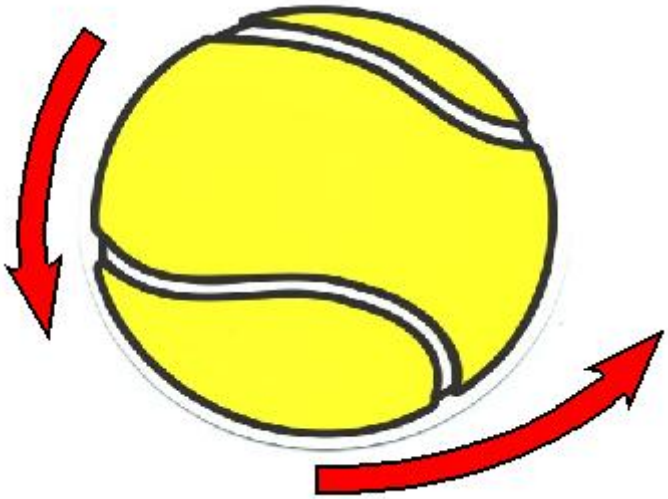
TopSpin

Version 2.1



Automatic symmetry reduction
for the SPIN model checker

User Manual

Alastair F. Donaldson

Before You Begin

What this manual covers

This user manual provides details of how to download, install and use TopSPIN, an automatic symmetry reduction tool for the SPIN model checker. TopSPIN can potentially aid in the verification of *safety* properties of concurrent systems specified in Promela.

- **Downloading and Installing** Chapter 1 provides details of other packages on which TopSPIN depends, including where these packages can be found, and explains how to download and install TopSPIN.
- **Worked Example** Chapter 2 provides a worked example showing how to run TopSPIN to obtain symmetry reduction for an example specification.
- **Overview of Options** An overview of TopSPIN options is presented in Chapter 3.
- **Compiling From Source** For users who wish to experiment further with TopSPIN, Chapter 4 explains how to obtain, compile and test the TopSPIN source code.
- **Troubleshooting and Bug Reporting** Chapter 5 presents solutions to common problems associated with the installation and operation of TopSPIN, and provides details of how bugs should be reported.

What this manual does not cover

- **Theory** Users interested in the theory on which TopSPIN is based should refer to relevant papers and a Ph.D. thesis, which are available from the tool web page (see below).
- **Use of SPIN** This manual assumes that the reader is familiar with the SPIN tool and the Promela language. Details of and documentation for the SPIN tool are available from the SPIN web page.[1]

Online resources

- **TopSPIN web page** `http://www.allydonaldson.co.uk/topspin/`
- **SourceForge** `https://www.sourceforge.net/projects/symmetryglasgow/`

1. `http://www.spinroot.com/`

1 Downloading and Installing

1.1 Prerequisites

TopSPIN is written in Java and GAP, interfaces with the GAP and SPIN packages, and produces C code which must then be compiled. Figure 1.1 summarises the packages which must be installed before TopSPIN can be used, and provides a URL for each package. For each package, the version used during the development of TopSPIN is specified. Use of these or newer versions is recommended.

Before going further, make sure each of the packages of Figure 1.1 is installed on your system

Important Make sure that the SPIN tool is installed in such a way that it can be invoked by name from a command prompt, i.e. so that typing `spin` will launch SPIN. To verify that SPIN is set up appropriately, type `spin` in a fresh command prompt. You should see something like:

```
C:\>spin
Spin Version 5.1.6 -- 9 May 2008
reading input from stdin:
```

If you have renamed the SPIN executable from `spin` to something like `spin-linux` or `spin516` then you will get errors when you run TopSPIN.

1.2 Downloading

To download TopSPIN version 2.1, go to the TopSPIN web page:

`http://www.allydonaldson.co.uk/topspin/`

and download the following file:

`TopSPIN_2.1.tgz`

Use the `tar` utility, the *WinRAR* tool,[1] or another suitable program to extract this archive to an appropriate location, e.g. `C:\Program Files\TopSPIN_2.1` under Windows, or `/usr/local/TopSPIN_2.1` under Linux. This location is referred to as the TopSPIN *root directory*.

The archive should contain the following:

- **`TopSPIN.jar`** Executable jar for the TopSPIN program
- **`documentation`** Folder containing this document, some related research papers and a Ph.D. thesis
- **`examples`** Folder containing example Promela specifications for use with TopSPIN
- **`Common`** Folder containing various GAP and C files files used by TopSPIN
- **`saucy`** Folder containing source code for the *saucy* program, which must be compiled (as described in §1.3.1) before TopSPIN can be used.

1. `http://www.rarlab.com/`

Package	URL	Version
Java runtime environment	`http://java.sun.com/`	1.5.0_06
GAP system	`http://gap-system.org/`	4.4.6
SPIN model checker	`http://www.spinroot.com/`	5.1.6
GNU C Compiler (gcc)	`http://gcc.gnu.org/`	3.4.4

Figure 1.1: TopSPIN prerequisites.

