1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Model-based Synthesis of Reactive Planning On-line Testers

Marko Kääramees
Jüri Vain
Kullo Raiend

Tallinn University of Technology
Eliko Competence Centre
Elvior

European Union
Regional Development Fund

Investing in your future

Overview

- Scope and main idea of the work
- Workflow of testing
- Off-line preparation algorithm and example
- On-line testing algorithm and example
- Implementation and complexity issues
- Conclusions

Scope of the work

- Black box model based testing
  - tests are generated from the model
- Model is non-deterministic
  - output observability assumed
- Several test goals are tackled at the same time
  - minimizing the amount and length of the tests

Testing non-deterministic models

On-line testing is needed

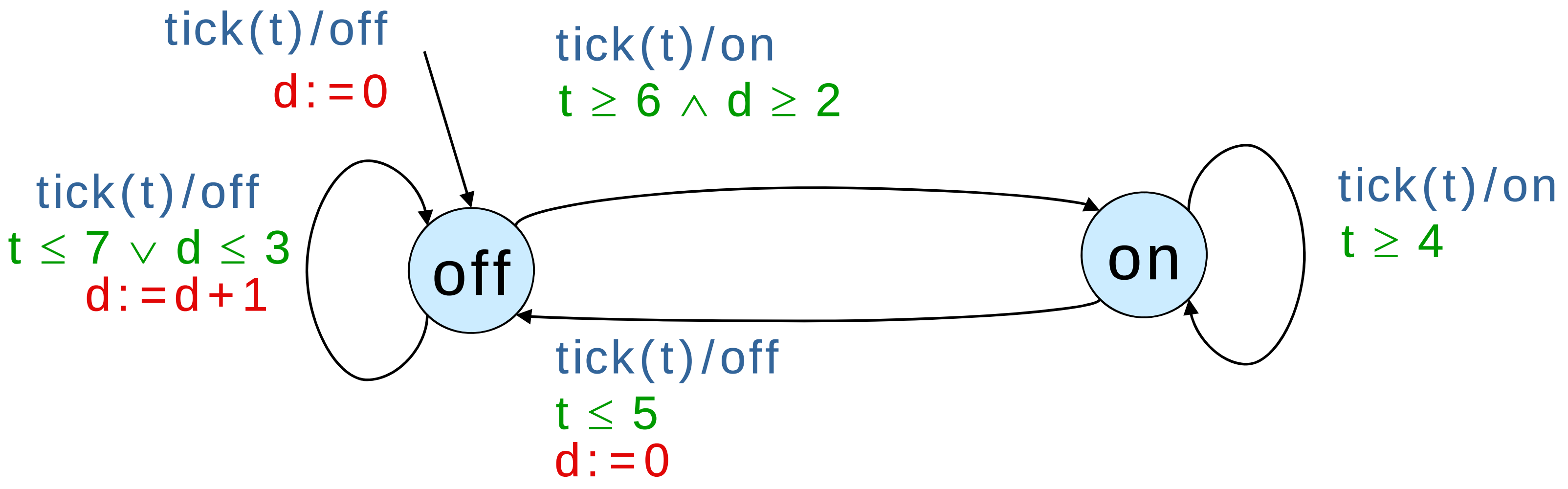
- Test cases cannot be prepared beforehand
- Tester must decide inputs during the test based on observed outputs and active goals
- Test planning is costly and not feasible on-line

Proposed solution

- Model is analysed off-line
- Result is expressed as a set of data constraints for each test goal
- Data instance generation is done on-line

