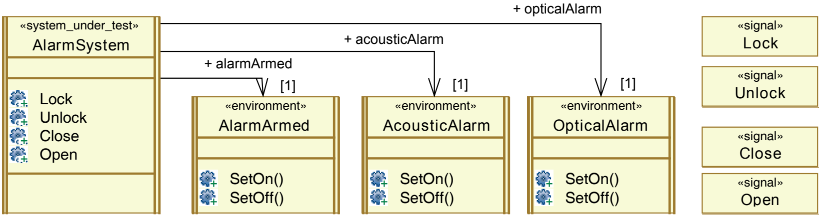
- FP7 STREP: Model-based Generation of Tests for Dependable Embedded Systems
- 10 Partners:
    - A.I.T (Austria), SP (Sweden)
    - Budapest Univ. of Techn. (Hungary), TU Graz (Austria), Oxford Univ./ETH (UK, Switzerland)
    - Ford (Germany), Prolan (Hungary), Prover (Sweden), Re:Lab (Italy), Thales Rail Signalling Solutions (Austria)
- Duration: 1.1.2008–31.12.2010

Model-Based Mutation Testing

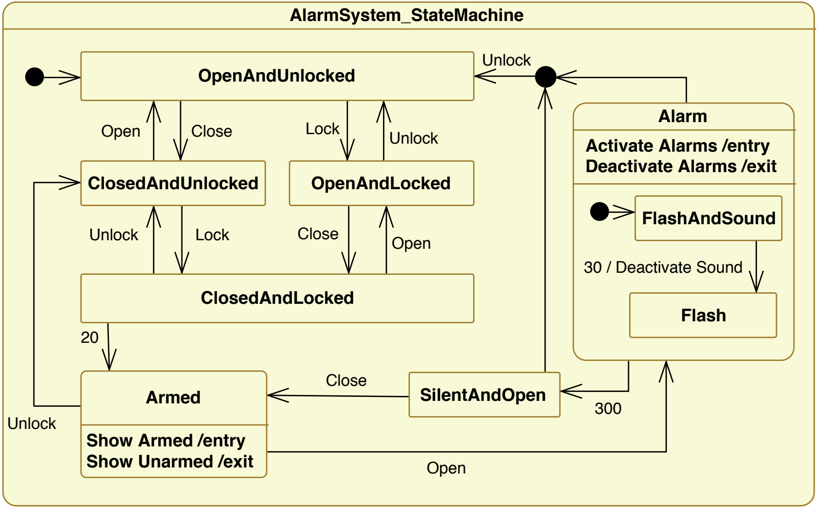


- UML State Charts: hierarchical, parallel states, constraints over quantors etc. (Papyrus)
- Object-Oriented Action Systems provide formal semantics
- Test-objective based on fault models: mutations in model
- Idea: generate test cases to prevent modeled faults in SUT
- Translator joint work with AIT
- TCG also for Hybrid Systems

Car Alarm System: Interface



Car Alarm System: Behaviour



Action Systems Semantics

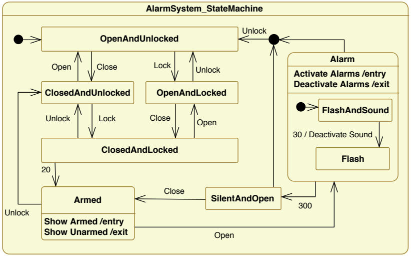
Action System:

$$
\begin{array}{ll}
|[& \textbf{var} \quad v : T := init \\
 & \textbf{methods} \\
 & M_1; \ldots; M_n \\
 & \textbf{actions} \\
 & A_1 = g_1 \rightarrow v := e_1; \\
 & \ldots; \\
 & A_m = g_m \rightarrow M_i(e_i); \\
 & \textbf{do}\ A_1 \\
 & \quad \Box \\
 & \quad A_2; A_3 \\
 & \quad // \\
 & \quad A_m \\
 & \textbf{od} \\
]| & : M_I
\end{array}
$$

Motivation:

- Well-suited for embedded systems modeling (Event-B)
- Action view maps naturally to LTS testing theories
- Solid foundation:
  - precise semantics
  - refinement
- Compositional modeling
- Many extensions available:
  - object-orientation
  - hybrid systems

UML State Transition as Action



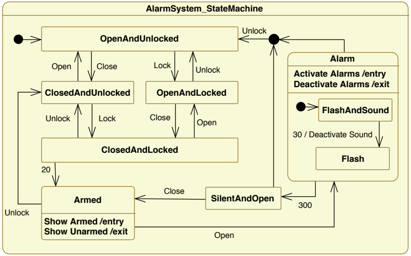

```
transition_Armed_to_Alarm =
  requires
    state = Armed and
    events <> [nil] and
    (hd events)[0] = Open
  :
    state := Alarm
  end;
...
dequeue =
  requires
    events <> [nil]
  :
    events := tl self.events
  end
```