1.3 Installing

1.3.1 Compiling *saucy*

TopSPIN uses a prototype extension of the *saucy* program, which is used to compute symmetries of directed graphs. A version of *saucy* with this capability will eventually be available from the *saucy* website.[2] For the time being, a source distribution of *saucy* with the required extended functionality is provided with TopSPIN.[3]

Before using TopSPIN, you need to compile *saucy* on your platform. To do this, navigate to the `saucy` folder, which is inside the TopSPIN root directory, and type `make`:

```
C:\Program Files\TopSPIN_2.1>cd saucy C:\Program
Files\TopSPIN_2.1\saucy>make gcc -ansi -pedantic -Wall -O3   -c -o
main.o main.c gcc -ansi -pedantic -Wall -O3   -c -o saucy.o saucy.c
gcc -ansi -pedantic -Wall -O3   -c -o saucyio.o saucyio.c gcc  -o
saucy main.o saucy.o saucyio.o
```

1.3.2 Creating a GAP workspace

In order to start GAP efficiently, TopSPIN requires a GAP *workspace* to be set up. Essentially, a workspace is an image of a GAP session with a selection of libraries and files already loaded and ready to be executed. For TopSPIN, the workspace consists of the GAP components of TopSPIN which have been developed for automatic symmetry reduction. These are in the `Common` folder, inside the TopSPIN root directory.

Navigate into the `Common` folder, and start GAP:

```
C:\Program Files\TopSPIN_2.1>cd Common

C:\Program Files\TopSPIN_2.1\Common>gap

C:\Program Files\TopSPIN_2.1\Common>rem sample batch file for GAP

C:\Program Files\TopSPIN_2.1\Common>C:\GAP4R4\bin\gapw95.exe -m 14m
-l C:\GAP4R4\
```

```
     Information at:  http://www.gap-system.org
     Try '?help' for help. See also  '?copyright' and  '?authors'

   Loading the library. Please be patient, this may take a while.
GAP4, Version: 4.4.6 of 02-Sep-2005, i686-pc-cygwin-gcc
Components:  ...
Packages:    ...
gap>
```

Now create a workspace as follows:

```
gap> Read("WorkspaceGenerator.gap");
gap> SaveWorkspace("gapworkspace");
```

---

2. `http://vlsicad.eecs.umich.edu/BK/SAUCY/`
3. Permission for including the *saucy* distribution with TopSPIN has been granted by Paul Darga, lead developer of *saucy*.

```
true
gap> quit;
```

Ensure that these commands are typed *exactly* as shown, ensuring that the semi-colon is included after the `quit` command. When you exit GAP, you should find a file called `gapworksapce` in the `Common` directory.

1.4 Executing the TopSPIN jar file

It should now be possible to run the TopSPIN jar file on no input to display a list of TopSPIN options:

```
C:\>java -jar "C:\Program Files\TopSPIN_2.1\TopSPIN_2.1.jar"

Usage: topspin [check,detect] <filename>

Configuration file options:

 OPTION           PURPOSE           DEFAULT
 ------           -------           -------
  ...               ...               ...
```

The options displayed by the tool are described fully in Chapter 3. For now, successfully displaying these options confirms that the Java runtime environment is correctly installed on your system, and that the TopSPIN jar file you have obtained is in working order. If TopSPIN does *not* correctly display its options, proceed to Chapter 5 to try to deduce what is wrong, and to file a bug report if necessary.

In Chapter 2, instructions for running TopSPIN to perform symmetry reduction on a Promela specification are given.

2 Worked Example

In this chapter, the basic workings of TopSPIN are illustrated via a worked example.

2.1 Loadbalancer specification

The `examples` folder in the TopSPIN root directory contains a file, `loadbalancer.p`. This is a Promela specification for a *loadbalancer*, which forwards requests from a pool of clients to a pool of servers in a fair manner.

Components in the loadbalancer are a set of 2 *server* and 4 *client* processes with associated communication channels, and a *loadbalancer* process with a dedicated input channel. The *load* of a server is the number of messages queued on its input channel. Client processes send requests to the loadbalancer, and if some server's channel is not full, the loadbalancer forwards a request nondeterministically to one of the least-loaded server queues. Each request contains a reference to the input channel of its associated client process, and the server designated by the loadbalancer uses this channel to service the request.

2.2 Applying SPIN to the loadbalancer specification

Before applying TopSPIN to this example, it is worth checking that SPIN is in good working order by model checking the example without symmetry reduction.

Navigate to the `examples` directory, and run the following commands:

```
C:\Program Files\TopSPIN_2.1\examples>spin -a loadbalancer.p
C:\Program Files\TopSPIN_2.1\examples>gcc -o pan -O2 -DSAFETY pan.c
C:\Program Files\TopSPIN_2.1\examples>pan -m100000
```

SPIN should successfully verify that the model associated with the specification is deadlock-free, producing output similar to:

```
(Spin Version 5.1.6 -- 9 May 2008)
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +

State-vector 108 byte, depth reached 73413, errors: 0
   170903 states, stored
   413074 states, matched
   583977 transitions (= stored+matched)
        6 atomic steps
hash conflicts:     48755 (resolved)

   24.974       memory usage (Mbyte)

unreached in proctype client
        line 20, state 6, "-end-"
        (1 of 6 states)
unreached in proctype loadBalancer
        line 34, state 13, "-end-"
        (1 of 13 states)
unreached in proctype server
```

```
        line 43, state 7, "-end-"
        (1 of 7 states)
unreached in proctype :init:
        (0 of 9 states)

pan: elapsed time 0.818 seconds pan: rate 208927.87 states/second
```

2.3 Setting up a TopSPIN configuration file

You are almost ready to use TopSPIN for symmetry reduction! When you invoke TopSPIN on a Promela specification, the tool requires that a file called `config.txt` is available in the current directory. This file tells TopSPIN where to find the GAP and *saucy* programs, the location of some common source code, and the values of various runtime options. The ile consists of a series of lines, each of which has the form:

attribute=*value*

Three attributes are required in every configuration file:

1. **`gap`** – the absolute path required to launch GAP. The value for the `gap` attribute must be such that typing this value at the command prompt is all that is required to launch the GAP program.
2. **`saucy`** – the absolute path for the *saucy* executable. Again, the value for the `saucy` attribute must be exactly what you type to run *saucy* from the command line.
3. **`common`** – the absolute path to the `Common` directory in the TopSPIN root directory, followed by a (back- or forward-, depending on your operating system) slash.

Under Windows, `config.txt` might look like this:

```
gap=C:\gap4r4\bin\gap.bat
saucy=C:\Program Files\TopSPIN_2.1\saucy\saucy.exe
common=C:\Program Files\TopSPIN_2.1\Common\
```

whereas a possible Linux version of `config.txt` could be:

```
gap=/users/grad/ally/Scripts/gap
saucy=/users/grad/ally/TopSPIN_2.1/saucy/saucy
common=/users/grad/ally/TopSPIN_2.1/Common/
```

The remainder of `config.txt` is used to specify TopSPIN options for a particular specification. One of these options is discussed in §2.5, and a complete overview of options is given in Chapter 3.

A configuration file must always be present in the directory from where you invoke TopSPIN. You will probably use different configuration files for different specifications, depending on the nature of the symmetry associated with these specifications. However, since the `gap`, `saucy` and `common` attributes are likely to be the same in all configuration files, it makes sense to keep a "skeleton" configuration file containing just these options, which you can then copy and extend for a given Promela specification.

2.4 Symmetry reduction with the *fast* strategy

When you run TopSPIN, you can specify a symmetry reduction *strategy* for the tool to use. The choice of strategy influences the speed of symmetry reduction and the factor of reduction obtained though the use of symmetry. Some strategies provide the maximum possible state-space reduction due to symmetry, at the expense of a slow state-space search. Other strategies are more lightweight, providing potentially sub-optimal reduction, but executing more quickly.

By default, TopSPIN uses the *fast* strategy. In a nutshell, when using this strategy TopSPIN attempts to work out an efficient symmetry reduction algorithm based on the type of symmetry associated with the input specification. The symmetry reduction algorithm used is *approximate* in the sense that it may result in exploration of more than one state per symmetric equivalence class.

2.4.1 Running TopSPIN

Copy the basic configuration file created in §2.3 into the `examples` directory, and run TopSPIN on `loadbalancer.p` as follows:[1]

```
C:\Program Files\TopSPIN_2.1\examples>java -jar
"C:\Program Files\TopSPIN_2.1\TopSPIN.jar" loadbalancer.p

---------------------------------------
TopSPIN version 2.1
---------------------------------------
Configuration settings:
    Symmetry detection method: static channel diagram analysis
    Using 0 random conjugates
    Timeout for finding largest valid subgroup: 0 seconds
    Reduction strategy: FAST
    Using transpositions to represent permutations: true
    Using stabiliser chain for enumeration: true
    Using vectorisation: false
---------------------------------------

Typechecking input specification...

Specification is well typed!

Launching saucy via the following command: C:\Program Files\TopSPIN_2.1\saucy\saucy.exe -d
"C:\Program Files\TopSPIN_2.1\Common\graph.saucy"

Starting GAP with command: C:\gap4r4\bin\gap.bat -L
"C:\Program Files\TopSPIN_2.1\Common\gapworkspace" -q

The group:
   G = <(3 2)(clients0 clients1),(5 4)(clients2 clients3),(servers1 servers0)(7 6),
        (3 4)(clients2 clients1)>
is a valid group for symmetry reduction.

Generating symmetry reduction algorithms

The symmetry group has size 48
Completed generation of sympan verifier which includes algorithms for symmetry reduction!

To generate an executable verifier use the following command:
   gcc -o sympan sympan.c group.c
together with SPIN compile-time directives for your specification.

Execute the verifier using the following command:
   sympan.exe
together with SPIN run-time options for your specification.
```

If TopSPIN does *not* produce output like this, then proceed to Chapter 5 to try to deduce what is wrong, and to file a bug report if necessary.

The first part of the TopSPIN output specifies which version of the tool is being used, and how the various tool options have been configured. These options have all been set to default values. For now, simply note that the *fast* strategy is being used.

The tool then typechecks the input specification, reporting that the specification is well-typed.