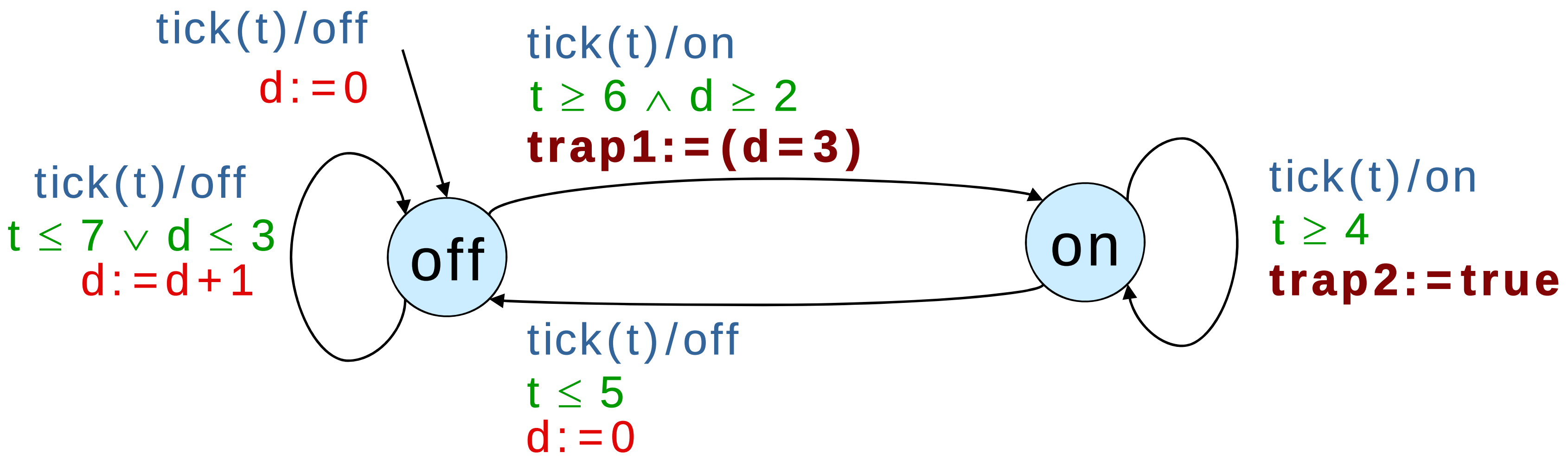
Model of SUT

- Model is given as EFSM
  - input/output, guard, update
  - input parameter t [temp] and variable d [delay]
- Requirements
  - fridge must switch off when t is 4..5
  - fridge must switch on when t is 6..7 and it has been off 20..39 seconds (tick every 10 seconds)

