MOTOR: The MODEST Tool Environment

Henrik Bohnenkamp[1], Holger Hermanns[2], and Joost-Pieter Katoen[1,3]

[1] Software Modeling and Verfication Group, Informatik 2
RWTH Aachen University, 52056 Aachen, Germany
[2] Department of Computer Science
Saarland University, D-66123 Saarbrücken, Germany
[3] Formal Methods and Tools Group, Department of Computer Science
University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract. The MODEST Tool Environment (MOTOR) is a tool to facilitate the transformation, analysis and validation of MODEST models. MODEST is a modelling language to describe stochastic real-time systems. MOTOR implements the formal semantics of MODEST and is designed to transform and abstract MODEST specifications such that analysis can be carried out by third-party tools. For the time being, a fragment of MODEST can be model-checked using CADP. The main analytical workhorse behind MOTOR is discrete-event simulation, which is provided by the MÖBIUS performance evaluation environment. We are experimenting with prototypical connections to the real-time model checker UPPAAL.

1 The MODEST Approach

The *Modeling and Description Language for Stochastic and Timed Systems* (MODEST) [2] is a specification formalism for describing stochastic real-time systems. The language is rooted in classical process algebra, *i.e.* the specification of models is compositional. Basic activities are expressed with atomic actions, more complex behaviour with constructs for sequential composition, nondeterministic choice, parallel composition with CSP-style synchronisation, looping and exception handling. A special construct exists to describe probabilistic choice. Clocks, variables and random variables are used to describe stochastic real-time aspects. All constructs and language concepts have a pleasant syntax, inspired by Promela, LOTOS, FSP, and Java. The screen shot in Fig. 2 gives an impression of the language syntax. MODEST is equipped with a structural operational semantics mapping on so-called stochastic timed automata (STA). The MODEST semantics is described in full detail in [2]. MODEST allows one to describe a very large spectrum of models, including: ordinary labelled transition systems, timed automata, probabilistic automata, stochastic automata [8], Markov decision processes, and various combinations thereof (*cf.* [2]). Remarkably, the language is designed in a way that all these models correspond to syntactic subsets of the language, and can thus be identified while parsing a MODEST specification.

With MODEST, we take a *single-formalism*, *multi-solution* approach, similar to [1]. Our view is to have a single specification that addresses various aspects of the system under consideration. This is contrary to the more common approach to construct different models to describe different aspects of a system and then analyse these models. Generally, no guarantee of consistency between these models can be given, be it for lack of a rigorous semantics or because a proper relation between the different model classes is not known. Thus, the validity of results w.r.t. the original system under study is often questionable.

Instead, with MODEST we advocate an approach to describe a system with *one* model and analyse it by extracting simpler models that are tailored to the specific properties of interest. For instance, for checking reachability properties, a possible strategy is to *distill* an automaton from the MODEST specification and feed it, *e.g.,* into an existing model checker. For carrying out an evaluation of the stochastic process underlying a MODEST specification, a discrete-event simulator can be used. This approach has the advantage that the modelling itself takes less time, since only one model has to be specified. Moreover, if the abstractions used to derive sub-models are sound, the *multi-solution* approach can ensure validity of the respective analysis results and thus significance for the modelled system.

2 MOTOR

In order to facilitate the analysis of the different models, tool support is essential. The MODEST Tool Environment (MOTOR) is a software tool that implements the MODEST semantics and is the central vehicle in the multi-solution analysis of MODEST models. The fundamental idea behind MOTOR is to simplify specifying MODEST models (e.g. by providing a macro-preprocessor), and to translate or adapt the models in a way such that the actual analysis work can be carried out by third-party state-of-the-art tools, such as PRISM [12], UPPAAL [13], or CADP [10].