After typechecking, the *saucy* and GAP programs are launched, to perform automatic symmetry detection. Note that the paths to these tools have been taken from the configuration file. The *saucy* tool is passed the `-d` option to indicate that the graph it will operate on is *directed*. A temporary file, `graph.saucy`, is also passed to the program: this is a graph generated by TopSPIN from which symmetries of `loadbalancer.p` are derived. The GAP package is launched with the `-L` option to indicate that a workspace (§1.3.2) should be loaded. The path to this workspace – `gapworkspace` inside the `Common` directory – is also passed to GAP. The `-q` operation launches GAP in *quiet* mode, which improves the efficiency of communication between the GAP and Java components of TopSPIN.

---

1. For reasons of space, some lines in the TopSPIN output have been wrapped. Also note that the output on your system will vary slightly according to the paths you have specified in `config.txt`.

TopSPIN successfully computes generators for a group of symmetries associated with the loadbalancer specification. From the generators of this group, it can be seen that there is full symmetry between the *client* processes, as well as symmetry between the *server* processes. TopSPIN then goes on to generate symmetry reduction algorithms for this group, reporting that the number of symmetries is 48 (4! symmetries between the clients, and 2 symmetries between the servers, leads to $24 \times 2 = 48$ symmetries overall).

The end of the TopSPIN output details how the C files generated by SPIN and TopSPIN should be compiled and executed.

2.4.2 Compiling and executing the `sympan` verifier

The `examples` directory should now contain a number of files, including `sympan.c` and `group.c`. The `sympan.c` file is a modified version of the `pan.c` file produced by SPIN, which includes symmetry reduction algorithms. The `group.c` file includes functions for computing with permutations.

Compile and execute this verifier using commands analogous to those shown in §2.2:

```
C:\Program Files\TopSPIN_2.1\examples>gcc -o sympan -O2 -DSAFETY sympan.c group.c
C:\Program Files\TopSPIN_2.1\examples>sympan -m100000

(Spin Version 5.1.6 -- 9 May 2008)
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +

State-vector 108 byte, depth reached 2323, errors: 0
     4960 states, stored
    12254 states, matched
    17214 transitions (= stored+matched)
        6 atomic steps
hash conflicts:        35 (resolved)

    5.735       memory usage (Mbyte)

unreached in proctype client
        line 20, state 6, "-end-"
        (1 of 6 states)
unreached in proctype loadBalancer
        line 34, state 13, "-end-"
        (1 of 13 states)
unreached in proctype server
        line 43, state 7, "-end-"
        (1 of 7 states)
unreached in proctype :init:
        (0 of 9 states)

pan: elapsed time 0.369 seconds pan: rate 13441.734 states/second
```

Comparing this model checking result with the result without symmetry reduction (§2.2) shows that the *fast* strategy leads to a state-space reduction factor of 34.5. The theoretical maximum symmetry reduction factor is 48, which is the size of the symmetry group computed by TopSPIN, so this is a reasonably good result. Notice that speedup factor is just 2.2, and as a result the *states/second* rate for verification is significantly lower when symmetry reduction is applied.

2.5 Symmetry reduction with the *enumerate* strategy

Although the *fast* strategy provides effective symmetry reduction for the loadbalancer example, the strategy does not provide space-optimal symmetry reduction. The *enumerate* strategy, on the other hand, is guaranteed to provide full symmetry reduction.

To use the *enumerate* strategy, open `config.txt` in the `examples` directory, and add the following line:

```
strategy=enumerate
```

This additional option tells TopSPIN to use the *enumerate* strategy over the default *fast* strategy.

Now apply TopSPIN to the loadbalancer specification, and compile and run the generated verifier:

```
C:\Program Files\TopSPIN_2.1\examples>java -jar
"C:\Program Files\TopSPIN_2.1\TopSPIN.jar" loadbalancer.p

   ... TopSPIN output ...

C:\Program Files\TopSPIN_2.1\examples>gcc -o sympan -O2 -DSAFETY sympan.c group.c
C:\Program Files\TopSPIN_2.1\examples>sympan -m100000

(Spin Version 5.1.6 -- 9 May 2008)
        + Partial Order Reduction

Full statespace search for:
        never claim             - (none specified)
        assertion violations    +
        cycle checks            - (disabled by -DSAFETY)
        invalid end states      +

State-vector 108 byte, depth reached 1916, errors: 0
     4213 states, stored
    11181 states, matched
    15394 transitions (= stored+matched)
        6 atomic steps
hash conflicts:        23 (resolved)

    5.638       memory usage (Mbyte)

unreached in proctype client
        line 20, state 6, "-end-"
        (1 of 6 states)
unreached in proctype loadBalancer
        line 34, state 13, "-end-"
        (1 of 13 states)
unreached in proctype server
        line 43, state 7, "-end-"
        (1 of 7 states)
unreached in proctype :init:
        (0 of 9 states)

pan: elapsed time 0.647 seconds pan: rate  6511.592 states/second
```

Verification using the *enumerate* strategy provides a better reduction factor: 40.6 vs. 34.5 with the *fast* strategy (§2.4). This comes at an expense: the speedup factor is reduced from 2.2 for the *fast* strategy to 1.3. For specifications with more symmetric components, this time penalty is more significant: the *enumerate* strategy works by applying *every* symmetry associated with the input specification to *every* state encountered during model checking. Since a specification with n symmetric components has a symmetry group of size at least $n!$, use of the *enumerate* strategy quickly becomes infeasible.