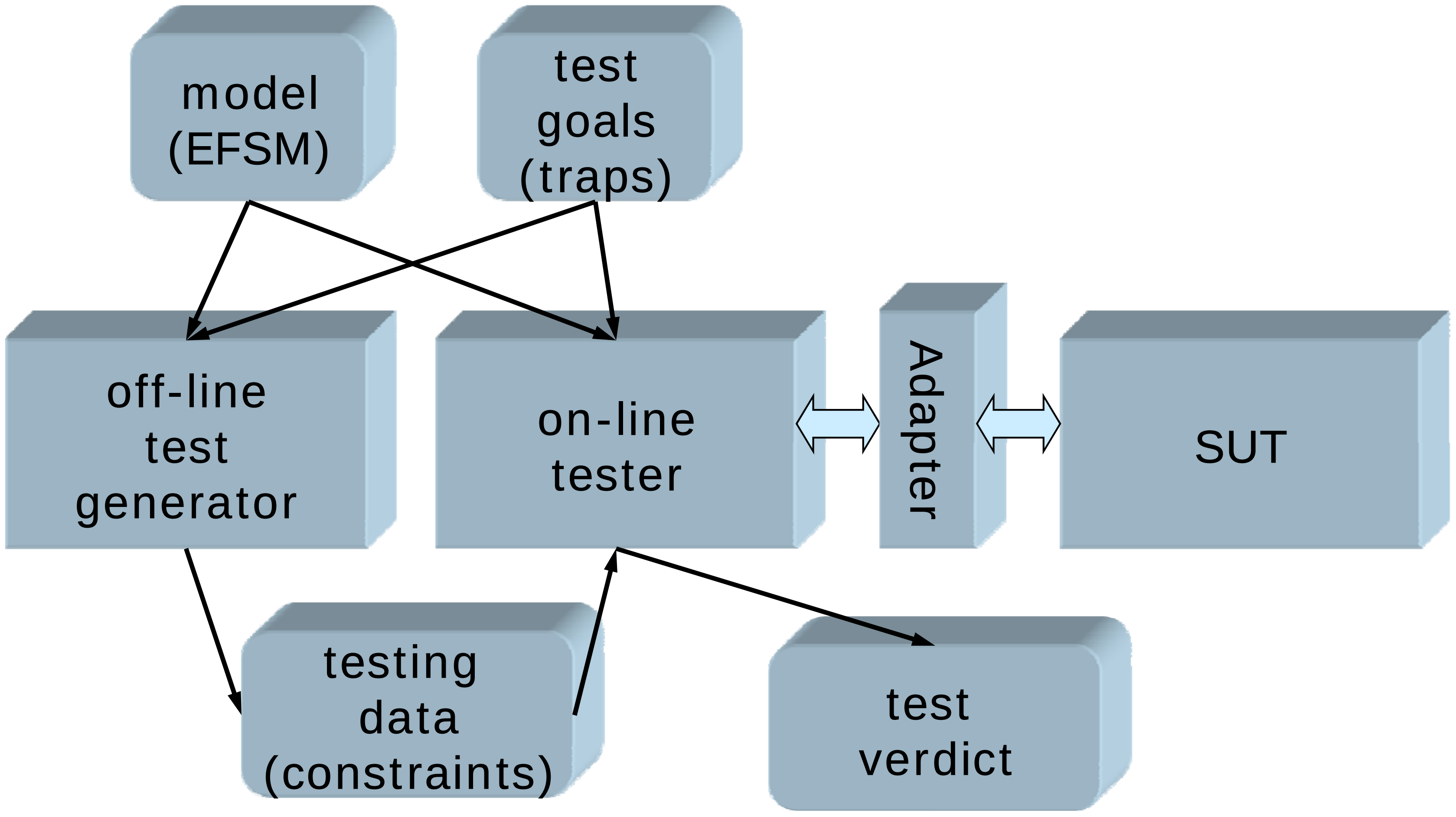


Modeling of test goals

- Test goals are expressed by traps
  - trap is a pair <transition,predicate>
  - expressed as update of trap variable in model
- Can express
  - transition coverage
  - transition sequence
  - repeated pass using auxiliary variable



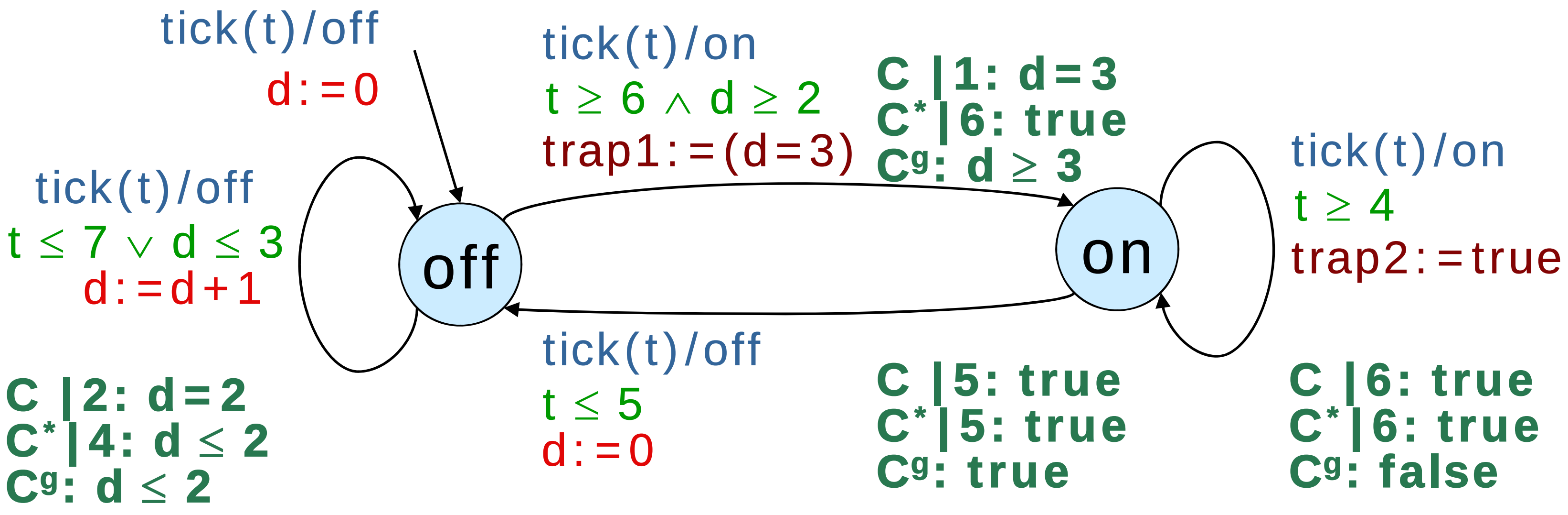
Workflow


model (EFSM)
test goals (traps)
off-line test generator
on-line tester
Adapter
SUT
testing data (constraints)
test verdict

