An Introduction to `claspre` *

Stefan Ziller Martin Gebser Benjamin Kaufmann
Torsten Schaub **

October 7, 2009

Abstract

This document gives an overview of the Answer Set Programming (ASP; [1]) tool `claspre`, developed at the University of Potsdam. `claspre` is based on the ASP solver `clasp` [4, 2] and specialized to pre-processing functionalities.

By default, `claspre` prints a pre-processed version of the input logic program in Smodels Internal Format [8]. This enables ASP solvers to make use of `clasp`'s advanced pre-processing [3], including for instance equivalence reasoning. Command-line options can be used for customization, e.g., `--trans-ext` may be configured to compile extended rules into normal ones.

As a second functionality, `claspre` allows for extracting static and dynamic features of logic programs. The latter are obtained via terminable solving, re-using the search engine of `clasp`. Command-line options can be used to customize the maximum amount of initial solving with `claspre`, done in order the extract dynamic features.

*Tool `claspre` is available at [6].

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

Contents

1 Functionalities

The pre-processing tool `claspre` is written in `C++` and published under GNU General Public License [5]. Sources are available at [6]. `claspre` serves two main purposes: (1) obtaining a compact representation of a logic program or (2) extracting features from a logic program. In case (1), an ASP solver that accepts Smodels Internal Format [8] can start from the pre-processed version of an input logic program. In case (2), `claspre` collects features of the input program, both static and dynamic ones (the latter determined through commenced solving), which can be used to analyze the program at hand. In either case, command-line options, such as `--trans-ext` for compiling extended rules into normal ones, can be provided to customize the pre-processing.

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

1.1 Pre-Processing a Logic Program

By default, `claspre` prints the pre-processed version of an input logic program in Smodels Internal Format, obtained by applying techniques described in [3]. As an example, we take a Blocks-World problem from [2]. The invocation looks as follows:[1]

```
gringo blocks.lp world4.lp --ifixed 9 | \
claspre
```

This makes `claspre` output the following:

```
1  1 2 0 0
2  2 179 121 0 1 58 59 60 ...
3  ...
4  4055 move(b10,b9,9)
5  0
6  B+
7  2
8  0
9  B-
10 1
11 180
12 655
13 1141
14 1627
15 2113
16 2599
17 3085
18 3571
19 4057
20 0
21 1
```

The output can then directly be processed by another solver, such as `smodels` [7]:

```
gringo blocks.lp world4.lp --ifixed 9 | \
claspre | smodels
```

1.2 Extracting Features of a Logic Program

The second functionality of `claspre` consists of extracting features. To this end, `claspre` provides option `--noLP` to suppress the output of Smodels Internal Format and option `--features` to print statistic information about a logic program. Features are distinguished into static and dynamic ones. While the former are printed only once, some of the latter are grabbed on each restart. For controlling the efforts spent on the

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

extraction of dynamic features, options `--endC` and `--endR` allow for limiting the number of conflicts and restarts, respectively, to be conducted by `claspre`. In fact, `claspre` stops its solving as soon as one of the limits is reached. The restart strategy and other behaviors of `claspre` can be set via options inherited from `clasp` (cf. [2]).

As a typical use of `claspre` to extract features, consider the following invocation:

```
gringo blocks.lp world4.lp --ifixed 9 | \
claspre --noLP --features --restarts 50,1 --endR 3
```

