University of Antwerp
Faculty of Science

Model Driven Engineering Presentation

Mert Ege CAN
20212580

EXPLICIT MODELLING AND SYNTHESIS OF DEBUGGERS FOR HYBRID SIMULATION LANGUAGES (2017)

Simon Van Mierlo
Cláudio Gomes
Hans Vangheluwe

Van Mierlo, S., Gomes, C., & Vangheluwe, H. (2017, April). Explicit modelling and synthesis of debuggers for hybrid simulation languages. In Proceedings of the Symposium on Theory of Modeling & Simulation (pp. 1-12)

University of Antwerp
Faculty of Science

INTRODUCTION

Complex systems requires

1. Decomposition
2. Multiple formalisms
3. Separation of concerns

Multi Paradigm Modelling advocates the most appropriate use of

- Level of abstraction
- Formalisms

Combination of CBD and T-FSA

BACKGROUND

Statecharts (SC)

- Timed
- Reactive
- Autonomous system behaviour

Statecharts and Class Diagrams (SCCD)

- Attributes
- Methods
- SC model

BACKGROUND

Timed Finite State Automata (T-FSA)

- Timed variant of Finite State Automata
- Single clock
- Simplified version of SC
- States and Transitions
- Timed or environmental event triggering

Algorithm 1 T-FSA Operational Semantics.

function $\text{SIMTFSA}(M, s_0\ evs, \Delta t)$
    $clock \leftarrow 0$
    $state \leftarrow s_0$
    $\varepsilon \leftarrow 0$
    while $state \notin \text{FINALSTATES}(M)$ **do**
        $continue \leftarrow true$
        while $continue$ **do**
            $(evs, e_i) \leftarrow \text{POPEV}(evs, clock)$
            if $e_i = \emptyset$ **then**
                $tr \leftarrow \text{TRELAP}(M, state, \varepsilon)$
            else
                $tr \leftarrow \text{TREV}(M, state, e_i)$
            end if
            if $tr \neq \emptyset$ **then**
                $\varepsilon \leftarrow 0$
                $state \leftarrow \text{TARGET}(M, tr)$
            else
                $continue \leftarrow false$
            end if
        end while
        $clock \leftarrow clock + \Delta t$
        $\varepsilon \leftarrow \varepsilon + \Delta t$
    end while
    return $clock, state$
end function

BACKGROUND

Casual Block Diagram (CBD)

- Blocks and connections
- Inputs and one output
- Algebraic operations
- Time-sensitive operations

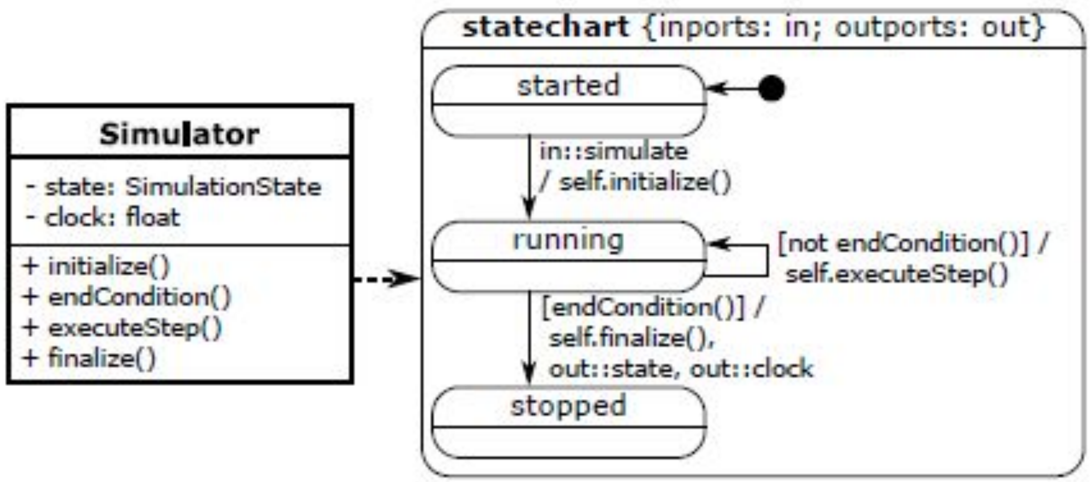
Algorithm 2 CBD Operational Semantics.

```
function SIMULATECBD(M, maxIters, Δt)
    clock ← 0
    state ← INITSIGNALS(M)
    numIters ← 0
    while numIters < maxIters do
        g ← DEPGRAPH(M, numIters)
        s ← LOOPDETECT(g)
        for c in s do
            if c = {gblock} then
                state ← COMPB(c, state)
            else
                state ← COMPL(c, state)
            end if
        end for
        clock ← clock + Δt
        numIters ← numIters + 1
    end while
    return clock, state
end function
```

MODELLING SIMULATION ALGORITHMS

Generic Simulator Template

1. Initialization
2. Execution of simulation ‘steps’
3. Finalization



MODELLING SIMULATION ALGORITHMS

Hierarchical Canonical Representation

"executeStep()" needs to be refined

Algorithm 3 Generic simulation algorithm.

```
function SIMULATE(M, params)
    initialize(params)
    while not endCondition() do
        executeStep()
    end while
    finalize()
    return getState(), getTime()
end function
```