Off-line constraint generation

Constraints for a trap (trap1 on example) generated by breath-first backwards constraint propagation algorithm:

- Constraints C|L give the condition and length for the shortest path
- Constraints $C^*|L^*$ give the condition and length for all paths up to fixpoint (or search depth)
- Constraints C^g give the condition for choosing the next transition depending on the values of variables

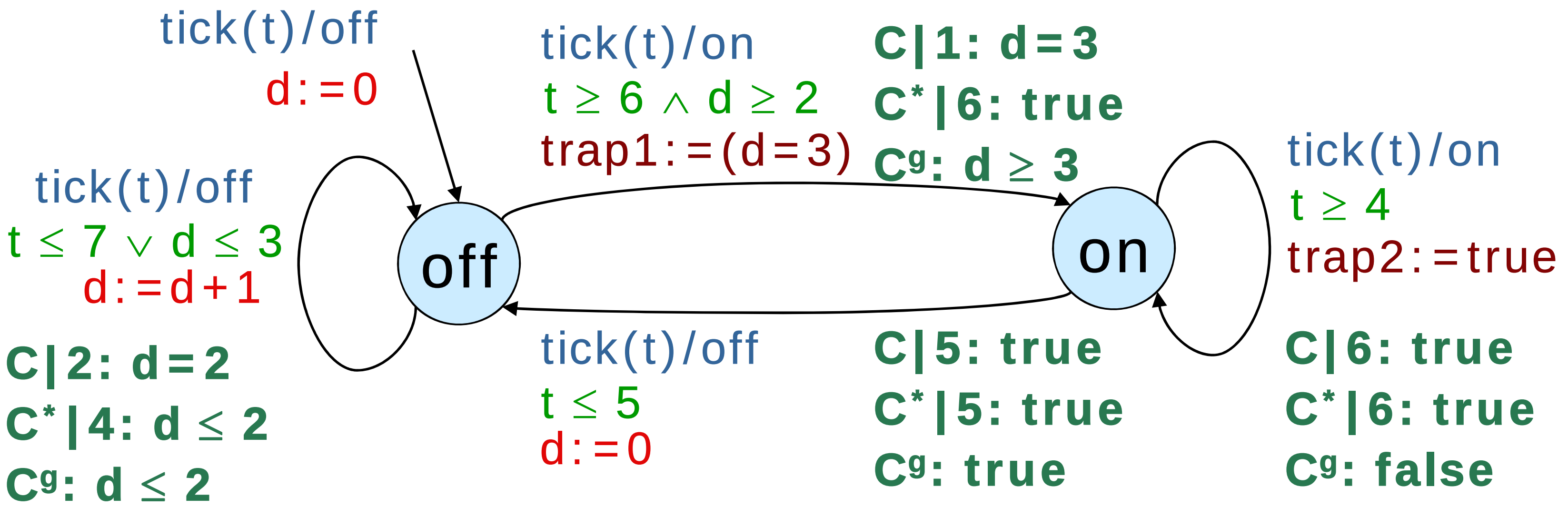


Offline algorithm for trap *tr*

```
initialise $C$ to $false$, $L$ to 0
$C^*_t = guard_t \wedge condition_{tr}$
while fixpoint or search $depth$ is reached
    for each state $s$ on the depth level do
        $C^*_s = \text{simplify}(C^{*'}_s \vee \exists I: C^*_{ti})$      // $ti$ - $t$ leaving from $s$; $I$ - $input$
        if SAT$(\neg(C^*_s \Rightarrow C^{*'}_s))$                   // $C^*_s$ changed
            $L^*_s = depth$
            if not $C_s$                                  // minimal constraint
              $C_s = C^*_s$ ; $L_s = L^*_s$
            for each transition $t$ coming to $s$
                $C^*_t = \text{simplify}(C^{*'}_t \vee guard_t \wedge wp(update_t, C^*_s))$
                record $L^*_s$, $C_t$, $L_s$ if needed
                $C^g_t = \text{simplify}(C^{g'}_t \vee (\exists I: C^*_t \wedge \neg C^*_{source(t)}))$
```

