TopSpin User Manual

Alastair F. Donaldson

Department of Computing Science
University of Glasgow
Glasgow, Scotland.
{ally}@dcs.gla.ac.uk

Abstract. This document is a short user manual for the TopSpin symmetry reduction package for the Spin model checker.

1 Introduction

TopSpin [3] is an automatic symmetry reduction package for the Spin model checker [8]. TopSpin can potentially aid in the verification of *safety* properties of concurrent distributed systems modelled in Promela, if the system exhibits replicated structure. Safety properties can be specified using assertions or monitor processes. The tool does not yet support LTL model checking with never claims. This document outlines the process of downloading, configuring and installing TopSpin. For details of symmetry reduction techniques see a recent survey [9]. We are keen for questions, queries and bug reports – please use the email address above.

2 Installing TopSpin

Section 2.1 provides details of how to get hold of prerequisite packages which TopSpin requires. Downloading of TopSpin is described in Section 2.2, while Section 2.3 explains how to configure TopSpin.

2.1 Prerequisites

TopSpin is implemented in Java 5, and uses an additional PERL script, and thus requires a recent Java distribution and a PERL implementation to run. Spin also requires the **gcc** compiler to be installed. TopSpin runs under Linux or Windows.

The automatic symmetry detection component of TopSpin uses the *saucy* graph automorphism program [1], and the Gap computational algebra system [7] is used for symmetry detection, classification of group structure before search and (if the *segment* strategy is used) representative computation during search. This section provides details on how to get hold of these required packages.

Java 5 This can be downloaded free from `www.java.sun.com`.

PERL This comes as standard with most Linux distributions, and can be downloaded from `www.perl.com`.

gcc The GNU C compiler also comes as standard with Linux, and can be downloaded from `gcc.gnu.org`.

Make sure `java`, `perl` and `gcc` are on your path before you try to use TopSpin.

Gap Windows and Linux distributions of Gap can be downloaded from `www.gap-system.org`, and detailed installation instructions are available from this location.

saucy Source code for the graph automorphism program [1] can be downloaded from `vlsicad.eecs.umich.edu/BK/SAUCY`, and is straightforward to compile using `gcc`.

One of the files in the TopSpin distribution, `PreparePan.pl`, is Linux system dependent as it uses various Linux commands such as `cp` and `ls`. If you have `cygwin` installed under Windows (see `www.cygwin.com` for details), this is fine. Otherwise please email the author and a Windows version of this file will be prepared.

2.2 Downloading TopSpin

Go to `www.dcs.gla.ac.uk/people/personal/ally/topspin` and download a `.zip` file containing the most recent TopSpin distribution. This archive contains the following:

- `TopSPIN.jar` – the executable `.jar` file containing TopSpin.
- `config` – a plain text file containing configuration details, which must be edited as described in Section 2.3.
- `Documentation` – a folder containing this document and some related research papers.
- `lib` – a folder containing `junit.jar`, the JUnit test framework used in the TopSpin implementation (see textttwww.junit.org).
- `TempFiles` – an empty folder which TopSpin will use for temporary storage.
- `Common` – a folder containing various Gap and C source code files used by TopSpin.

Extract this archive using something like WinZip (`www.winzip.com`) to an appropriate location (e.g. `C:/Program Files/TopSPIN/`). We refer to this location below as `topspinfolder`.

2.3 Configuring TopSpin

Creating a Gap workspace So that Gap can be started by TopSpin in a state where the Gap code in the `topspinfolder/Common` directory is already loaded, you must create a *workspace*. Navigate to the `Common` directory, start Gap and type:

```
Read("WorkspaceGenerator.gap");

SaveWorkspace("gapworkspace");

quit;
```

The middle command should output `true`.

Editing the `config` file A text file called `config` is provided with the distribution. Edit this file so that `gap` and `saucy` refer to the absolute locations of your Gap and *saucy* executables, and `tempfiles` and `common` refer to the folders `TempFiles` and `Common` within the TopSpin distribution.

To run TopSpin you need to copy `config` to your current directory. This is easy to forget!

3 Running TopSpin

To run TopSpin, navigate to the folder in which you would like your executable verifier to be generated (as you would when using Spin as standard). Copy the `config` file to this directory. Then type

```
java -jar topspinfolder/TopSPIN.jar detect_opt reduce_opt model
```

Here `model` is the name of the Promela specification you wish to check, `detect_opt` is a symmetry detection option (see Section 3.1), and `reduce_opt` is a symmetry reduction option (see Section 3.2). If neither option is specified, the Promela file is just type-checked. If only a symmetry detection option is given, the detected symmetries are displayed but no verifier is generated.

