ConcurrencySuite: Teaching concurrency and nondeterminism with Spin

Mordechai (Moti) Ben-Ari
Department of Science Teaching
Weizmann Institute of Science
Rehovot 76100 Israel
moti.ben-ari@weizmann.ac.il

Abstract

The Spin model checker is an excellent system for teaching concepts such as concurrency, verification and nondeterminism. I will show pedagogical tools that I have built based upon Spin: the jSpin environment for developing and verifying concurrent programs; SpinSpider, a tool for generating graphical representations of state diagrams of concurrent programs; VN, a tool for ”experiencing” the nondeterminism of finite automata.

1. Introduction

Concurrency and nondeterminism are two of the most challenging topics in computer science. Concurrency is difficult because you cannot test a concurrent program so formal verification techniques are indispensable [2]. Nondeterminism is difficult because it is unnatural to students who have studied deterministic algorithms and programming [1].

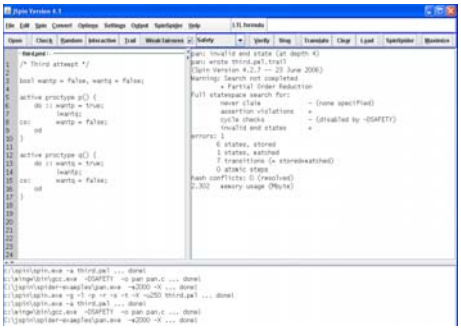
The classic way of teaching concurrency is with a concurrency simulator [4]. While writing the latest edition of my textbook [2], I found that it is advantageous to use the *Spin* model checker [5] as the primary teaching tool (although the textbook demonstrates Spin only as one possible system for implementating concurrency). To facilitate learning this important software tool, I have recently written an introductory textbook [3]. I have also developed a suite of Spin-based pedagogical software tools.

2. Spin

Spin was developed by Gerard J. Holzmann for verifying communications protocols; it earned him the ACM Software System Award for 2001. I find Spin attractive for teaching: although it is a professional tool that is widely used in industry, it is simple enough for undergraduate students to use. Concurrent programs are written in a simple C-like language called *Promela*, and simulation and verification are carried out by Spin. Spin is a command-line program that needs no installation and is portable.

3. jSpin

Students expect to work with an integrated development environment not a command-line tool, so I developed *jSpin* as an IDE for Spin. jSpin filters the scenario data that is generated by Spin so that it can be flexibly displayed by the student:

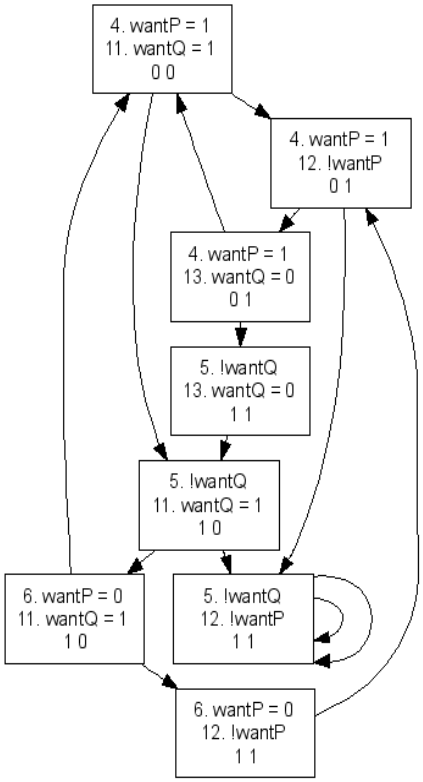


4. SpinSpider

The semantics of concurrent programs are defined in terms of automata and can be displayed graphically as state transition diagrams. *SpinSpider* is a tool that uses data available from Spin to construct the state transition diagram, which is then written in the *dot* language and the

`dot` program of GraphViz is called to layout the graph.

Here is the diagram generated *automatically* by SpinSpider for the "third attempt" to solve the critical section problem [2, Section 3.7]; the potential for deadlock is instantly apparent in the state near the bottom of the diagram, and the reason for the deadlock is easy to explain using the visualization:



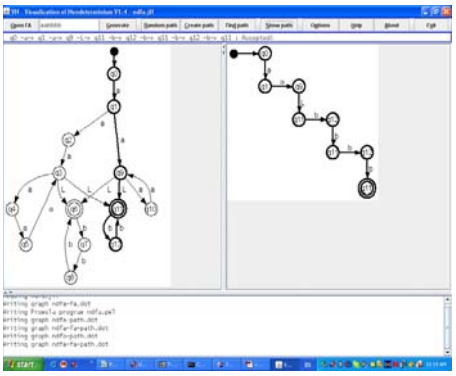
5. VN: Visualization of Nondeterminism

Nondeterminism appears in several contexts such as nondeterministic finite automata (NDFA) and nondeterministic algorithms. Students find it difficult to understand the definition of acceptance of a string by an NDFA-the string is accepted iff there *exists* an accepting computation; this mathematical definition is at odds with the intuition of a machine that searches for a solution, for example, by backtracking.

The specification of VN was worked out in collaboration with Michal Armoni. VN is a software tool that leverages the nondeterministic execution of a Promela program by Spin and the graph layout capabilities of GraphViz to enable the student "experience" the nondeterminism of an NDFA. VN can run in three modes: (1) random resolution of nondeterminism to emphasize that the result of a single computation of an NDFA is arbitrary; (2) interactive resolution of nondeterminism which demonstrates that a "magic coin" can guide the NDFA to an accepting computation; (3) verification mode to find an accepting computation if one exists, demonstrating that an accepting computation can be found by an exhaustive deterministic search. VN can also find all accepting computations of all inputs of a given length, and for deterministic automata, it can compute the partition of the input strings accepted at each node.

The input to VN is the description of an NDFA created by JFLAP and a string entered by the user. VN generates a Promela program which is then executed by Spin in one of the above modes. A *dot* file is generated and the `dot` program of GraphViz is called to layout the graph.

Here is the visualization of an accepting computation of an NDFA, where the path within the NDFA is shown in bold in the left pane, while the full path (including repetitions) is shown in the right pane:



6. Resources

The software tools I developed can be downloaded from:

`http://stwww.weizmann.ac.il/g-cs/benari/home/software.html`

or from:

`http://sourceforge.net/projects/pcdp`

The tools are written in Java and are available under the GNU GPL. The other tools that are needed (Spin, Graphviz, a C compiler, JFLAP) can also be freely downloaded.

References

[1] M. Armoni and J. Gal-Ezer. Introducing nondeterminism. *J. of Computers in Mathematics and Science Teaching*, 25(4):325--359, 2006.

[2] M. Ben-Ari. *Principles of Concurrent and Distributed Programming (Second Edition)*. Addison-Wesley, Harlow, UK, 2006.

[3] M. Ben-Ari. *Principles of Spin*. Springer, London, 2008.

[4] B. Bynum and T. Camp. After you, Alfonse: A mutual exclusion toolkit. *SIGCSE Bulletin*, 28(1):170--174, 1996.

[5] G. J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, Boston, MA, 2004.