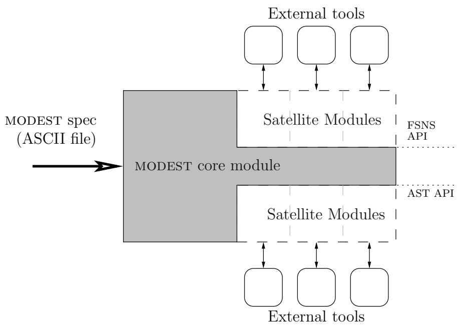


Fig. 1. The MOTOR architecture [4]

MOTOR is designed to facilitate easy access to all language features of MODEST, and thus to allow easy extraction of all imaginable model classes from a specification. The design allows straightforward extensibility of the tool. To realise this, MOTOR provides two programming interfaces, see Fig. 1: the AST API, which gives the programmer access to the abstract syntactic representation of the MODEST specification, and the FSNS API, which allows a programmer access to a first-state/next-state interface, and which allows convenient state-space exploration of the MODEST-specification. It provides the access to the STA defined by the MODEST semantics. The kernel of the tool is thus quite small: it comprises a parser and the implementation of the functionality behind the two interfaces. These two APIs enable modular design and extendability of MOTOR, the former for translation-oriented transformations, the latter for state-oriented approaches. This particular tool architecture is described in [4], and has since been actively developed. A prototype connection to the CADP tool box exists, which allows model-checking of untimed, non-probabilistic MODEST models. We are currently experimenting with another prototypical connection targeting at the UPPAAL model-checker for timed automata. The most interesting tool we connected MOTOR to is the MÖBIUS performance evaluation environment [7].

3 MOTOR and MÖBIUS

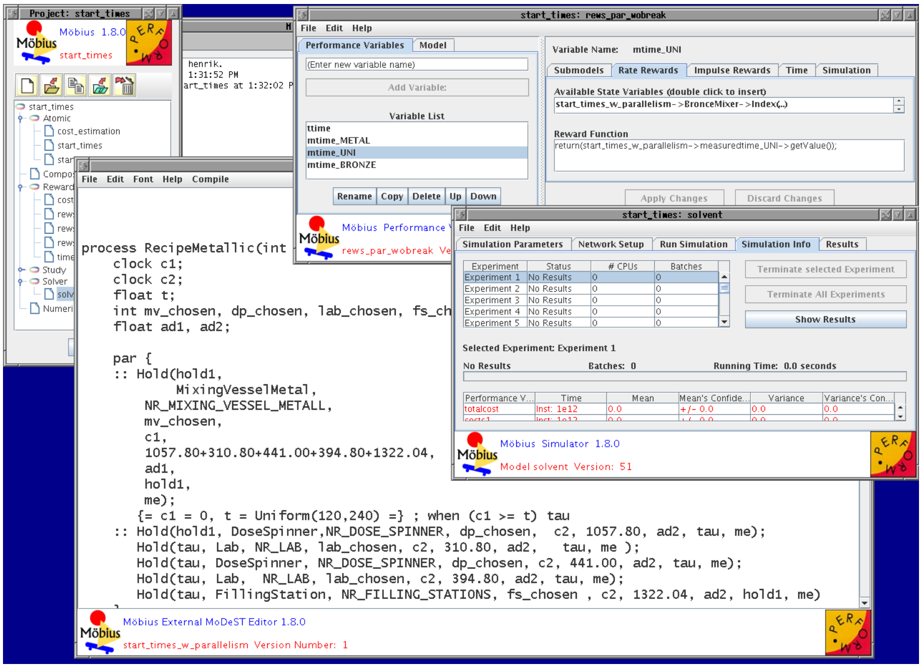
The by now most mature connection of MOTOR is the link with the MÖBIUS performability evaluation environment. MÖBIUS has been developed independently from MODEST and MOTOR at the University of Illinois at Urbana-Champaign [7]. MÖBIUS is designed as an integrated tool environment for the analysis of performability and dependability models. It allows specification of models in different formalisms, based on, for instance, Petri net-like formalisms or Markovian process algebra. The tool provides efficient discrete-event simulation capabilities and numerical solvers, such as Markov chain solvers.