Example (on-line)

1.	tick(true):	off, d=0	
2.	tick(true):	off, d=1	
3.	tick(true):	off, d=2	
4.	tick($t < 6$):	off, d=3	
5.	tick($t \geq 6$):	on, d=3 trap1☺	off, d=4
6.	tick($t > 7$):		on, d=4
7.	tick($t > 5$):		on, d=4 trap2☺
8.	tick($t < 4$):		off, d=0



On-line algorithm (greedy)

```
while exist uncovered traps                              //at state $s$
    select nearest reachable trap $tr$                   // using SAT()
    select transition with $C^g_t$ satisfiable           // using SAT()
    select input parameters valuation by
        solving $C_t$ or $C^*_t$                         // constraint solving
    communicate the inputs to SUT
    if the output does not conform to the model          // using SAT()
        stop(test_failed)
    move to the next state
end while
stop(test_passed)
```

Implementation issues

- UPPAAL used for modelling (Uppsala & Aalborg U)

- Z3 SMT solver suite (Microsoft Research)
  - simplification of constraints
  - quantifier elimination
  - SAT solver
  - constraint solving (model generation)

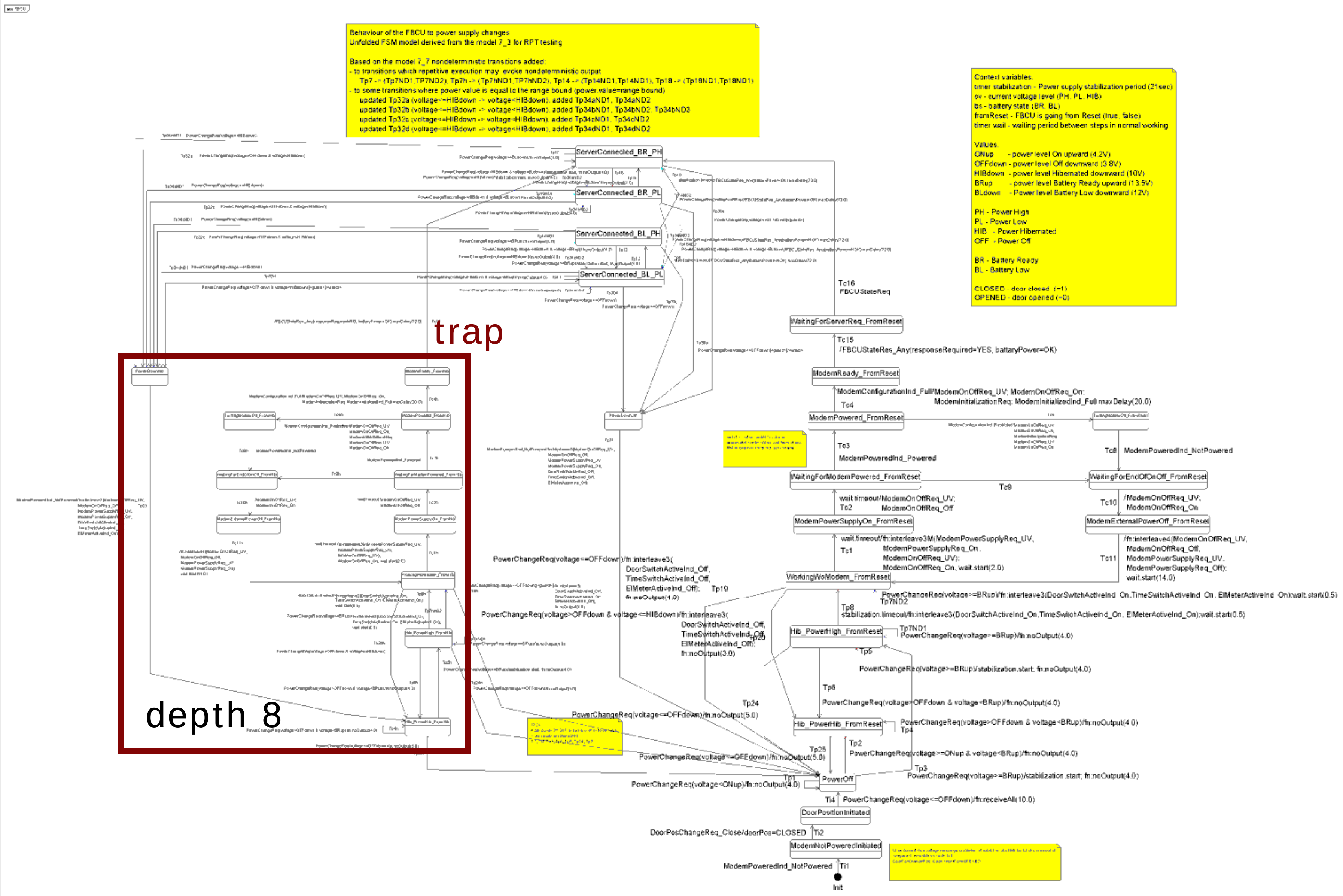
- Python scripts for parsing and constraining generation algorithm implementation