If both symmetry reduction options are specified then an executable verifier, `sympan.c`, is generated. This can be compiled and executed using the same method that would be used with the `pan.c` file generated by Spin, except that the file `group.o` must appear in the compilation command. For example, one could compile `sympan.c` as follows:

```
gcc -w -o sympan -D_POSIX_SOURCE -MEMLIM=256 -DSAFETY -DNOCLAIM
-DNOFAIR sympan.c group.o
```

3.1 Symmetry Detection Options

TopSpin provides two techniques for automatic symmetry detection – *static channel diagram* or *identifier adjacency diagram* analysis. Additionally, TopSpin allows generators for a symmetry group to be specified manually. The following options specify which symmetry detection method should be used, and should be given as the first parameter to TopSpin:

-scd With this option, symmetry detection is via static channel diagram analysis. Details of this technique in general can be found in [6], with Spin-specific details in [2]. Static channel diagram analysis works well when fixed communication links are passed to processes as parameters on instantiation.

-iad *Identifier adjacency diagrams* are a (currently unpublished) alternative to static channel diagrams, and are suitable for symmetry detection when the communication structure of a model is encoded in the Promela specification via a lookup macro. For example....... Occurrences of literal *pid* values or channel names are used to build a directed graph, from which model automorphisms are derived using the computational group theoretic approach of [2].

-manual <filename>
If a specification is know to exhibit component symmetry, but this symmetry is not detected using the above approaches, it is possible to manually specify generators for a symmetry group. The file specified by <**filename**> should contain one line per generator, and each generator should be a product of permutations of global channel names and literal *pid* values.

3.2 Symmetry Reduction Options

-enumerate The *enumerate* symmetry reduction option computes orbit representatives by exhaustively enumerating images of a state under a group G. Thus the time complexity of computing a representative is proportional to $|G|$, which may be as large as $|n!|$ where n is the number of system components. Enumeration is memory optimal and is suitable when G is small, or for purposes of comparison with other strategies.

-fast With the *fast* option, a symmetry reduction strategy will be chosen for the group G based on its structure, as described in [4]. If G is small, the *enumerate* option may be used. If G is unclassifiable, the approximate *hillclimbing* option (below) will be used. Otherwise a strategy based on minimising sets for G will be applied, resulting in fast representative computation. Memory optimality is not guaranteed if components in the specification hold references to one another, but in practice a close-to-optimal factor of reduction is usually achieved.

-hillclimbing This option means that representatives will be computed using an approximate strategy described in [4] based on hillclimbing local search.

-segment This is an experimental option described in [5] which combines features of the *fast* and *enumerate* options to provide memory optimal verification using an approach which, for groups with identifiable structure, is less naïve than pure enumeration. To use this option, you must copy your executable `sympan` file into `topspinfolder/Common`, navigate there, launch GAP, and type

```
Read("Verify.gap");

Verify("sympan",[]);
```

Currently it is not possible to pass command line arguments to `sympan` this way, so you must make modifications directly to `sympan.c` before compiling it. The most common command line argument is one to change the maximum search depth. To do this via `sympan.c`, open the file and find the line where the variable `maxdepth` is initialised (to 10000 by default). Change the initial value to a more appropriate one.

4 Troubleshooting

This section will be expanded to include a list of common problems with the use of TopSPIN. In the mean time, please email the author with any queries.

References

1. P. T. Darga, M. H. Liffiton, K. A. Sakallah, and I. L. Markov. Exploiting structure in symmetry detection for CNF. In *DAC'04*, pages 530–534. ACM Press, 2004.
2. A.F. Donaldson and A. Miller. Automatic symmetry detection for model checking using computational group theory. In *FM'05*, LNCS 3582, pages 418–496. Springer, 2005.
3. A.F. Donaldson and A. Miller. A computational group theoretic symmetry reduction package for the SPIN model checker. In *AMAST'06*, LNCS 4019, pages 374–380. Springer, 2006.
4. A.F. Donaldson and A. Miller. Exact and approximate strategies for symmetry reduction in model checking. In *FM'06*, to appear in LNCS. Springer, 2006.
5. A.F. Donaldson and A. Miller. Extending symmetry reduction techniques to a realistic model of computation. Submitted for publication, June 2006.
6. A.F. Donaldson, A. Miller, and M. Calder. Finding symmetry in models of concurrent systems by static channel diagram analysis. In *AVoCS'04*, ENTCS 128(6), pages 161–177. Elsevier Science Publishers B.V, 2005.
7. The Gap Group. *GAP–Groups Algorithms and Programming, Version 4.2*. Aachen, St. Andrews, 1999. http://www-gap.dcs.st-and.ac.uk/~gap.
8. G. J. Holzmann. *The SPIN model checker: primer and reference manual*. Addison Wesley, 2003.
9. A. Miller, A. Donaldson and M. Calder. Symmetry in temporal logic model checking. *Computing Surveys*, To appear, 2006.