The integration of MOTOR into the MÖBIUS framework is established by a direct mapping from MODEST-constructs onto the programming interface available for MÖBIUS, closely following the STA semantics as implemented in the FSNS API [4]. More concretely, a MODEST specification is interpreted as a so-called *atomic model*, the most basic model within MÖBIUS, which is made up of state variables that hold information about the state of a model and actions that are used for changing model state.

From a user perspective, the MOTOR/MÖBIUS tandem enables one to perform simulation of MODEST models, and to gather performability results. A complete simulation model in MÖBIUS consists, in addition to the atomic model, of two more sub-models. The *reward model* defines which performance measures (such as means, variances, distributions etc.) are to be estimated with the simulation. These rewards are based on the global variables of the atomic MODEST model. The *study model* defines intervals or sets of values as parameters for which the MODEST model is to be simulated. Experiment parameters are declared inside MODEST as special *external* constants. For each set of parameters an *experiment*, *i.e.* simulation process is started (in parallel or sequentially, depending on the configuration and number of available processors), where the external constants of the MODEST model are preset with the respective values defined in the study model.

Atomic MODEST models are entered in a dedicated MÖBIUS text-editor. MOTOR is called from MÖBIUS to translate the model into a C++ program, which is then compiled and linked together with an implementation of the MODEST semantics and the simulator libraries of MÖBIUS. Fig. 2 shows a screen shot of the different MÖBIUS editors, the MODEST editor in the center.

The reason to choose simulation as the prime analysis method of MODEST models to integrate into MOTOR is that simulation covers the largest language



```
henrik.
1:31:52 PM
art_times at 1:32:02 P
```

(Enter new variable name)

Variable List

ttime
mtime_METAL
mtime_UNI
mtime_BRONZE

Variable Name: mtime_UNI

Available State Variables (double click to insert)
start_times_w_parallelism->BronceMixer->Index(...)

Reward Function
return(start_times_w_parallelism->measuredtime_UNI->getValue());

```
process RecipeMetallic(int
    clock c1;
    clock c2;
    float t;
    int mv_chosen, dp_chosen, lab_chosen, fs_ch
    float ad1, ad2;

    par {
    :: Hold(hold1,
             MixingVesselMetal,
        NR_MIXING_VESSEL_METALL,
        mv_chosen,
        c1,
        1057.80+310.80+441.00+394.80+1322.04,
        ad1,
        hold1,
        me);
       {= c1 = 0, t = Uniform(120,240) =} ; when (c1 >= t) tau
    :: Hold(hold1, DoseSpinner,NR_DOSE_SPINNER, dp_chosen,  c2, 1057.80, ad2, tau, me);
       Hold(tau, Lab, NR_LAB, lab_chosen, c2, 310.80, ad2,   tau, me );
       Hold(tau, DoseSpinner, NR_DOSE_SPINNER, dp_chosen, c2, 441.00, ad2, tau, me);
       Hold(tau, Lab,  NR_LAB, lab_chosen, c2, 394.80, ad2, tau, me);
       Hold(tau, FillingStation, NR_FILLING_STATIONS, fs_chosen , c2, 1322.04, ad2, hold1, me)
```

Experiment	Status	# CPUs	Batches
Experiment 1	No Results	0	0
Experiment 2	No Results	0	0
Experiment 3	No Results	0	0
Experiment 4	No Results	0	0
Experiment 5	No Results	0	0

Selected Experiment: Experiment 1

No Results Batches: 0 Running Time: 0.0 seconds

Performance V...	Time	Mean	Mean's Confide...	Variance	Variance's Con...
totalcost	Inst: 1e12	0.0	+/- 0.0	0.0	0.0

Fig. 2. MÖBIUS with MODEST editor

fragment of MODEST: the only concept that can not be supported by simulation is nondeterminism, in particular of delay durations and non-deterministic choice between actions. We exclude the former by assuming maximal-progress with respect to delays. We do not restrict action nondeterminism, since it is a convenient modelling instrument. However, no mechanisms, like a well-specified-check [9], is implemented yet to ensure validity of the simulation statistics when action nondeterminism is present.

