An Introduction to `claspfolio` *

Stefan Ziller Martin Gebser Benjamin Kaufmann
Torsten Schaub **

August 10, 2010

Abstract

This document gives an overview of the Answer Set Programming (ASP; [1]) solver `claspfolio`, developed at the University of Potsdam.

`claspfolio` is a portfolio solver for ASP that makes use of different configurations of the `clasp` [3, 2] solver. Before solving, the features of an input logic program are extracted by means of `claspre`. These features are then utilized by `claspfolio` for algorithm selection, in order to launch the most promising configuration of clasp for solving. Algorithm selection is done by appeal to machine-learning techniques.

In 2009, `claspfolio` was the best single-system solver in the global ranking of the Second ASP Competition.

*Tool `claspfolio` is available at [5].

**{ziller,gebser,kaufmann,torsten}@cs.uni-potsdam.de

Contents

1 Functionalities

The pre-processing tool `claspfolio` is mostly written in `Python` and published under GNU General Public License [4]. Sources are available at [5]. `claspfolio` serves three main purposes: (1) solve a given logic program. (2) select the most promising conguration of clasp for solving (3) predict satisfiability of a given logic program. In case (1), `claspfolio` ist used such as other ASP solver. In case (2), the selected configuration is not run, but only print out for analysing. In case (3), only the sat prediction is run.

We now describe on examples how `claspfolio` is utilized for its two functionalities.

1.1 Use `claspfolio` to solve a Logic Program

By default, `claspfolio` solve an input logic program. As an example, we take a Blocks-World problem from [2]. The invocation looks as follows:[1]

```
gringo blocks.lp world4.lp --ifixed 9 | \
claspfolio.py
```

This makes `claspfolio` output the following:

```
1  claspfolio version 0.8 (Models based on clasp 1.3.4)
2  ----------------------
3  Reading from pipe    ...
4  Running clasp(pre)   ...
5  Algorithm selection  ...
6  SAT-score          : 0.689 (SATISFIABLE)
7  Solving              ...
8  clasp version 1.3.4
9  Reading from stdin
10 Solving...
11 Answer: 1
12 move(b10,b2,1) move(b9,table,2) move(b4,b9,3) move(b8,b3,4) move(b7,b8,5) move(b10,b6,6)
13  move(b2,b10,7) move(b1,b2,8) move(b0,b4,9)
14 SATISFIABLE

16 Models      : 1+
17 Time        : 0.140s (Solving: 0.10s/1st Model: 0.10s/Unsat: 0.00s)
```

If the input logic program is solved while preprocessing, it looks like in the following example:

```
gringo blocks.lp world1.lp --ifixed 9 | \
claspfolio.py
```

This makes `claspfolio` output the following:

```
1  claspfolio version 0.8 (Models based on clasp 1.3.4)
2  ----------------------
3  Reading from pipe    ...
4  Running clasp(pre)   ...
5  Answer: 1
6  move(b3,b2,1) move(b1,b0,2) move(b3,b1,3) move(b3,b2,4) move(b1,table,5) move(b3,table,6)
7   move(b1,b0,7) move(b2,b1,8) move(b3,b2,9)
```

1.2 Select the most promising conguration of `clasp` for solving

The second functionality of `claspfolio` consists of giving the selected the most promising conguration of `clasp` for solving.

Using the same example, the invocation looks as follows::

[1] The "\" in command-line calls indicates that line breaks are escaped and used only for readability.

```
gringo blocks.lp world4.lp --ifixed 9 | \
claspfolio.py -s
```

The resulting output is as follows:

```
claspfolio version 0.8 (Models based on clasp 1.3.4)
----------------------
Reading from pipe    ...
Running clasp(pre)   ...
Algorithm selection  ...
SAT-score          : 0.689 (SATISFIABLE)
Choosen solver     : 29 (clasp --local-r --heu=VSIDS --del=3,1.1,1000
 --restarts=100,1.5,20000, score: 0.717)
```

If the input logic program is solved while preprocessing, it looks like in the following example:

```
gringo blocks.lp world1.lp --ifixed 9 | \
claspfolio.py -s
```

This makes `claspfolio` output the following:

```
claspfolio version 0.8 (Models based on clasp 1.3.4)
----------------------
Reading from pipe    ...
Running clasp(pre)   ...
Choosen solver     : claspre (while preprocessing)
```

1.3 Predict SAT / UNSAT on a given benchmark

A third functionality of `claspfolio` consists of predict the probability for the satisfability of giving logic program.

