PyMBD - A Library of MBD Algorithms and a Light-weight Evaluation Platform

This package provides a set of model-based diagnosis algorithms written in Python/C++ and some scripts to conduct experimental evaluations using those algorithms and different scenarios. It includes two ready-made scenarios:

- ISCAS85 circuit diagnosis [1]
- LTL-Specification diagnosis [2], [3]

PyMBD also includes some pure-MHS (minimal hitting set) algorithms as well that do not compute conflicts on-the-fly but rely on pre-computed ones.

Installation and Prerequisites

PyMBD requires:

- Python 2.7 (tested with 2.7.3)
- Some Python packages, that can be installed via `install.sh`:
    - bitstring, tested with version version 2.1.1
    - blist, tested with version 1.3.4
    - python-graph-core, tested with version 1.8.0
    - pyparsing, tested with version 2.0.1
- numpy (tested with 1.6.0) and scipy (tested with 0.9.0), (for the STACCATO MHS algorithm only)
- C++ compiler with C++11 support for the C++-Algorithms (tested with Apple LLVM version 5.1 under OS X 10.9.2)
- Boost-Library (tested with version 1.55.0) [http://www.boost.org/]
- swig (tested with version 2 and 3) [http://www.swig.org/]
- gnuplot and pdflatex if you want to generate graphs

PyMBD is intended to be run directly from the directory where you extracted it. PyMBD uses several backends (solvers and a particular parser engine) that you might need to compile or provide as a binary. You can build everything by executing `lib/build_resources.sh`.

- **Yices** (SMT Solver) -- **NOT PROVIDED**

  copy the Yices1 binary (tested with version 1.0.29) for your platform from the Yices website to `lib/yices`

- **PicoSAT** (SAT Solver)

PicoSat source (version 936, http://fmv.jku.at/picosat/) is provided in the `lib/picosat` directory and can be compiled using `./build.sh`

- **SCryptoMinisat** (SAT Solver)

  SCryptoMinisat source (http://amit.metodi.me/research/scrypto/SCrypto_src.zip) is provided in the `lib/SCryptoMinisat` directory and can be compiled using `./build.sh`

- **DParser** (Parser Library/Generator for parsing the LTL formulae)

  DParser (http://dparser.sourceforge.net/) source is provided in the `lib/dparser` directory and can be compiled using `./build.sh`. This also builds a CPython module in the `lib/dparser/python` directory.

The binaries we ship target OS X 10.9.

C++-Algorithms

Some algorithms have been built using C++ and the Boost::Python library. To compile those algorithms into a CPython module run the script `cresources/src/build-module.sh`.

Running Experiments

The script `run.sh` provides some examples on how to run experiments using PyMBD. We provide three ready-made experiments (ISCAS, LTL and MHS) as a demonstration. Each experiment can host several runs (for example, with different parameters or to compare algorithmic changes). In order to keep the scenarios flexible, each experiment uses its own set of computation scripts and files:

- `compute.py` can run a set of algorithms on a set of samples (problems) and save the results into a subdirectory (a so-called run). Run `./compute.py -h` for a complete set of arguments.
- `data_structures.py` contains the definition of the input and output files (that is, statistics that get recorded into `results-*.txt` files).
- `generate_problems.py` is used to (re-)generate the input data (for example, injecting random faults into ISCAS circuits)
- `plot_graphs.py -r run01` uses the `results-*.txt` files from a finished run (`run01`) and aggregates them into datafiles ready for plotting the results. They are placed into a `graphs` subdirectory and also runs `gnuplot` and `pdflatex` for any gnuplot file found in there.

License

See file LICENSE.

---

1. M. Hansen, H. Yalcin, and J. P. Hayes, Unveiling the ISCAS-85 bench- marks: A case

study in reverse engineering, IEEE Design and Test, 6, 72–80, (1999). http://www.cbl.ncsu.edu:16080/benchmarks/ISCAS85/ ↩

2. Ingo Pill and Thomas Quaritsch. Behavioral Diagnosis of LTL Specifications at Operator Level. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013), Beijing, China. August 2013. Available from: http://ijcai.org/papers13/Papers/IJCAI13-160.pdf↩
3. Ingo Pill and Thomas Quaritsch. Exploiting Parse Trees in LTL Specification Diagnosis. In Proceedings of the 24th International Workshop on Principles of Diagnosis (DX 2013), Jerusalem, Israel. October 2013. Available from: http://www.ist.tugraz.at/pill/downloads/DX13aPillQuaritsch.pdf↩