Artifacts for paper Scalable Sampling of Highly-Configurable Systems: Generating Random Instances of the Linux Kernel

Anonymous Author(s)

ACM Reference Format:
Anonymous Author(s). 2022. Artifacts for paper Scalable Sampling of Highly-Configurable Systems: Generating Random Instances of the Linux Kernel. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The artifacts include the logic translations for the models featured in the paper, the forests of bdds, the samples generated with our tools and the tools necessary to build the forests, create samples and a script to extract the individual configurations from a sample file. In the following sections we describe how to compile and use the tools, as well as the file formats used.

2 COMPILATION

This artifact has been tested on Linux and macOS. It requires recent versions of flex and bison. In macOS you can get then using brew or mac ports. You can get them in Linux by typing:

```
sudo apt install flex bison libfl-dev
```

To compile the tools, just type:

```
cd code
./configure
make
cd ..
```

The binaries should be in code/bin and scripts. Add these directories to your PATH. Each binary includes usage information.

3 USAGE

The models include several file types:

- **.var** : The original ordering of the Boolean variables of the translation of a model
- **.ord** : The variable ordering actually used to build the forest of BDDs
- **.exp** : A file of propositional logic constraints
- **.cnf** : A DIMACS file containing the CNF translation obtained applying the Tseitin transformation to the original .exp file
- **.dddmp** : A BDD or forest of BDDs in DDDMP format. The header is in readable format
- **.log** : A log of the forest of BDDs creation, including the command used to build it in the first line.

The .ord file contains the variable names in a suitable order to build the forest of BDDs. The .exp file contains the rest of the information regarding the Extensible Logic Groups (ELGs). Each group starts with a comment containing the name of the first output variable in the group, followed by a number of propositional formulae, one per line. The constraint of the group is the conjunction of the formulae. The other output variables for each group can be derived from the .ord file. For instance, the .exp file for axtls begins with:

```
# bool_AXHTTPD
# bool_BINDINGS
# bool_DEBUG
# bool_OPENSSL_COMPATIBLE
# bool_SAMPLES
# bool_SSL_CTX_MUTEXING
# bool_SSL_GENERATE_X509_CERT
# bool_PLATFORM_LINUX
not bool_PLATFORM_LINUX or not bool_CONFIG_PLATFORM_WIN32
not bool_PLATFORM_CYGWIN or not bool_CONFIG_PLATFORM_WIN32
not bool_PLATFORM_CYGWIN or not bool_CONFIG_PLATFORM_LINUX
not bool_PLATFORM_LINUX or not bool_CONFIG_PLATFORM_CYGWIN
not bool_PLATFORM_LINUX or not bool_CONFIG_PLATFORM_WIN32
not bool_PLATFORM_CYGWIN or not bool_CONFIG_PLATFORM_WIN32
bool_PLATFORM_LINUX or bool_PLATFORM_CYGWIN \
   or bool_PLATFORM_WIN32
```

This represents eight ELGs. For the first seven, the constraint is true. For the last one, the constraint means that exactly one of the platforms is true and the others are false.

To create the .dddmp file for, say, the axtls model, you may cd to the models directory and type:

```
Logic2BDD -base axtls-manybdds.dddmp -manybdds -score file \
  axtls-manybdds.ord axtls-manybdds.var axtls-manybdds.exp
```

To create a file of 100 samples from the forest previously created, type:

```
KconfigSampler 100 axtls-manybdds.dddmp >samples.txt
```

To extract sample number 7 from the samples.txt file to a file named .config, type

```
extractConfig.sh 7 samples.txt >.config
```

You can use the provided forest files (.dddmp) or re-create them from the logical translations using Logic2BDD. The sample files produced by KconfigSampler include one complete configuration per line. The provided samples were used in the tables in the paper to test speed and are not decoded, (i.e., they are just a bunch of zeros and ones instead of Name=Value pairs). You can create decoded samples with KconfigSampler and get one configuration per line in Name=Value format. Extract them to .config files with the extractConfig.sh script to adequately handle negated booleans.

Conference'17, July 2017, Washington, DC, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn