SMART: Stochastic Model-checking Analyzer for Reliability and Timing

G. Ciardo¹  R. L. Jones, III¹  R. M. Marmorstein¹  A. S. Miner²  R. Siminiceanu¹

¹Department of Computer Science, College of William and Mary  ²Department of Computer Science, Iowa State University

SMART (http://www.cs.wm.edu/~ciardo/SMART/) is a software package integrating logic and stochastic modeling formalisms into a single environment. Models expressed in different formalisms can be combined in the same study. To study logical behavior, both explicit and symbolic state-space generation techniques, as well as CTL model-checking algorithms, are available. To study stochastic and timing behavior, both explicit and Kronecker-based numerical solution approaches are available. Since SMART is intended as an industry and research tool, it is written in a modular way that allows for easy integration of new formalisms and solution algorithms.

Language and features. SMART uses a strongly-typed declarative language with four predefined types, bool, int, real, and string, which can be further composed as sets, arrays, and aggregates. A type can be modified by a nature: const, for non-stochastic quantities; ph, for discrete or continuous phase-type random variables; rand, for arbitrary random variables. Combining ph types produces a ph type if phase-type distributions are closed under that operation; otherwise, SMART considers the result as generally distributed. Syntactically, objects defined in SMART are functions, possibly recursive, that can be overloaded. Arrays are declared using for statements, and can be multi-dimensional. Fixed-point iterations are specified using converge statements, which stop iterating when subsequent values for the objects declared in them differ by less than ε. converge and for statements can be nested. Stochastic processes indexed by time can be defined. Currently, the dtmc (discrete-time Markov chains), ctmn (continuous-time Markov chains), and spn (stochastic Petri nets) formalisms are implemented. The type of stochastic process underlying an spn is determined automatically according to the distributions specified for the transitions. Arrays of objects (e.g., states, places) are allowed. The design of SMART allows for relatively easy addition of new formalisms.

SMART implements various state-of-the-art solution algorithms that can be used for logical and stochastic analysis of large and complex models.

State-space generation and CTL model checking. The generation and storage of the state space S is a major component of any state-space based solution technique, including model checking. We have implemented several strategies, based on multi-valued decision diagrams (MDDs), that achieve much better time and memory efficiency than traditional BDD-based approaches. SMART can generate extremely large state spaces, usually in minutes, and compute the results of CTL model-checking queries.

Markov and non-Markov models. SMART provides the ability to describe discrete- or continuous-time Markov behavior, resulting in a DTMC or CTMC, which SMART can solve numerically. If both behaviors are present, the result can be a semi-regenerative model, which SMART can solve numerically for steady-state, or a generalized semi-Markov process, which SMART can study using either batch means or regenerative simulation.

Kronecker-based numerical solution. SMART provides solution methods based on a Kronecker expression of the transition rate matrix R of the CTMC underlying a continuous Markov model. SMART also offers a particularly efficient data structure that combines the idea of decision diagrams with that of Kronecker algebra: matrix diagrams. When this option is exercised, the computational overhead normally required for Kronecker-based approaches is significantly lower. These methods are quite effective in reducing the memory requirements, and allow SMART to solve models having over one order of magnitude more states than with the traditional sparse-matrix approach.

Approximations. Using MDDs for S and Kronecker encodings for R, SMART can compactly represent an underlying CTMC. Then, SMART can use these compact representations to compute approximate stationary performance measures. The technique performs K approximate aggregations, where each aggregation is based on the structure of the MDD representing S. Since each aggregation may depend on the probabilities computed for the other aggregations, fixed-point iterations are used to break cyclic dependencies. For models with product-form solutions, the results are exact; in general, the results are quite accurate.