Discrete Time Promela and Spin

Dragan Bošnački * and Dennis Dams **

Dept. of Computing Science, Eindhoven University of Technology
PO Box 513, 5600 MB Eindhoven, The Netherlands
fax: +31 40 246 3992, e-mails: {dragan,wsindd}@win.tue.nl

Abstract. Spin is software package for verification of concurrent systems. The formal models of the systems that are verified, are built in Promela - Spin's input language. We present an extension of Promela and Spin with discrete time that provides an opportunity to model systems which correct functioning crucially depends on timing parameters. The new version of the tool is completely compatible with all the features the standard package, including partial order reduction. We have tested the prototype tool on several applications known in the verification literature and the first results are very promising.

The Spin Model Checker. Originally, Promela and Spin have been developed for analysis and validation of communication protocols [10]. The language syntax is derived form C, but also uses the denotations for communications from Hoare's CSP and control flow statements based on Dijkstra's guarded commands. The full presentation of Promela and Spin is beyond the scope of this paper and we suggest [10] as a reference to the interested reader. Following [11] we only give a short overview of the language and validation tools. In Promela, the system components are modeled as *processes* that can communicate via *channels* either by buffered message exchanges or rendez-vous operations, and also trough shared memory represented as global *variables*. The execution of actions is considered asynchronous and interleaved, which means that in every step only one *enabled* action is performed and without any additional assumptions of the relative speed of the process execution.

Given as an input a Promela model Spin can do random or interactive simulations or to generate a C program that performs a validation of the system by scanning the state space. The validation with the compiled C program is done in separate phases. The one phase is validation of the *safety* properties (absence of deadlock, unspecified message receptions, invalid end states and assertions) trough a search of the state space. Verification of most of the *liveness* properties can be handled by detection of progress and acceptance cycles. The never-claims constructs are the most general way to verify the system by providing the possibility to express properties as linear-time temporal logic (LTL) formulae. All kinds of errors are reported by saving the trace of the actions leading to an invalid state or cycle. The erroneous sequence can be repeated as a guided simulation. The simplest way to do scanning of the state space is trough an exhaustive search.

* On leave from the Insitiute of Informatics, Faculty of Natural Sciences and Mathematics, University "Sts. Cyril and Methodius", Skopje, Macedonia. Supported by EC ESPRIT LTR Project No. 23498 (VIRES).

** Supported by the Netherlands Computer Science Research Foundation (SION).

For large state spaces some more sophisticated methods are used as state-vector compressing, partial-order reduction and bit-state hashing. The last technique only gives approximate results in a sense that there is only a guarantee that some part of the state space is covered.

Xspin is a graphical interface for Spin.It provides an integrating windowing environment for writing Promela models and carrying out virtually all Spin functions in a user friendly manner. The outputs are displayed in various ways, among which are Message Sequence Charts.

Discrete Time Extensions of Promela and Spin. In the time model that is used in discrete time Promela and Spin the time is divided into slices indexed by natural numbers. The actions are then framed into those slices, obtaining in that way a good *quantitative* estimation for the intervals between the events belonging to different slices. Within a slice however, we can only have the qualitative relation between the events, as in the time free case. The elapsed time between the events is measured in ticks of a global digital clock which is increased (decreased) by one with every such a tick. The actions between two consecutive ticks have the same integer time stamp and only a qualitative time ordering between them can be established. (Such a time model corresponds to *fictitious-clock model* from [1] or *digital-clock model* from [9].)

For modeling of timing features the standard Promela is extended with a new variable type `timer` corresponding to discrete time countdown timers. Two new additional statements, `set` and `expire`, that operate on `timer`s are added. Their syntax is `set(tmr, val)` and `expire(tmr)`, where `tmr` is a `timer` variable and `val` is of type `short` integer. The two basic statements are sufficient for modeling all timing features of the system. For instance, a time interval of `val` clock ticks is denoted in a straightforward way by assigning the value `val` to a timer with `set` and waiting with `expire` until its value becomes 0. Having `set` and `expire` it is easy to derive new compound statements as Promela macro definitions. For example, using iteration with guarded commands an unbounded nondeterministic delay can be modeled. This is necessary for modeling of actions that can happen in any time slice. In a similar way a bounded delay up to `val` clock ticks can be realized, as well as delay within an time interval. The two basic statements and the macros that implement the timing features are defined in `dtime.h` header file which is included at the begining of each discrete time Promela model. The extended tool is fully compatible with the standard package and all properties that are on Spin's repertoir can be also validated for discrete-time models. Besides qualitative properties, by using in the assertions and LTL formulae boolean expressions on timer values a broad range of quantitative timing properties can be verified.

Discrete-time Promela models can be regarded as parallel compositions (networks) of *automata with timers* from [6] assuming discrete-time instead of dense-time model, or their equivalent *timed automata* from [1].

Experimental Results. We have tested the implementation on various models known in the literature like Train Gate Controller, Seitz Circuit [15], Parallel Acknowledgment with Retransmission Protocol [14, 12] , Bounded Retransmission Protocol [7, 5, 4]. We give the results of Fischer's mutual exclusion protocol, which has become a standard

benchmark for tools that implement timing. The version of the protocol that was verified is a translation of the same model written in Promela with real (dense) time of [15], with the same timing parameters. The obtained results for the verification of the mutual exclusion property are shown in the table below (N is the number of processes). All tests were performed on a Sun SPARC Station 5 with 64 MBytes of memory.Besides partial order reduction (POR) we used as an additional option *minimized automata*, a technique for reduction of the state space recently included in the standard Spin distribution. In the *options* column of the table, "n", "r" and "ma" denote verifications without POR, with POR, and with POR together with minimized automata, respectively.