2.6 Summary so far

You should now have successfully installed TopSPIN and its prerequisite components, created a configuration file, and applied the tool to a Promela example. If you have encountered any problems during this process then proceed to Chapter 5 for troubleshooting ideas and information on how to report TopSPIN bugs.

At this stage, you may have learned all you need to know about TopSPIN for your basic symmetry reduction needs. Chapters 3 and 4 are for advanced users who wish to explore TopSPIN's more sophisticated options, and compile the tool from source, respectively.

3 Overview of Options

This chapter will be expanded into a complete reference for the various TopSpin options.

4 Compiling From Source

4.1 Downloading TopSPIN source code

The TopSPIN source code is stored in a Subversion repository, hosted by the Department of Computing Science at the University of Glasgow at the following URL:

```
https://ouen.dcs.gla.ac.uk/repos/symmetry/
```

Use subversion to check out the TopSPIN source code to an appropriate location:

```
C:\prog>svn checkout https://ouen.dcs.gla.ac.uk/repos/symmetry/trunk TopSPINSource
A TopSPINSource/TestModels
A TopSPINSource/TestModels/EtchTesting
A TopSPINSource/TestModels/EtchTesting/ParsePassTests
A TopSPINSource/TestModels/EtchTesting/ParsePassTests/peterson
A TopSPINSource/TestModels/EtchTesting/ParsePassTests/hello
A TopSPINSource/TestModels/EtchTesting/ParsePassTests/pathfinder
A TopSPINSource/TestModels/EtchTesting/ParsePassTests/snoopy
A TopSPINSource/TestModels/EtchTesting/ParsePassTests/mobile1
A TopSPINSource/TestModels/EtchTesting/ParsePassTests/mobile2
A TopSPINSource/TestModels/EtchTesting/ParsePassTests/pftp
A TopSPINSource/TestModels/EtchTesting/ParsePassTests/leader
A TopSPINSource/TestModels/EtchTesting/ParsePassTests/loops
...
A TopSPINSource/Common/Minimising.gap
A TopSPINSource/Common/parallel_symmetry_cell_ppu.c
A TopSPINSource/Common/Verify.gap
A TopSPINSource/Common/WorkspaceGenerator.gap
A TopSPINSource/Common/parallel_symmetry_cell_spu.c
A TopSPINSource/Common/parallel_symmetry_cell_ppu.h
A TopSPINSource/symmextractor_common_config.txt
Checked out revision 75.
```

4.2 Generating the Promela parser

TopSPIN is based on a Promela parser which is constructed using the SableCC parser generator. Download SableCC version 3.2 from the Sable website[1], and generate the parser as follows (adapting the command according to where you have installed SableCC):

```
java -jar C:\sablecc-3.2\lib\sablecc.jar promela.grammar
```

This command should result in output similar to the following:

```
C:\prog\TopSPINSource>java -jar C:\sablecc-3.2\lib\sablecc.jar
promela.grammar

SableCC version 3.2 Copyright (C) 1997-2003 Etienne M. Gagnon <etienne.gagnon@uqam.ca>
and others.  All rights reserved.

This software comes with ABSOLUTELY NO WARRANTY.  This is free software,
and you are welcome to redistribute it under certain conditions.

Type 'sablecc -license' to view the complete copyright notice and license.

 -- Generating parser for promela.grammar in C:\prog\TopSPINSource
Adding productions and alternative of section AST.
Verifying identifiers.
```

1. http://sablecc.org/

```
Verifying ast identifiers.
Adding empty productions and empty alternative transformation if necessary.
Adding productions and alternative transformation if necessary.
computing alternative symbol table identifiers.
Verifying production transform identifiers.
Verifying ast alternatives transform identifiers.
Generating token classes.
Generating production classes.
Generating alternative classes.
Generating analysis classes.
Generating utility classes.
Generating the lexer.
 State: INITIAL
 - Constructing NFA.
.........................................................................
.........................................................................
 - Constructing DFA.
.........................................................................
.........................................................................
 - resolving ACCEPT states.
Generating the parser.
.........................................................................
.........................................................................
```

4.3 Compiling and creating a jar

You have now downloaded and generated all the Java source code for TopSPIN, and can proceed to compile this source code. The `javac` compiler and `jar` utility are required to compile the source and create an executable jar file respectively. The source code is all contained in directories within `src`.

4.3.1 Compiling

There are two options for compilation:

1. Create an Eclipse project from the TopSPIN source code, in which case Eclipse will automatically compile the code.
2. Use the `make` utility and the supplied `Makefile` to compile the source code, by typing the command `make classes` in the TopSPIN root directory. The provided `Makefile` uses the `sed` and `find` utilities, which are provided as standard with Linux, and are available on Windows via Cygwin.