- TestCast - TTCN3 toolset (Elvior)
  - running generated TTCN3 scripts

Complexity issues

- Constraints limited to decidable theories
  - linear arithmetic (+ others supported by solver)
- Theoretical limits
  - SAT problem is NP-complete
  - decision procedures and simplification of Presburger arithmetic is double-exponential
- Practical aspects
  - number of constraints is in O(traps*transitions)
  - Z3 does a good job in SAT and simplification
- Search depth
  - complexity of the constraints depends on the structure of the model and search depth
  - search depth can be constrained off-line when the time for the SAT check needed on-line exceeds the predefined limit

Constrained search



Behaviour of the FBCU to power supply changes
Unfolded FSM model derived from the model 7_3 for RPT testing

Based on the model 7_7 nondeterministic transitions added:
- to transitions which repetitive execution may evoke nondeterministic output
  Tp7 -> (Tp7ND1,TP7ND2), Tp7h -> (Tp7hND1,TP7hND2), Tp14 -> (Tp14ND1,Tp14ND1), Tp18 -> (Tp18ND1,Tp18ND1)
- to some transitions where power value is equal to the range bound (power.value=range bound)
  updated Tp32a (voltage<=HIBdown -> voltage<HIBdown), added Tp34aND1, Tp34aND2
  updated Tp32b (voltage<=HIBdown -> voltage<HIBdown), added Tp34bND1, Tp34bND2, Tp34bND3
  updated Tp32c (voltage<=HIBdown -> voltage<HIBdown), added Tp34cND1, Tp34cND2
  updated Tp32d (voltage<=HIBdown -> voltage<HIBdown), added Tp34dND1, Tp34dND2

Context variables.
timer stabilization - Power supply stabilization period (21sec)
cv - current voltage level (PH, PL, HIB)
bs - battery state (BR, BL)
fromReset - FBCU is going from Reset (true, false)
timer wait - waiting period between steps in normal working

Values.
ONup - power level On upward (4.2V)
OFFdown - power level Off downward (3.8V)
HIBdown - power level Hibernated downward (10V)
BRup - power level Battery Ready upward (13.5V)
BLdown - Power level Battery Low downward (12V)

PH - Power High
PL - Power Low
HIB - Power Hibernated
OFF - Power Off

BR - Battery Ready
BL - Battery Low

CLOSED - door closed (=1)
OPENED - door opened (=0)