MODELLING SIMULATION ALGORITHMS

Hierarchical Canonical Representation

Instead of having one "*Simulator*" class, four classes were proposed

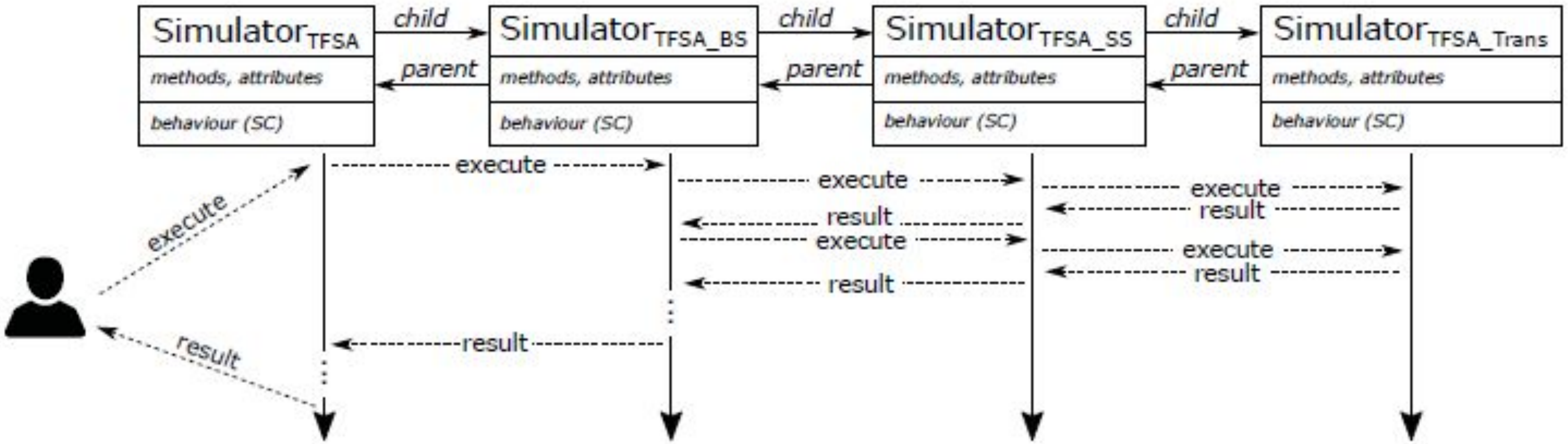


Figure 1: The hierarchical structure of the T-FSA simulator.

MODELLING SIMULATION ALGORITHMS

Debugging

Time:

- Run as-fast-as possible
- Scale-able realtime
- Pause

Control:

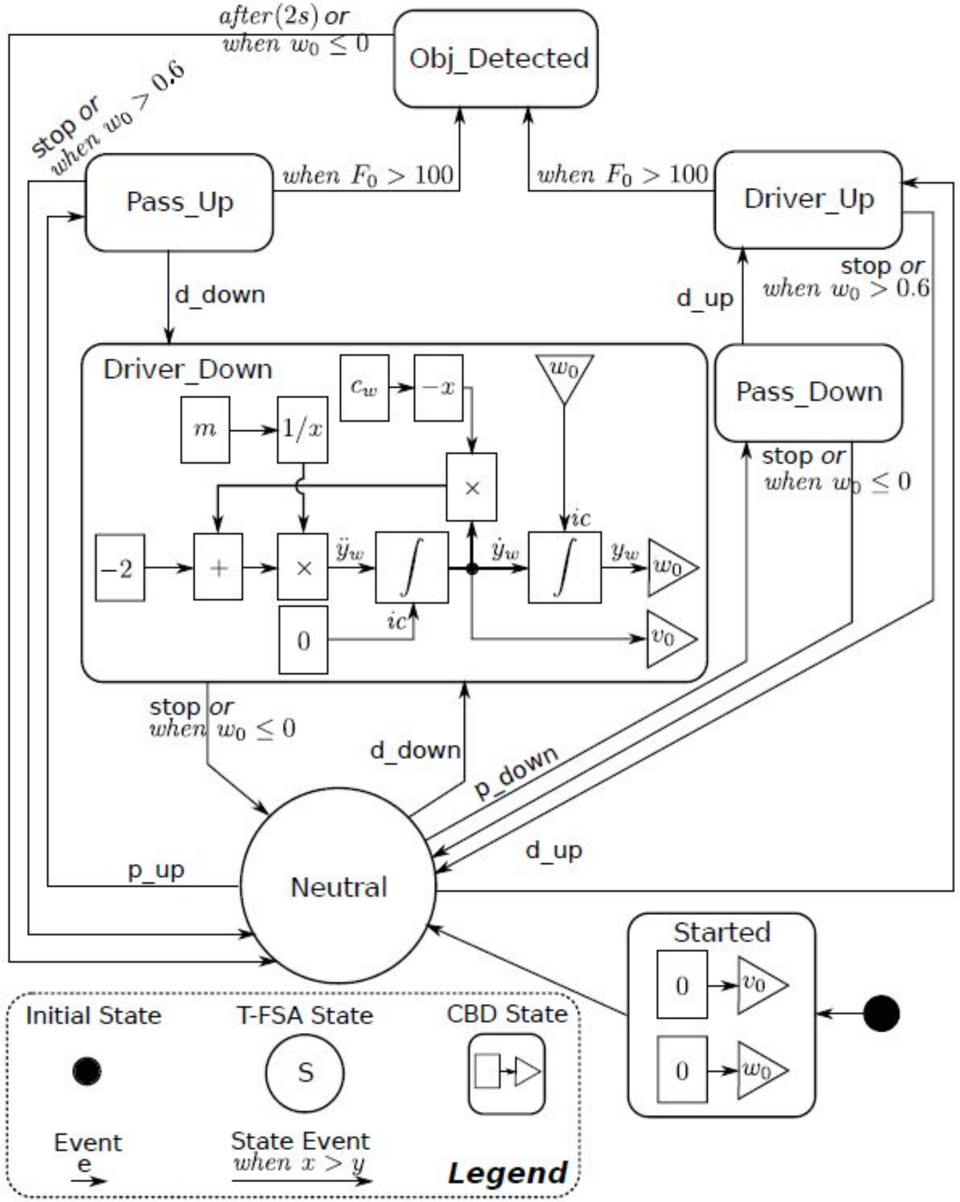
- Step through
- Big/Small step
- Breakpoint