Given that discrete-event simulation (DES) is supported by many tools, one may question what the benefits are of using MODEST. The main difference with existing simulation notations is that MODEST has a formal semantics. Consequently, the underlying stochastic model for simulation is well-defined and obtained simulation results—given that DES is a well-understood technique—is trustworthy. In commercially available simulation tools it is often unclear how simulation models are obtained from the modelling language. This is recently witnessed in e.g.,. [6] by obtaining significant different results from models that were fed into different simulators.

4 Status and Availability

MOTOR and its connection to MÖBIUS is mature and has been tested in a number of non-trivial case studies. In [11], it has been used for reliability analysis of the upcoming European Train Control System standard. In [3], it has been applied to the analysis of an innovative plug-and-play communication protocol, which has led to a patent application of our industrial partner. In [5], MOTOR has been used for the optimisation of production schedules, in combination with timed automata-based schedule synthesis with UPPAAL.

MOTOR is available as source code from `http://www.purl.org/net/motor` under the GPL license. MÖBIUS is freely available for educational and research purposes from `http://www.mobius.uiuc.edu/`. MOTOR can be installed as an add-on package into the MÖBIUS installation directory.

References

1. M. Bernardo, W.R. Cleaveland, S.T. Sims, and W.J. Stewart. TwoTowers: A tool integrating functional and performance analysis of concurrent systems. In *Proc. FORTE/PSTV 1998*, pages 457–467. Kluwer, 1998.
2. H. Bohnenkamp, P.R. D'Argenio, H. Hermanns, and J.-P. Katoen. Modest: A compositional modeling formalism for real-time and stochastic systems. *IEEE Trans. Soft. Eng.*, 32(10):812–830, 2006.
3. H. Bohnenkamp, J. Gorter, J. Guidi, and J.-P. Katoen. Are you still there? — A lightweight algorithm to monitor node presence in self-configuring networks. In *Proc. DSN 2005*, pages 704–709. IEEE CS Press, June 2005.
4. H. Bohnenkamp, H. Hermanns, J.-P. Katoen, and J. Klaren. The MoDeST modeling tool and its implementation. In *Proc. TOOLS 2003*, LNCS 2794. Springer, 2003.
5. H. Bohnenkamp, H. Hermanns, J. Klaren, A. Mader, and Y. Usenko. Synthesis and stochastic assessment of schedules for lacquer production. In *Proc. QEST '04*. IEEE CS Press, 2004.
6. D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of MANET simulators. In *ACM Workshop On Principles Of Mobile Computing*, pages 38–43, 2002.
7. D. Daly, D. D. Deavours, J. M. Doyle, P. G. Webster, and W. H. Sanders. Möbius: An extensible tool for performance and dependability modeling. In *Proc. TOOLS 2000*, LNCS 1786. Springer, 2000.
8. P. R. D'Argenio and J.-P. Katoen. A theory of stochastic systems. Part I. Stochastic automata. *Inf. & Comp.*, 203:1–38, 2005.
9. D. D. Deavours and W. H. Sanders. An efficient well-specified check. In *Proceedings PNPM '99*, pages 124–133. IEEE Computer Society, 1999.
10. H. Garavel. OPEN/CÆSAR: An open software architecture for verification, simulation, and testing. In *Proc. TACAS '98*, volume 1384 of *LNCS*, 1998.
11. H. Hermanns, D. N. Jansen, and Y. S. Usenko. From StoCharts to MoDeST: a comparative reliability analysis of train radio communications. In *Proc. WOSP '05*. ACM Press, 2005.
12. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM. In *Proc. TACAS '02*, LNCS 2280. Springer, 2002.
13. Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, October 1997.