Compiling the source code using the `Makefile` gives output along the following lines:

```
C:\prog\TopSPINSource>make classes
javac src/etch/checker/Check.java
Note: .\src\promela\parser\Parser.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
javac src/etch/checker/CheckerTest.java
javac src/etch/testing/EtchTestCase.java
javac src/etch/testing/EtchTester.java
javac src/group/Group.java
javac src/promela/analysis/ReversedDepthFirstAdapter.java
javac src/symmextractor/InlineReplacer.java
javac src/symmextractor/SymmExtractor.java
javac src/symmextractor/testing/SymmExtractorFailTestOutcome.java
javac src/symmextractor/testing/SymmExtractorRunTestOutcome.java
javac src/symmextractor/testing/SymmExtractorTestCase.java
javac src/symmextractor/testing/SymmExtractorTester.java
javac src/symmreducer/testing/ModelCheckingResult.java
javac src/symmreducer/testing/SymmReducerFailTestOutcome.java
javac src/symmreducer/testing/SymmReducerTestCase.java
javac src/symmreducer/testing/SymmReducerTester.java
javac src/testing/RunAllTests.java
```

Note the compile warnings generated for `Parser.java`: these are due to code generated by SableCC. All other files should compile without warnings.

4.3.2 Creating a jar file

If you are using Eclipse then you can create an executable jar file for TopSPIN by exporting your TopSPIN project as a jar, and selecting `src.TopSpin` as the *main* class.

If using the supplied `Makefile`, typing `make jars` will produce two jar files:

```
C:\prog\TopSPINSource>make jars

jar cmf manifest.txt TopSPIN.jar src/etch/checker/Check.class src/etch/checker/Checker.class
... <many .class files> ...
src/utilities/Strategy.class src/utilities/StringHelper.class src/promela/lexer/lexer.dat
src/promela/parser/parser.dat && echo "TopSPIN.jar built successfully."
TopSPIN.jar built successfully.

jar cmf tests_manifest.txt TopSPINTests.jar src/etch/checker/Check.class
src/etch/checker/Checker.class
... <many .class files> ...
src/utilities/Strategy.class src/utilities/StringHelper.class src/promela/lexer/lexer.dat
src/promela/parser/parser.dat && echo "TopSPINTests.jar built successfully."
TopSPINTests.jar built successfully.
```

Note that you can skip the `make classes` step, and type `make jars` to both compile the Java files and produce jar files. To delete all jar and class files, use `make clean`.

4.4 Try your compiled version on an example

Now that you have successfully compiled a jar file from the TopSPIN source, you are effectively in the same position as a user who has downloaded the TopSPIN jar file using the instructions given in §1.2. Of course, you have the advantage of being able to modify TopSPIN to suit your purposes!

The next steps involve compiling *saucy*, creating a GAP workspace, and testing TopSPIN on a simple example. Therefore you should work your way through Chapters 1 and 2, using your compiled jar file in place of the downloaded jar file.

4.5 Acceptance Tests

The TopSPIN source checkout includes a reasonably large set of acceptance tests. It is highly recommended that you run these tests using the instructions below before commencing any development work on TopSPIN – passing the acceptance tests confirms that you are starting from a stable base. The acceptance tests can also be run regularly during development, to ensure that the addition of new features to TopSPIN does not adversely affect the tool's existing features.

Important Before running the acceptance tests, make sure that the GCC tool is installed in such a way that it can be invoked by name from a command prompt, i.e. so that typing `gcc` will launch GCC. To check this, try typing `gcc` from a fresh prompt. You should see something like:

```
C:\>gcc
gcc: no input files
```

4.5.1 Setting up a configuration file for testing

The TopSPIN test suite uses a called `symmextractor_common_config.txt` for some configuration options which apply to most tests. Open this file, and edit the `gap` line to use the appropriate command for your setup. You should not have to change the `Common` or `saucy` lines: the source code checkout is organised so that the `saucy` and `Common` directories are sub-directories of the directory in which `symmextractor_common_config.txt` is contained.

4.5.2 Running the tests

You can now run the acceptance tests via the `TopSPINTests.jar` file:

```
java -ea -jar TopSPINTests.jar 2> temp.txt
```

The `-ea` argument switches on assertion checking, which is useful for finding potential problems with TopSPIN. The final part of the command, `2> temp.txt`, pipes error messages generated during testing to the file `temp.txt`. This is useful since many of the tests are *fail* tests, which check that the TopSPIN typechecker correctly rejects Promela specifications which are not suitable for processing by TopSPIN. Redirecting these error messages to a text file means that the screen is not cluttered with error messages.