trap
depth 8
ServerConnected_BR_PH
ServerConnected_BR_PL
ServerConnected_BL_PH
ServerConnected_BL_PL
Tc16
FBCUStateReq
WaitingForServerReq_FromReset
Tc15
/FBCUStateRes_Any(responseRequired=YES, battaryPower=OK)
ModemReady_FromReset
ModemConfigurationInd_Full/ModemOnOffReq_UV; ModemOnOffReq_On;
ModemInitializationReq; ModemInitializedInd_Full maxDelay(20.0)
Tc4
ModemPowered_FromReset
Tc3
ModemPoweredInd_Powered
Tc8
ModemPoweredInd_NotPowered
WaitingForModemPowered_FromReset
Tc9
WaitingForEndOfOnOff_FromReset
wait.timeout/ModemOnOffReq_UV;
Tc2
ModemOnOffReq_Off
Tc10
/ModemOnOffReq_UV;
ModemOnOffReq_On
ModemPowerSupplyOn_FromReset
ModemExternalPowerOff_FromReset
wait.timeout/fn:interleave3M(ModemPowerSupplyReq_UV,
ModemPowerSupplyReq_On,
ModemOnOffReq_UV);
ModemOnOffReq_On, wait.start(2.0)
Tc1
Tc11
/fn:interleave4(ModemOnOffReq_UV,
ModemOnOffReq_Off,
ModemPowerSupplyReq_UV,
ModemPowerSupplyReq_Off);
wait.start(14.0)
WorkingWoModem_FromReset
PowerChangeReq(voltage<=OFFdown)/fn:interleave3(
DoorSwitchActiveInd_Off,
TimeSwitchActiveInd_Off,
ElMeterActiveInd_Off);
Tp19
fn:noOutput(4.0)
PowerChangeReq(voltage>=BRup)/fn:interleave3(DoorSwitchActiveInd_On,TimeSwitchActiveInd_On, ElMeterActiveInd_On);wait.start(0.5)
Tp7ND2
Tp8
stabilization.timeout/fn:interleave3(DoorSwitchActiveInd_On,TimeSwitchActiveInd_On, ElMeterActiveInd_On);wait.start(0.5)
PowerChangeReq(voltage>OFFdown & voltage<=HIBdown)/fn:interleave3(
DoorSwitchActiveInd_Off,
TimeSwitchActiveInd_Off,
ElMeterActiveInd_Off);
fn:noOutput(3.0)
Tp20
Hib_PowerHigh_FromReset
Tp7ND1
PowerChangeReq(voltage>=BRup)/fn:noOutput(4.0)
Tp5
PowerChangeReq(voltage>=BRup)/stabilization.start; fn:noOutput(4.0)
Tp6
PowerChangeReq(voltage>OFFdown & voltage<BRup)/fn:noOutput(4.0)
Tp24
PowerChangeReq(voltage<=OFFdown)/fn:noOutput(5.0)
Hib_PowerHib_FromReset
PowerChangeReq(voltage>OFFdown & voltage<BRup)/fn:noOutput(4.0)
Tp4
Tp25
Tp2
PowerChangeReq(voltage>=ONup & voltage<BRup)/fn:noOutput(4.0)
PowerChangeReq(voltage>=OFFdown)/fn:noOutput(5.0)
Tp3
PowerChangeReq(voltage>=BRup)/stabilization.start; fn:noOutput(4.0)
Tp1
PowerOff
PowerChangeReq(voltage<ONup)/fn:noOutput(4.0)
Ti4
PowerChangeReq(voltage<=OFFdown)/fn:receiveAll(10.0)
DoorPositionInitiated
DoorPosChangeReq_Close/doorPos=CLOSED
Ti2
ModemNotPoweredInitiated
ModemPoweredInd_NotPowered
Ti1
Init

Main results

- Tester for non-deterministic EFSM
- Efficient on-line test planning
    - supported by off-line preparation
- Off-line computation is usable also for off-line test cases generation for deterministic models
- On-line planning drives the test towards uncovered test goals resulting a test with sub-optimal length

- Future plans:
    - modelling SUT and test scenarios (goals) using hierarchical automata
    - Improvement of simplification

Offline algorithm for trap *tr*

```
**initialise** $C$ to $false$, $L$ to 0
$C^*_t = guard_t \wedge condition_{tr}$
**while** fixpoint or search $depth$ is reached
    **for** each state $s$ on the depth level do
        $C^*_s = \text{simplify}(C^{*\prime}_s \vee \exists I:$📫$C^*_{ti})$    // $ti$ - $t$ leaving from $s$; $I$ - $input$
        **if** SAT$(\neg(C^*_s \Rightarrow C^{*\prime}_s))$    // $C^*_s$ changed
            $L^*_s = depth$
            **if not** $C_s$    // minimal constraint
              $C_s = C^*_s$ ; $L_s = L^*_s$
            **for** each transition $t$ coming to $s$
                $C^*_t = \text{simplify}(C^{*\prime}_t \vee guard_t \wedge wp(update_t, C^*_s))$
                record $L^*_s$ , $C_t$ , $L_s$ if needed
                $C^g_t = \text{simplify}(C^{g\prime}_t \vee (\exists I: C^*_t \wedge \neg C^*_{source(t)}) )$
```