The resulting output is as follows:

```
claspre 0.7.9 based on clasp 1.2.0
Reading from stdin
Reading      : Done(0.000s)
Preprocessing: Done(0.020s)
Solving...

Features of lp:

runtime features...
Iteration             : 1
maxLearnt             : 1944
maxConflicts          : 50
Constraints           : 6746
LearntConstraints     : 35
FreeVars              : 3880
Vars/FreeVars         : 1.50309
FreeVars/Constraints  : 0.575156
Vars/Constraints      : 0.864512
maxLearnt/Constraints : 0.288171
====================
Iteration             : 2
maxLearnt             : 1944
maxConflicts          : 50
Constraints           : 6652
LearntConstraints     : 70
FreeVars              : 3819
Vars/FreeVars         : 1.5271
FreeVars/Constraints  : 0.574113
Vars/Constraints      : 0.876729
maxLearnt/Constraints : 0.292243
====================
Iteration             : 3
maxLearnt             : 1944
maxConflicts          : 50
Constraints           : 6670
LearntConstraints     : 102
FreeVars              : 3819
Vars/FreeVars         : 1.5271
FreeVars/Constraints  : 0.572564
Vars/Constraints      : 0.874363
maxLearnt/Constraints : 0.291454
====================
Completed             : No

Atoms                 : 4420   (Original: 4420 Auxiliary: 0)
Rules                 : 5424   (BasicR: 3340, ConstraintR: 2075, ChoiceR: 9, WeightR: 0)
NormalRules/ExtRules  : 1.60886
Bodies                : 5252
Equivalences          : 6812   (Atom=Atom: 202 Body=Body: 40 Other: 6570)
Tight                 : Yes
Variables             : 5832   (Eliminated:    0)
Constraints           : 7238   (Binary: 55.0566% Ternary: 20.42% Other: 24.5233%)

Models                : 0
Choices               : 615
Conflicts             : 150
Restarts              : 3
Constraints deleted   : 0
Backtracks            : 0
Backjumps             : 150 ( Bounded: 0 )
Skippable Levels      : 586
```

```
62  Levels skipped        : 586 (100%)
63  Max Jump Length       : 122 ( Executed: 122 )
64  Max Bound Length      : 0
65  Average Jump Length   : 3.90667 ( Executed: 3.90667 )
66  Average Bound Length  : 0
67  Average Model Length  : 0
68  Lemmas                : 150    (Binary:   18% Ternary:   12% Other:   70%)
69    Conflicts           : 150    (Average Length: 10.2)
70    Loops               : 0      (Average Length: 0)

72  Time        : 0.150  (Solving: 0.130)
```

Observe that some dynamic features are printed three times in Line 10–42 (one block per restart), while static features and dynamic feature summaries are provided only once in Line 43–70. Also note that, in case `claspre` finds some model upon feature extraction, it is output before the features, unless suppressed via option `-q`.

A compressed feature format can be obtained by using option `--claspfolio` instead of `--features`. The admissible values for this option are as follows:

`--claspfolio 1`
: Print a separate line with dynamic features' values on each restart, followed by a line with static features' values and dynamic feature summaries.

`--claspfolio 2`
: Print one line with dynamic features' values (including all restarts), followed by a line with static features' values and dynamic feature summaries.

`--claspfolio 3`
: Like `--claspfolio 2`, but if some model (or unsatisfiability) is found upon feature extraction, it is printed instead of the features.

The `--claspfolio` option is provided for obtaining an easily machine-readable output, as it is used by portfolio-solver `claspfolio`, available at [6].

An example invocation of `claspre` using option `--claspfolio 1` along with corresponding output is given next:

```
gringo blocks.lp world4.lp --ifixed 9 | \
claspre --noLP --claspfolio 1 --restarts 50,1 --endR 3
```

```
1  1944,50,6746,35,3880,1.50309,0.575156,0.864512,0.288171
2  1944,50,6652,70,3819,1.5271,0.574113,0.876729,0.292243
3  1944,50,6670,102,3819,1.5271,0.572564,0.874363,0.291454
4  No,4420,4420,0,5424,3340,2075,9,0,1.60886,5252,6812,202,40,6570,Yes,NA,NA,
   5832,0,7238,55.0566,20.42,24.5233,0,615,150,3,0,0,150,0,586,586,100,
   122,122,0,3.90667,3.90667,0,0,150,18,12,70,150,10.2,0,0
```

Similar output is obtained with either:

```
gringo blocks.lp world4.lp --ifixed 9 | \
claspre --noLP --claspfolio 2 --restarts 50,1 --endR 3
```

or:

```
gringo blocks.lp world4.lp --ifixed 9 | \
claspre --noLP --claspfolio 3 --restarts 50,1 --endR 3
```