You should see something like this when you run the tests:

```
ETCH TESTS
==========
   [PASS] expected and actual: BadlyTyped, file: TestModels/EtchTesting/FailTests/failrec...
   [PASS] expected and actual: WellTyped, file: TestModels/EtchTesting/PassTests/testtele...
   [PASS] expected and actual: WellTyped, file: TestModels/EtchTesting/PassTests/test_ex_...
   [PASS] expected and actual: WellTyped, file: TestModels/EtchTesting/PassTests/testtele...
   [PASS] expected and actual: BadlyTyped, file: TestModels/EtchTesting/FailTests/faildup...
   [PASS] expected and actual: ParsePass, file: TestModels/EtchTesting/ParsePassTests/pet...
   ...


SYMMEXTRACTOR TESTS
===================
   [PASS] expected and actual: (well typed, group size = 72, coset search: no), file: Tes...
   [PASS] expected and actual: (well typed, group size = 1296, coset search: no), file: T...
   [PASS] expected and actual: BreaksRestrictions, file: TestModels/SymmExtractorTests/Ba...
   [PASS] expected and actual: (well typed, group size = 3628800, coset search: no), file...
   [PASS] expected and actual: BreaksRestrictions, file: TestModels/SymmExtractorTests/Ba...
   ...


SYMMREDUCER TESTS
=================
   [PASS] expected and actual: ((well typed, group size = 120, coset search: no), no. sta...
   [PASS] expected and actual: ((well typed, group size = 3628800, coset search: no), no....
   [PASS] expected and actual: ((well typed, group size = 6, coset search: no), no. state...
   [PASS] expected and actual: ((well typed, group size = 5040, coset search: no), no. st...
   [PASS] expected and actual: ((well typed, group size = 720, coset search: no), no. sta...
   ...


Summary:
   238 passes
   0 fails


Acceptance tests passed - you may commit your changes!
```

There are in the order of 250 test cases. These are divided into ETCH tests, which test the typechecking component of TopSPIN; SymmExtractor tests, which check the symmetry detection capabilities of the tool, and SymmReducer tests, which perform symmetry reduction on Promela examples, checking that verification results for these examples are as expected. The ETCH tests run very quickly, the SymmExtractor tests run at a moderate speed, and the SymmReducer tests run fairly slowly. Testing takes approximately 10 minutes on an average PC.

4.5.3 Problems running the tests

If you have compiled a fresh checkout of TopSPIN then all of the acceptance tests should pass, since their passing is a condition for committing changes to the source code repository. Therefore, if you have any problems running the tests this is likely due to a problem with the way you have set up GAP, *saucy*, SPIN, or GCC. Have a look at §5.1 which details common problems encountered when using TopSPIN. If test cases continue to fail then please send a bug report to the TopSPIN development team: see §5.3 for details.

5 Troubleshooting and Bug Reporting

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.

5.1 Common problems

5.1.1 Missing configuration file

Problem: There is no file named `config.txt` in your working directory when you run TopSPIN.

Example error message:

```
Error opening configuration file "config.txt", which should be
located in the directory from which you run TopSPIN.
```

Solution: Follow the instructions in §2.3 on how to create `config.txt`.

5.1.2 The *saucy* program is not correctly installed

Problem: You may not have correctly installed and compiled *saucy*, the graph automorphism program on which TopSPIN relies.

Example error message:

```
Error launching saucy with command: C:\Program Files\TopSPIN_2.1\saucy\saucy.exe -d
"C:\Program Files\TopSPIN_2.1\Common\graph.saucy"
java.io.IOException: CreateProcess: C:\Program Files\TopSPIN_2.1\saucy\saucy.exe -d
"C:\Program Files\TopSPIN_2.1\Common\graph.saucy" error=2
        at java.lang.ProcessImpl.create(Native Method)
        ... rest of Java stack trace
```

Solution: Source code for *saucy* is included with the TopSPIN distribution, which can be downloaded via the instructions of §1.2. However, you need to compile the *saucy* source code using GCC. To do this, follow the instructions given in §1.3.1.

5.1.3 Path to *saucy* in configuration file is wrong

Problem: From TopSPIN's point of view this is the same as the previous problem. From your point of view there is a difference: you may have correctly installed and compiled *saucy*, but but mistyped the path to *saucy* in `config.txt`.

Example error message:

```
Error launching saucy with command: C:\Program Files\TopSPIN_2.1\saucy\tsaucy.exe -d
"C:\Program Files\TopSPIN_2.1\Common\graph.saucy"
java.io.IOException: CreateProcess: C:\Program Files\TopSPIN_2.1\saucy\tsaucy.exe -d
"C:\Program Files\TopSPIN_2.1\Common\graph.saucy" error=2
        at java.lang.ProcessImpl.create(Native Method)
        ... rest of Java stack trace
```

Note that TopSPIN is trying to launch `tsaucy.exe` rather than `saucy.exe`, due to a typo in `config.txt`.
Solution: Make sure *saucy* is correctly downloaded and compiled (§1.3.1), and ensure that `config.txt` contains a line for *saucy* with *exactly* the absolute path to the tool (§2.3).

5.1.4 GAP is not correctly installed

Problem: You may not have correctly installed GAP, the computational group theory package on which TopSPIN relies.

Example error message:

```
Starting GAP with command: C:\gap4r4\bin\gap.bat -L
"C:\Program Files\TopSPIN_2.1\Common\gapworkspace" -q
Exception in thread "main"
java.io.IOException: CreateProcess: C:\gap4r4\bin\gap.bat -L
"C:\Program Files\TopSPIN_2.1\Common\gapworkspace" -q error=2
        at java.lang.ProcessImpl.create(Native Method)
        ... rest of Java stack trace
```

Solution: Download and install GAP from the GAP website. The URL for this website and the version of GAP which TopSPIN supports are given in Figure 1.1 (§1.1).