State:

- God event
- Trigger transition
- Manual state change

HYBRID AUTOMATA

- Combination of CBD and T-FSA
- States can contain any CBD
- CBD is simulated when that state is reached
- Outgoing transition triggered based on the output of the CBD
- Boundary Crossing Condition



HYBRID AUTOMATA

T-FSA-CBD Simulator

1. Synchronization of Δt parameter
2. An outer-while loop executes the model
3. When big step starts, the algorithm checks if any CBD model defined in current state
4. After each iteration, the algorithm checks whether any state events occur
5. T-FSA small step reads next event and executes any enabled transition

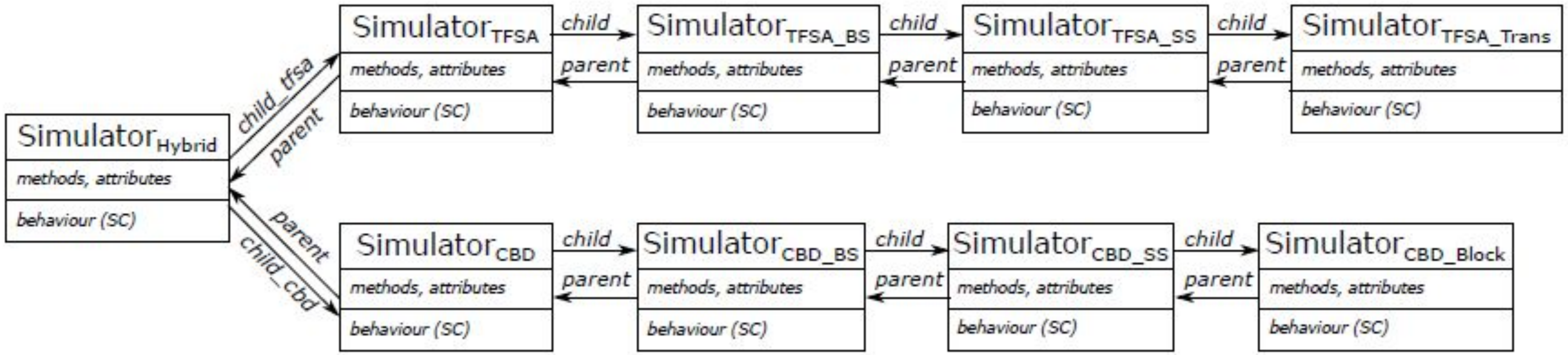


Figure 4: The hierarchical structure of the T-FSA-CBD simulator.

DISCUSSION

Hybrid Simulator satisfies the properties:

- Language Continuity
- Step Progression
- Step Synchronization

Debugging operations properties:

- Continuity
- Soundness
- Big Step-Small Step Correspondence

CONCLUSION

A novel method for implementing a debugging-featured simulator based on hybrid formalisms were presented.

The simulation algorithm is implemented by following explicit modelling using SCCD.

The deconstruction and reconstruction of CBD into states of the Statecharts were shown.

IDEAS

Model-based System Engineering combines the engineering disciplines using model-based approaches. As time and attribute based Statechart formalisms (Software Eng.) were merged with CBDs (Control Eng.) in order to form a new hybrid formalism, the proposed work can be modelled as a domain-specific modelling language. The metamodel should inherit both Statechart and CBD metamodels. Moreover, simulation-specific debugging properties were also introduced. Therefore, with this study, any suitable formalisms can be combined to model complex systems then based on this model the simulation can be executed.