The difference to `--claspfolio 1` is that iterated dynamic information in Line 1–3 is output on one line. Finally, `--claspfolio 3` outputs models (or UNSATISFIABLE) instead of features if found out during pre-processing:

```
gringo blocks.lp world4.lp --ifixed 9 | \
claspre --noLP --claspfolio 3 --restarts 50,1 --endR 4
```

```
1  Answer: 1
2  move(b10,b2,1) move(b9,table,2) move(b4,b9,3) move(b8,b3,4) move(b7,b8,5)
   move(b10,b6,6) move(b2,b10,7)   move(b1,b2,8) move(b0,b4,9)
```

The names of features output by `claspre` with option `--claspfolio` can be obtained via option `--listFeatures`. This is achieved via the following call:

```
claspre --listFeatures
```

The corresponding output is:

```
1  maxLearnt,maxConflicts,Constraints,LearntConstraints,FreeVars,
   Vars/FreeVars,FreeVars/Constraints,Vars/Constraints,maxLearnt/Constraints
2  Completed,Atoms,_Original,_Auxiliary,Rules,_BasicRule,_ConstraintRule,
   _ChoiceRule,_WeightRule,NormalRules/ExtRules,Bodies,Equivalences,
   _Atom=Atom,_Body=Body,_Other,Tight,_SCCs,_Nodes,Variables,_Eliminated,
   Constraints,_Binary,_Ternary,_Other,Models,Choices,Conflicts,Restarts,
   Constraints deleted,Backtracks,Backjumps,_Bounded,Skippable Levels,
   Levels skipped,_%,Max Jump Length,_Executed,Max Bound Length,
   Average Jump Length,_Executed,Average Bound Length,Average Model Length,
   Lemmas,_Binary,_Ternary,_Other,Conflicts,_Average Length,Loops,_Average Length
```

It indicates names as used in output obtained with option `--features` of iterated dynamic features (in Line 1) and of residual features (in Line 2). The values printed with `--claspfolio` correspond to the listed feature names in the same order.

Finally, we note that `claspre` can also be run as an ASP solver by providing option `--noLP`, but neither `--features` nor `--claspfolio`. As with feature extraction, all options inherited from `clasp` can be used to customize solver behavior. In addition, `--endC` and `--endR` can be set to force termination after a certain number of conflicts or restarts, respectively.

2 Summary of `claspre` Options

This section gives a quick overview of command-line options provided by `claspre` to configure the pre-processing functionalities described in Section 1. Beyond these, `claspre` inherits many command-line options from `clasp` (cf. [2]), allowing for the customization of pre-processing. For instance, `--trans-ext` may be configured to compile extended rules into normal ones.

By default, `claspre` applies static pre-processing techniques [3] of `clasp` to obtain a compact representation of an input logic program in Smodels Internal Format [8]. Further options of `claspre` are listed below:

`--noLP`
: Do not print pre-processed logic program and instead run `claspre` as a solver.

`--endC` *`n`*
: Stop solving after encountering *n* conflicts, *n* = 0 standing for no limit on conflicts.

`--endR` *`n`*
: Stop solving after performing *n* restarts, *n* = 0 standing for no limit on restarts.

`--features`
: Print static and dynamic features, the latter obtained upon solving. When using `--features`, by default, `--endC 500` and `--endR 20` are applied as limits on conflicts and restarts, respectively.

`--claspfolio 1|2|3`
: Print static and dynamic features in a compressed, easily machine-readable format. When using `--claspfolio`, by default, `--endC 500` and `--endR 20` are applied as limits on conflicts and restarts, respectively.

`--listFeatures`
: Print a list of feature names. The feature names are the same as the ones displayed with option `--features`. The list format is the same as the one of `--claspfolio 1` (when performing exactly one restart), but showing feature names instead of their values.

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 [6]. 1, 3, 4, 7

[3] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. Advanced preprocessing for answer set solving. In M. Ghallab, C. Spyropoulos, N. Fakotakis, and N. Avouris, editors, *Proceedings of the Eighteenth European Conference on Artificial Intelligence (ECAI'08)*, pages 15–19. IOS Press, 2008. 1, 3, 7

[4] 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

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

[6] Potsdam answer set solving collection. University of Potsdam. `http://potassco.sourceforge.net/`. 1, 3, 5, 8

[7] P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234, 2002. 3

[8] T. Syrjänen. Lparse 1.0 user's manual. `http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz`. 1, 3, 7