5.1.5 Path to GAP in configuration file is wrong

Problem: From TopSPIN's point of view this is the same as the previous problem. From your point of view there is a difference: you may have correctly installed GAP, but mistyped the path to GAP in `config.txt`.

Example error message:

```
Starting GAP with command: C:\gap4r4\bin\tgap.bat -L
"C:\Program Files\TopSPIN_2.1\Common\gapworkspace" -q
Exception in thread "main"
java.io.IOException: CreateProcess: C:\gap4r4\bin\tgap.bat -L
"C:\Program Files\TopSPIN_2.1\Common\gapworkspace" -q error=2
        at java.lang.ProcessImpl.create(Native Method)
        ... rest of Java stack trace
```

Note that TopSPIN is trying to launch `tgap.bat` rather than `gap.bat`, due to a typo in `config.txt`.
Solution: Make sure you have correctly downloaded and installed GAP(§1.1), and ensure that `config.txt` contains a line for GAP with *exactly* the absolute path to the tool (§2.3).

5.1.6 Common directory does not exist, or user does not have permissions for this directory

Problem: The location of the TopSPIN `Common` directory has been specified incorrectly in `commmon.txt`.

Example error message:

```
Error while trying to create file "C:\Program Files\TopSPIN_2.1\Common\graph.saucy".
Make sure that the directory C:\Program Files\TopSPIN_2.1\Common\ exists,
and that you have write permission.
```

Solution: Make sure that `config.txt` contains a line of the form `common=`*name*, where *name* is the *absolute* path to the `Common` directory provided with the TopSPIN distribution. Make sure this path has not been mistyped, and that the path includes a terminating slash.

5.1.7 Bad GAP workspace

Problem: The file `gapworkspace` in the `Common` directory is either missing or corrupted.

Example error message:

```
Starting GAP with command: C:\gap4r4\bin\gap.bat -L
"C:\Program Files\TopSPIN_2.1\Common\gapworkspace" -q
Error -- bad GAP workspace specified in configuration file.

GAP produced errors:
====================
gap: Press <Enter> to end program
End of GAP errors
```

Solution: Check that:

1. You have followed the instructions in §1.3.2 on creating a GAP workspace
2. This workspace has been successfully saved in the `Common` directory

3. You have not changed the version of GAP you are using since creating the workspace
4. The workspace was created exactly the same machine, with the same operating system, on which you are running TopSPIN.

5.1.8 SPIN is not installed properly

Problem: You have not correctly installed SPIN in such a way that the tool can be invoked by simply typing `spin`, as discussed in §1.1.

Example error message:

```
An error occurred while constructing the "sympan" files.
java.io.IOException: CreateProcess: spin -a loadbalancer.p error=2
        at java.lang.ProcessImpl.create(Native Method)
        ... rest of Java stack trace
```

Solution: Make sure SPIN is correctly installed. Perhaps you have renamed `spin` to `spin-linux` or `spin516`; maybe you do not have execute permission for SPIN, or perhaps you have another application called `spin` installed on your machine. Resolve these issues so that `spin` is all that needs to be typed to launch the SPIN tool.

5.2 Limitations of TopSPIN

TopSPIN currently supports the verification of *safety* properties, expressed using assertions or monitor processes. The tool does not yet support symmetry-reduced LTL model checking using never claims.

5.3 Reporting bugs in TopSPIN

If you have a problem using TopSPIN, first check to see if your problem is covered by one of the *common problems* in §5.1. If this is not the case, then please take the time to report your bug to the TopSPIN developers, so that it can be immediately documented and ultimately fixed, for the benefit of you and other TopSPIN users.

Bug reports should be submitted using the contact details in §5.5. Please provide the following when reporting a bug:

- A description of the problem
- A test-case (example Promela specification and `config.txt` file) which exposes the problem
- The TopSPIN version you are using, the approximate date on which you downloaded TopSPIN
- Details as to whether you are using a pre-compiled version of TopSPIN, or a version you have compiled from source
- Any ideas you have as to what may have gone wrong!

Please note that bugs do not have to relate to the correctness of TopSPIN – a valuable bug report could be in response to a misleading error message generated by TopSPIN, where there *is* something wrong with the input specification, but the problem is not what the error message indicates. Please feel free to also submit suggestions for features to be added to TopSPIN.

5.4 Reporting bugs in this manual

Please also get in touch (§5.5) if you find typos, inconsistencies or ambiguities in this manual – this kind of feedback is extremely valuable.

5.5 Getting in touch

All correspondence related to TopSPIN should be by email, to Alastair Donaldson: `ally@codeplay.com`.

Acknowledgements

- TopSPIN was originally developed by Alastair Donaldson at the University of Glasgow, between 2003 and 2006.
- TopSPIN uses symmetry reduction theory developed by Alastair Donaldson and Alice Miller. This theory is in turn based on a decade of research by the model-checking community.
- The approach taken by the TopSPIN implementation was influenced greatly by the SymmSpin tool, developed by Dragan Bosnacki, Denis Dams and Lezek Holenderski.
- The layout for this manual was inspired by the GAP reference manual.
- Thanks to the developers of GAP, *saucy* and SableCC: the development of TopSPIN would have taken a lot longer without these excellent tools.