The reduction algorithm preserves safety and liveness properties of LTL (without the 'nexttime operator'), and is compatible with the newly introduced LTL verification capability [3]. Recently, parts of the algorithm have been formally verified [1] using the theorem prover HOL.

References

[1] C.T. Chou, D. Peled, Formal Verification of a Partial-Order Reduction Technique for Model-Checking, AT&T technical report.

[2] P. Godefroid, P. Wolper, A Partial Approach to Model Checking, 6th LICS, 1991, Amsterdam, 406–415, also in Information and Computation 110(2), 305–326.

[3] R. Gerth, D. Peled, M. Vardi, P. Wolper, Simple On-the-fly Automatic Verification of Linear Temporal Logic, Protocol Specification Testing and Verification, 1995, Warsaw, Poland,

[4] G. J. Holzmann, Design and Validation of Computer Protocols, Prentice Hall Software Series, 1992.

[5] G. J. Holzmann, D. Peled, An Improvement in Formal Verification, $7th$ International Conference on Formal Description Techniques, Berne, Switzerland, 1994, Chapmann & Hall, 197–211.

[6] L. Lamport, What good is temporal logic, in R.E.A. Mason (Ed.), Information Processing 83, Elsevier Science Publishers, 1983, 657-668.

[7] D. Peled, Combining Partial Order Reductions with On-the-fly Model-Checking, 6th International Conference on Computer Aided Verification, Stanford, CA, USA, 1994, LNCS 818, 377–390.

[8] A. Valmari, Stubborn sets for reduced state space generation, $10th$ International Conference on Application and Theory of Petri Nets, Vol. 2, 1–22, Bonn, 1989.

Partial Order Reduction of the state space

Gerard J. Holzmann and Doron Peled
AT&T Bell Laboratories
600 Mountain Avenue, Murray Hill, NJ 07974, USA

Partial order reduction for model-checking [7, 2, 8] is an approach to reduce the state space explosion caused when verifying concurrent programs and protocols. It is based on the observation that in many cases the relative order of concurrent actions of a program is unimportant to the checked property. By fixing some arbitrary order, and avoiding all possible orders, a significant gain in memory and time can be achieved in many cases. Sequences that are equivalent up to the order of concurrent actions are grouped together, and the reduction algorithm guarantees to generate a state space which contains at least one sequence per equivalence class.

We describe the new version of SPIN [4], which incorporates a partial order reduction technique [5, 7]. Experimental results show significant reductions. The partial order reduction affects how a subset of the successors are selected from each state, to guarantee that at least one sequence per equivalence class is generated. Care is taken to minimize the time and space used to calculate these subsets. Thus, achieving the reduction with least overhead.

When compiling a PROMELA specification (the SPIN specification language), each local state is annotated with information that can be used later by the reduction algorithm. Then, when constructing the reduced state space, it is possible to decide whether the posible transitions that can be taken from a given state can form a good subset. This correspond to simple cases like:

- All the transitions possible from the current local state are local to the process (i.e., involving no communications and no global variables).
- The transitions include communications, but each included send is not blocked because its queue is full, and each included receive is not blocked because its queue is empty.

In addition, the implementation checks that the chosen subset of immediately executable transitions does not close a cycle If no such subset exists, all the immediately executable transitions are taken.