N	*option*	*states*	*transitions*	*memory* [MB]	*time* [s]
2	n	528	876	1.453	0.1
	r	378	490	1.453	0.1
	ma	378	490	0.342	0.2
3	n	8425	10536	1.761	0.8
	r	3813	4951	1.596	0.4
	nr	3813	4951	0.445	0.3
4	n	128286	373968	7.085	15.5
	r	34157	44406	3.132	2.8
	ma	34157	44406	0.650	33.7
5	r	288313	377032	17.570	36.2
	ma	288313	377032	1.059	332.5
6	ma	2.35×10^6	3.09×10^6	2.118	6246.1

As expected, the state space growth is exponential and we were able to validate the model without using POR up to 4 processes. For 6 processes even POR was not enough and it had to be strengthen with minimized automata. The profit from POR is obvious and even becomes more evident as the number of processes grows. While for $N = 2$ the number of states is reduced to 72% compared to the one in the full state space and the transitions are reduced to 56% of the original number, for $N = 4$ the reduction increases to 27% and 12% for states and transitions, respectively. It is difficult to compare our implementation with the related tools (e.g. [2], [13]), mainly because they are based on different time model. Although often said to be of the same thoretical complexity as dense time (e.g. [1]), it seems that in practice discrete time often shows an advantage [2]. For instance, having in mind that the property that was checked was a qualitative one, for which discrete time (digital clocks) suffices [9], one can safely say that after $N = 4$ our implementation has better performance in the case of Fischer's protocol compared to [15]. In fact, for $N = 5$ the validator from [15] runs out of memory. Obviously, POR is the decisive factor, because without POR our implementation is also incapable to handle cases for $N > 4$.

An encouraging result is the successful validation of Bounded Retransmission Protocol, which is only a slightly simplified version of an industry protocol used by Philips. The version of the protocol from [5] was verified with less than 5 MB of memory, and our verification has the advantage that it can be completely done in Spin, unlike the one in [5] that uses combination of Uppaal (for the timing properties) and Spin (for consistency of the specification).

Currently, we are working on an extension of Promela and Spin with dense time based on region automata [1], using essentially the same approach as for discrete time. Another possible direction for further development of the tool is modeling of an interesting class of linear hybrid systems representable by discrete-time rectangular hybrid automata [8]. The first results obtained by the prototype (implemented entirely as Promela macro definitions, without any change of the Spin's source code) are promising [3].

Availability. The tool, together with some additional information and several examples, is available on request from the autohors, from `http://www.tue.nl/~dragan`, or via anonymous ftp from `ftp.win.tue.nl/pub/techreports/dragan/dtspin`. The discrete time extension is portable to all the platforms as the standard Spin distribution - i.e. Unix systems and Windows 95/Windows NT PC's.

References

1. Alur, R., Dill, D.L., *A Theory of Timed Automata*, Theoretical Computer Science, 126, pp.183-235, 1994.
2. Alur, R., Kurshan, R.P., *Timing Analysis in Cospan*, Hybrid Systems III, LNCS 1066, pp.220-231, Springer, 1996.
3. Bošnački, D., *Towards Modelling of Hybrid Systems in Promela and Spin*, to appear in Proc. of ERCIM Formal Methods in Industrial Critical Systems (FMICS) workshop, 1998
4. Dams, D., Gerth, R., *Bounded Retransmission Protocol Revisited*, Second International Workshop on Verification of Infinite Systems, Infinity, 1997
5. D'Argenio, P. R., Katoen, J.-P., Ruys, T., Tretmans, J., *The Bounded Retransmission Protocol Must Be on Time!*, TACAS'97, 1997
6. Dill, D., *Timing Assumptions and Verification of Finite-State Concurrent Systems*, CAV'89, LNCS 407, Springer, 1989
7. Groote, J. F., van de Pol, J., *A Bounded Retransmission Protocol for Large Data Packets*, in Wirsing, M., Nivat, M., ed., Algebraic Methodology and Software Technology, LCNS 1101, pp. 536-550, Springer-Verlag, 1996
8. Henzinger, T. A., Kopke, P. W., *Discrete-Time Control for Rectangular Automata*, Proceedings of the 24th International Coloquim on Automata, Languages and Programming (ICALP 1997), LNCS 1256, pp. 582-593, Springer-Verlag, 1997
9. Henzinger, T. A., Manna, Z., Pnueli, A., *What good are digital clocks?*, Proceedings of the ICALP'92, LNCS 623, pp.545-558, Spriger-Verlag,1992.
10. Holzmann, G. J., *Design and Validation of Communication Protocols*, Prentice Hall, 1991. Also: `http://netlib.bell-labs.com/netlib/spin/whatispin.html`
11. Kars, P., *Formal Methods in the Design of Storm Surge Barrier Control System*, Hand-outs of School on Embedded Systems, Veldhoven, The Netherlands, 1996
12. Klusener, A. S., *Models and Axioms for a Fragment of Real Time Process Algebra*, Ph. D. Thesis, Eindhoven University of Technology, 1993
13. Larsen, K. G., Pettersson, P., Yi, W., *UPPAAL: Status & Developments*, Computer Aided Verification CAV 97, LNCS 1254, pp.456-459, Springer-Verlag, 1992.
14. Tanenbaum, A., *Computer Networks*, Prentice Hall, 1989
15. Tripakis, S., Courcoubetis, C., *Extending Promela and Spin for Real Time*, TACAS '96, LCNS 1055, Springer Verlaag, 1996

This article was processed using the LaTeX macro package with LLNCS style