Brief Announcement:

Automatic Discovery of Mutual Exclusion Algorithms*

Yoah Bar-David †　　Gadi Taubenfeld ‡

We present a methodology for automatic discovery of synchronization algorithms. We built a tool and used it to automatically discover hundreds of new algorithms for the well-known problem of mutual exclusion. The methodology is rather simple and the fact that it is computationally feasible is surprising. Our brute force approach may require (even for very short algorithms) the mechanical verification of hundreds of millions of incorrect algorithms before a correct algorithm is found. Although many new interesting algorithms have been found, the main contribution of this work is in demonstrating that the approach suggested is feasible.

The architecture of the tool is as follows: Via a user-interface, the user can set the problem parameters: (1) number of processes (2) number of lines of code (3) number, size and type of variables (4) type of conditions: simple or complex. Complex conditions are composed of two simple conditions (terms) related by *and*, *or* or *xor*. The parameters are sent to the algorithm generator, which generates all the possible algorithms according to the given parameters. Each algorithm (which passes the optimization checks) is sent to verification. If an algorithm is verified as correct, it is sent back to the user-interface and displayed. Verification results are also returned to the algorithm generator for use in optimizations. A tool has been implemented in Java and C++, and has around 10,000 lines of code.

One of the challenges in building the tool was to be able to process enough algorithms in a reasonable time, so that interesting results can be found. To achieve this, many optimizations were implemented. As an example, consider algorithms for two processes with 3 shared bits, complex conditions, 4 entry and 1 exit commands. For this setting there are over 10^{21} possible algorithms in our (high-level) generation language. Using all optimizations, less than $3 \cdot 10^7$ algorithms were generated and tested, of which 105 correct algorithms were found, one of which is the famous Peterson's algorithm [1].

The tool was run to find correct mutual exclusion algorithms for two processes using 2, 3, 4, 5 or 6 shared bits. A methodical search was conducted to find the shortest solutions: the number of commands was incrementally increased until a solution was found. The lengths of the shortest solutions are summarized in the table below. The first column indicates the number of shared bits used; the second and third columns indicates the number of entry and exit commands. The fourth column indicates whether the use of complex conditions is allowed. The number of tested algorithms, and the number of correct algorithms found are displayed in the fifth and sixth columns. As a result of optimizations, not all generated algorithms were actually tested. The number of tested algorithms counts only those that were tested by the verifier. All correct algorithms that have been found are new (and shorter than previously known solutions), except for Peterson's algorithm [1]. All the tests were performed on a Pentium 4/1.6 GHz PC.

User-defined parameters				Result		
# of shared bits	# of entry	# of exit	complex conditions	tested algorithms	correct alg.	appx. running hours
2	6	1	yes	7,196,536,269	0	216
2	7	1		846,712,059	66	39
3	4	1	yes	25,221,389	105	0.4
3	6	1		1,838,128,995	10	47
4	4	1	yes	129,542,873	480	1
4	5	1		129,190,403	56	1
4	6	1		[1]900,000,000	80	12
5	5	1		[1]22,000,000	106	0.4
6	5	1		[1]70,000,000	96	1

The full version includes: new algorithms, more tests and results, details about the generator and the verifier (model-checker), description of optimizations and techniques which dramatically improve the performance.

References

[1] G.L. Peterson. Myths about the mutual exclusion problem. *Information Processing Letters*, 12(3):115-116, 1981.

[1]This run was stopped after few solutions were found. Not all possible algorithms were tested.

*A full version of this paper is available at `www.bardavid.com`, and `www.faculty.idc.ac.il/gadi/`. Part of the work was done while the authors were with the Open University of Israel.

†`yoah@bardavid.com`

‡The school of computer science, the Interdisciplinary Center, P.O.Box 167, Herzliya 46150, Israel. `tgadi@idc.ac.il`

PODC'03, July 13–16, 2003, Boston, Massachusetts, USA.