Using the same example, the invocation looks as follows::

```
gringo blocks.lp world4.lp --ifixed 9 | \
claspfolio.py --sat
```

This makes `claspfolio` output the following:

```
claspfolio version 0.8 (Models based on clasp 1.3.4)
----------------------
Reading from pipe    ...
Running clasp(pre)   ...
SAT-prediction       ...
Instance is        : SATISFIABLE (score: 0.689)
```

If the input logic program is solved while preprocessing, it looks like in the following example:

```
gringo blocks.lp world1.lp --ifixed 9 | \
claspfolio.py --sat
```

This makes `claspfolio` output the following:

```
claspfolio version 0.8 (Models based on clasp 1.3.4)
----------------------
Reading from pipe    ...
Running clasp(pre)   ...
SAT-prediction       ...
Instance is        : SATISFIABLE (solved while preprocessing)
```

1.4 Use stats to find best solver for a set of logic programs

At least you can see how often a solver is choosen to solve a logic program from a given directory.

The invocation looks as follows::

```
claspfolio.py --dir benchmarks -s
```

Using e.g. a directory with 5 benchmarks this makes `claspfolio` output the following:

```
claspfolio version 0.8 (Models based on clasp 1.3.4)
----------------------
Reading from       : benchmarks/qeq.12.lp
Running clasp(pre)   ...
Algorithm selection  ...
SAT-score          : 0.0 (UNSATISFIABLE)
Choosen solver     : 26 (clasp 1 --sat-pre=20,25,120 --trans-ext=dynamic
 --initial-look=10 --restarts=no, score: 0.998)

Reading from       : benchmarks/gryzzles.15.lp.lp
Running clasp(pre)   ...
Algorithm selection  ...
SAT-score          : 0.999 (SATISFIABLE)
Choosen solver     : 28 (clasp --restarts=256 --save-p, score: 0.995)

Reading from       : benchmarks/15-puzzle.ngrnd.10-step.lp
Running clasp(pre)   ...
Choosen solver     : claspre (while preprocessing)

Reading from       : benchmarks/knights_10_10_t14.lp.lp
Running clasp(pre)   ...
Choosen solver     : claspre (while preprocessing)

Reading from       : benchmarks/blockedqueens.28.1449787934.lp
Running clasp(pre)   ...
Choosen solver     : claspre (while preprocessing)

Summary
=======
Solver                     : claspre (while preprocessing) frequency: 0.6
Solver                     : 28 (clasp --restarts=256 --save-p) frequency: 0.2
Solver                     : 26 (clasp 1 --sat-pre=20,25,120 --trans-ext=dynamic
 --initial-look=10 --restarts=no) frequency: 0.2

Number of files            : 5.0
Number of files with error : 0.0

Most frequent solver       : 28 (clasp --restarts=256 --save-p) number: 1/5
Best predicted solver      : 19 (clasp --sat-prepro=yes) score : 1.382
```

2 Summary of `claspfolio` Options

This section gives a quick overview of command-line options provided by `claspfolio` to configure the functionalities described in Section 1. Beyond these, `claspfolio` inherits many command-line options from `clasp` (cf. [2]), allowing for the customization of solving. Further options of `claspfolio` are listed below:

`--help`
Show help message and exit.

`--verbose`
Verbosity level.

`--info`
Show model information and exit.

--skip-solving
Print the result of algorithm selection only.

--bin-path
Path to the directory containing the clasp binaries.

--svm-path
Path to the directory containing the libsvm binary.

--model-path
Path to the directory containing the models.

--file
Read from file instead of stdin.

--dir
Directory with logic programs, gives the likelihood of used solver at the end.

--recursive
Iterate over (logic program) files in subdirectories recursively.

--fstats
Print claspfolio stats, e.g. predicted scores.

--sat
Predict only SAT / UNSAT score.

--mode
Use models trained on SAT / UNSAT benchmarks for algorithm selection. Valid: all (default), sat, unsat, su, random

References

[1] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003. 1

[2] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele. A user's guide to `gringo`, `clasp`, `clingo`, and `iclingo`. Available at `http://potassco.sourceforge.net`. 1, 3, 5

[3] M. Gebser, B. Kaufmann, and T. Schaub. The conflict-driven answer set solver *clasp*: Progress report. In E. Erdem, F. Lin, and T. Schaub, editors, *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*, volume 5753 of *Lecture Notes in Artificial Intelligence*, pages 509–514. Springer-Verlag, 2009. 1

[4] GNU general public license. Free Software Foundation, Inc. `http://www.gnu.org/copyleft/gpl.html`. 3

[5] Potsdam answer set solving collection. University of Potsdam. `http://potassco.sourceforge.net/`. 1, 3