UML Entry Actions

Protocol layer in do-od-block serves to model entry actions:

```
do
  (  (transition_Armed_to_Alarm;
       call_ShowUnarmed;
       call_ActivateAlarm;
       call_AcousticAlarm_SetOn  )
     []
       transition_OpenAndLocked_to_ClosedAndLocked
     []
       /* .. other transitions .. */
  )
  // dequeue()
od
```

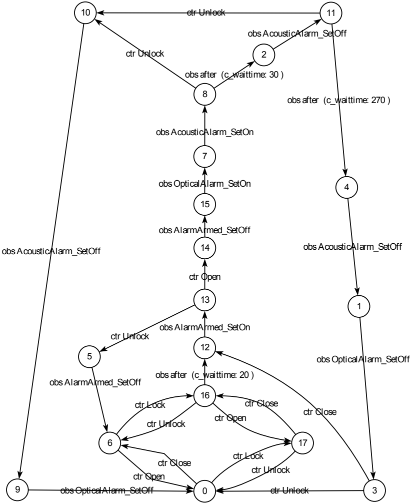
LTS Semantics of Action Systems



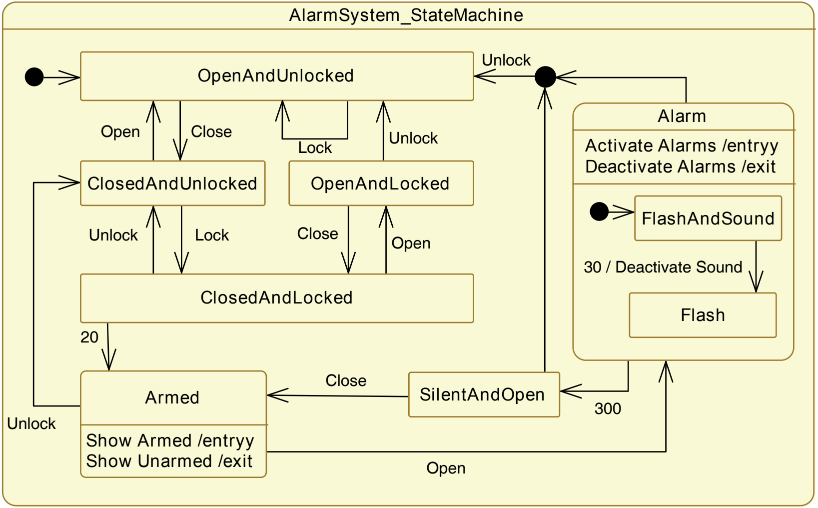
Mapping to CADP toolbox:

