The BULL Package: Installation Guide

October 12, 2012

The BULL (Boolean fUnction Learning Library) package has been compiled and tested on Linux and MacOS 10.6.8. It is freely available under the open source LGPL v3 license. The project web-site is `http://code.google.com/p/project-bull/`.

1 Get the Package

The package can be obtained via SVN:
`svn co http://project-bull.googlecode.com/svn/trunk/ project-bull-read-only`
(note that there is a space between "`...trunk/`" and "`project-bull-read-only`".)
It consists of the following 6 folders:

Src/core
: The core library code in C.

Src/c
: Example of an oracle implemented in C.

Src/cpp
: Example of an oracle implemented in C++.

Src/java
: Example of an oracle implemented in JAVA.

Src/ocaml
: Example of an oracle implemented in OCaml.

Src/solvers
: SAT Solvers used by the oracles. Currently we use minisat as the default solver for C++ and OCaml, SAT4J as the default for JAVA.

dimac
: CNF formulae that can be used as the target functions, taken from `ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf/`.

2 Prerequisites

1. GNU C,C++ complier and the make utility
2. JAVA developer kit $\geq$1.6 (needed for the JAVA oracle example)
3. OCAML 3.12.1 (needed for the OCaml oracle example, we assume OCaml libraries are in /opt/local or /usr/local)
4. Minisat 2.2.0 for the C and OCaml oracle examples and SAT4J for the JAVA oracle example. (For the convenience of the users, we pack the source code of minisat and the jar file of SAT4J into our tool.)

3 Compilation

We use `{BULL-dir}` to denote the root folder of BULL. The easiest way is to run the ./build script in the root folder of BULL. It will automatically compile solvers, the core library, and oracles in different programming languages For users who only need a part of the package, we describe how to compile each component in detail below.

3.1 The Core Library

1. `cd {BULL-dir}/Src/core`
2. make

3.2 Solvers

Before the compilation of the C or OCaml oracles, users have to first compile the minisat tool. For convenience, source code of minisat 2.2.0 is included in our package.

1. `cd {BULL-dir}/Src/solvers/minisat/core`
2. `g++ -c  -O2 -D __STDC_LIMIT_MACROS -D __STDC_FORMAT_MACROS -I../ Solver.cc`

3.3 Oracles

1. `cd {BULL-dir}/Src/[c, java, ocaml]`
2. make

4 Execution

Below we show how to run learning algorithms using the example oracles implemented in C, OCaml, and JAVA. We use `{Exec}` to denote the location of the executables.

- For C, `{Exec} = {BULL-dir}/Src/c/learn`
- For JAVA, `{Exec} = {BULL-dir}/Src/java/learn.sh`
- For OCaml, `{Exec} = {BULL-dir}/Src/ocaml/learn.btye`

Users can specify which learning algorithm to use via passing input arguments.

- `{Exec}` (without any input arguments): using the CDNF algorithm
- `{Exec} 0`: using the CDNF+ algorithm
- `{Exec} 1`: using the CDNF++ algorithm

The executable reads a target function (a cnf formula in DIMACS format) from standard input. For example, the command

```
{BULL-dir}/Src/java/learn.sh 0 < ../../dimacs/aim/aim-50-1_6-yes1-1.cnf
```

using the CDNF+ learning algorithm to learn a target function that equals the cnf formula in `aim-50-1_6-yes1-1.cnf` via java interface.