The ULg Partial-Order Package for SPIN

Patrice Godefroid
AT&T Bell Laboratories
1000 E. Warrenville Road
Naperville, IL 60566, U.S.A.
god@research.att.com

Abstract

This document presents an overview of the Partial-Order Package for SPIN developed at the University of Liège (ULg) in 1992 – 1994. The current version of the ULg Partial-Order Package (Version 3.0) is available free of charge for educational and research purposes by anonymous ftp from ftp.montefiore.ulg.ac.be in the /pub/po-package directory.

1 Introduction

SPIN is an automated validation system for communication protocols modeled in the *Promela* language [Hol91]. SPIN checks properties of such systems by exploring a global state graph, called *state space*, representing the combined behavior of all concurrent components in the system. This is done by recursively exploring all successor states of all states encountered during the search, starting from a given initial state, by executing all enabled transitions in each state. Of course, the number of visited states can be very large: this is the well-known *state-explosion problem*, which limits the applicability of state-space exploration techniques.

Partial-Order methods [God90, Val90] are state-space exploration techniques that can avoid parts of the state explosion due to the exploration of all possible interleavings of concurrent transitions. Precisely, given a property, partial-order methods explore only a reduced part of the global state space that is provably sufficient to check the given property. The difference between the reduced and the global state spaces is that all interleavings of concurrent events are not systematically represented in the reduced one. See [God94] for a complete survey of these methods.

Given any Promela program, the ULg Partial-Order Package explores only a reduced part of the global state space of the program that is sufficient for checking deadlocks, unreachable code, and assertion violations. Specifically, the partial-order package includes the implementation of a selective search using persistent sets [GP93], sleep sets [GW93], and the "proviso" introduced in [HGP92]. We refer the reader to [God94] for a detailed presentation of these algorithms and of their correctness proofs.

Using the Partial-Order Package decreases the resource requirements of the state-space exploration performed by SPIN: the memory and time requirements are usually much smaller when using the Package than without using it.

The reduction obtained depends on the coupling between the processes in the concurrent system. When the coupling is very tight, partial-order methods yield no reduction, and the selective search becomes equivalent to a classical exhaustive search. When the coupling between the processes is very loose, the reduction can be very impressive. For most realistic examples, partial-order methods provide a significant reduction of the memory and time requirements needed to verify protocols. Therefore, they broaden the applicability of verification by state-space exploration to more complex protocols.

2 Using the Partial-Order Package

To install the partial-order package, proceed as follows.

1. In a new directory, copy the files README and po-pack.tar.Z available by anonymous ftp from ftp.montefiore.ulg.ac.be in the /pub/po-package directory.
2. Execute the following unix commands:
   uncompress po-pack.tar.Z
   tar xvf po-pack.tar
3. Execute "make" in the PO-PACKAGE-3.0 directory.

The use of SPIN with the Partial-Order Package remains very similar to the use of SPIN alone. The validation of a PROMELA program can be performed as follows.

spin -a filename (build the validator)

cc -c pan.c (compile the validator)

cc -c po.c (compile the partial-order functions)

cc -o pan pan.o po.o (link the object code with the partial-order functions object code)

pan (run the validator)

By default, the validation reports deadlocks, unreachable code and assertion violations. In order to detect deadlocks only, use the DEADLOCK option as follows (the reduced state space can be further reduced when searching for deadlocks only):

```
cc -DDEADLOCK -c pan.c
cc -DDEADLOCK -c po.c
cc -o pan pan.o po.o
```

Version 3.0 of the ULg Partial-Order Package is compatible with the Promela language as defined in Version 1.6.5 of SPIN. A few minor changes to the Promela language have been made in order to make systems described in Promela compatible with the assumptions under which the partial-order algorithms have been developed, and in order to clarify the semantics of Promela when needed. For instance, dynamic process creation has been forbidden, and the use of the "atomic" Promela expression has been defined more strictly. Promela has also been extended with two predicates *Empty* and *Full* on FIFO channels [GP93], for which optimizations are implemented in the Package. See the README file for the complete and detailed list of these assumptions.

The supertrace option -DBITSTATE of SPIN is compatible with the Partial-Order Package. They can be used simultaneously by compiling the pan.c file as follows:

```
cc -DBITSTATE -c pan.c
cc -DBITSTATE -c po.c
cc -o pan pan.o po.o
```

3 State-Space Caching

The ULg Partial-Order Package also includes an implementation of the *state-space caching* technique. *State-space caching* [Hol85, JJ91] is a memory management technique for storing the states encountered during a depth-first search that consists in storing all the states of the current explored path (i.e., those in the current depth-first search "stack") plus as many other states as possible given the remaining amount of available memory. It thus creates a restricted *cache* of selected system states that have already been visited. Initially, all states encountered are stored into the cache. When the cache fills up, old states that are not in the stack are removed from the cache to accommodate new ones. This method never tries to store more states than possible in the cache. Thus, if the size of the cache is greater than the maximal size of the stack during the exploration, the search is not truncated, and eventually terminates.

In [GHP92, God94], it is shown that state-space caching and partial-order methods combine very well: the memory requirements needed to validate large protocol models can be strongly decreased (e.g., more than 100 times) without seriously increasing the time requirements.

When the state space being explored by SPIN is too large to be stored in memory, even with the Partial-Order Package, try the state-space caching option. A new option -nC specifying the maximum number C of states that can be stored in memory (i.e., the size of the cache) has been added to the executable file pan:

pan -nC -wN

where C is the maximum number of states that can be stored in main memory (C is typically set to slightly less than M/S where M is the amount of RAM available on your computer and S is the state vector size, which is indicated in the diagnosis produced by SPIN; C must be small enough to avoid paging) and where N is about $log_2(2C)$ (in order to keep the hash table at 50% not full).

4 More

The ULg Partial-Order Package is distributed free of charge for research and educational use only. No guarantee is expressed or implied by the distribution of this code.

This software was written by Patrice Godefroid, Didier Pirottin and Pierre Wolper, Computer Science Department, University of Liège, with the collaboration of Gerard J. Holzmann, AT&T Bell Laboratories. The main reference describing the algorithms implemented in the Partial-Order Package and presenting results of experimentations is [God94]. Please send your comments, questions, bug-reports and results of experiments to:

po-package@montefiore.ulg.ac.be

References

[GHP92] P. Godefroid, G. J. Holzmann, and D. Pirottin. State space caching revisited. In *Proc. 4th Workshop on Computer Aided Verification*, volume 663 of *Lecture Notes in Computer Science*, pages 178–191, Montreal, June 1992. Springer-Verlag.

[God90] P. Godefroid. Using partial orders to improve automatic verification methods. In *Proc. 2nd Workshop on Computer Aided Verification*, volume 531 of *Lecture Notes in Computer Science*, pages 176–185, Rutgers, June 1990. Springer-Verlag. Extended version in ACM/AMS DIMACS Series, volume 3, pages 321–340, 1991.

[God94] Patrice Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems – An Approach to the State-Explosion Problem.* PhD thesis, University of Liège, Computer Science Department, November 1994. (Also available by anonymous ftp from ftp.montefiore.ulg.ac.be in the pub/po-package directory, file thesis.ps.Z).

[GP93] P. Godefroid and D. Pirottin. Refining dependencies improves partial-order verification methods. In *Proc. 5th Conference on Computer Aided Verification*, volume 697 of *Lecture Notes in Computer Science*, pages 438–449, Elounda, June 1993. Springer-Verlag.

[GW93] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. *Formal Methods in System Design*, 2(2):149–164, April 1993.

[HGP92] G. J. Holzmann, P. Godefroid, and D. Pirottin. Coverage preserving reduction strategies for reachability analysis. In *Proc. 12th IFIP WG 6.1 International Symposium on Protocol Specification, Testing, and Verification*, pages 349–363, Lake Buena Vista, Florida, June 1992. North-Holland.

[Hol85] G. J. Holzmann. Tracing protocols. *AT&T Technical Journal*, 64(12):2413–2434, 1985.

[Hol91] G. J. Holzmann. *Design and Validation of Computer Protocols.* Prentice Hall, 1991.

[JJ91] C. Jard and Th. Jeron. Bounded-memory algorithms for verification on-the-fly. In *Proc. 3rd Workshop on Computer Aided Verification*, volume 575 of *Lecture Notes in Computer Science*, Aalborg, July 1991. Springer-Verlag.

[Val90] A. Valmari. A stubborn attack on state explosion. In *Proc. 2nd Workshop on Computer Aided Verification*, volume 531 of *Lecture Notes in Computer Science*, pages 156–165, Rutgers, June 1990. Springer-Verlag.