- model simplification
- model checking
- scenario-based TCG in TGV

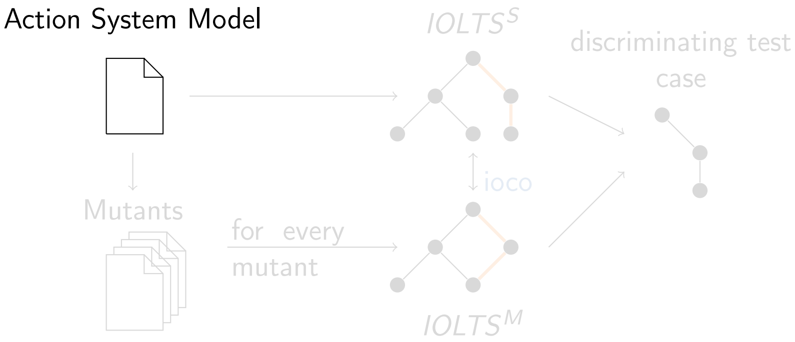
IOLTS testing theories



76 Alarm System Mutations, e.g.

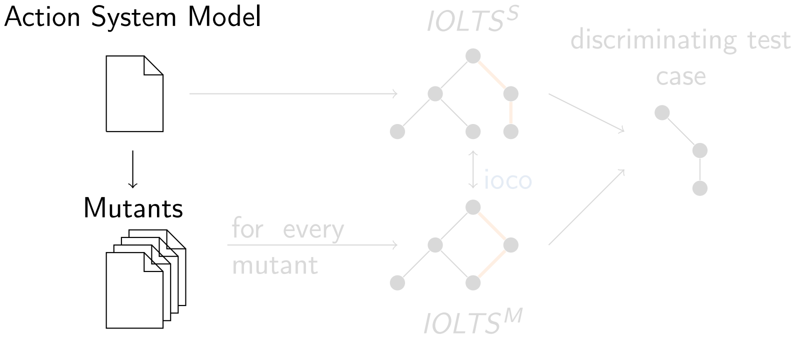


Mutation-Based Test Case Generation



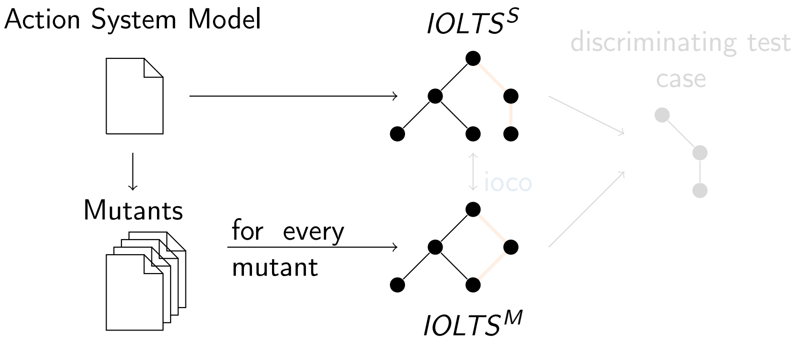
ioconf . . . input-output conformance

Mutation-Based Test Case Generation



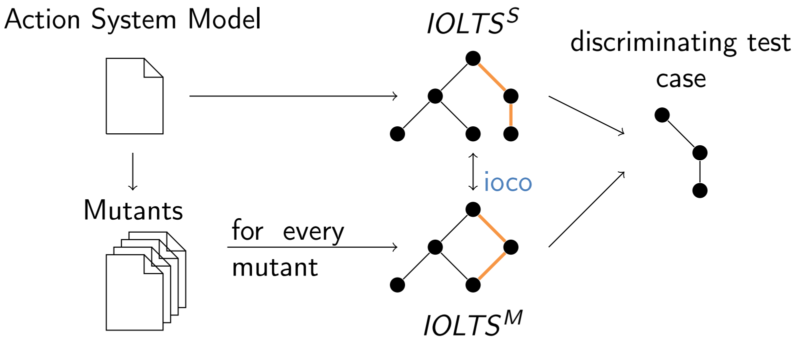
ioconf . . . input-output conformance

Mutation-Based Test Case Generation



ioconf . . . input-output conformance

Mutation-Based Test Case Generation



ioconf . . . input-output conformance

Tool Chain

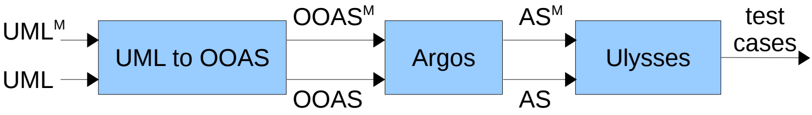


Figure: Test Case Generation Tool Chain.

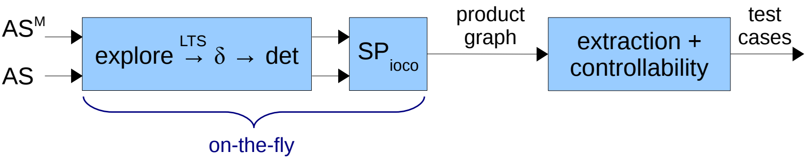


Figure: The computation steps of Ulysses.

Results: Generated Test Cases

	Mutation (Ulysses)	Scenarios (TGV)
UML Mutants [#]	76	-
Max. Depth	23	30
Gen. TCs [#]	63	9
Duplicates [#]	0	0
Unique [#]	59	9
Gen. Time [min]	23	-

Testing a Java Implementation

- Implemented the Car Alarm System in Java
- Created 72 mutants with μjava
- Selected 38 relevant mutants

	Mutants	Equiv.	Pairwise Equiv.	Different Faults
SetState	6	0	1	5
Close	16	2	6	8
Open	16	2	6	8
Lock	12	2	4	6
Unlock	20	2	8	10
Constr.	2	0	1	1
Total	72	8	26	38

Efficiency of Test Cases

Surviving Implementation Mutants:

	Mutation-Based TCG	TGV
SetState	0	0
Close	0	2
Open	1	4
Lock	0	2
Unlock	0	5
Constr.	0	0
Total	1	13
Detection Rate	97%	66%

Conclusions

- Action System Semantic for UML
  - precise, compositional background models
  - concurrency!
  - straightforward IOLTS interpretation
- Tools
  - CADP: scenario-based test case generation
  - Ulysses: mutation-based test case generation
- We need both!
  - Industry wants scenarios & requirements traceability
  - Scenarios fast, but ad-hoc
  - Mutations slow, but safer
- By the way: without UML our tool is 50% faster
- Ongoing: tool improvement, semantic mutations