PLRSF V0.9

Éric Würbel

Table des matières

1 Introduction

PLRSF is an application which implements Removed Set Merging of logic programs with answer set semantics [1, 2].

Removed set merging of logic programs consists in the retraction of rules from logic programs in order to restore consistency.

Two flavours of consistency of logic programs are considered : in a first definition, a program is considered as consistent iff it has at least one answer set. In a second definition, a program is considered consistent iff it has at least one SE-model.

The problem addressed by PLRSF is as follows : considering a profile $\{P_1, ..., P_n\}$ of logic programs (i.e. a multi-ensemble of logic programs), perform the merging of these programs. If the union of the programs is consistent, then the merging is a simple union, otherwise PLRSF removes some rules from som profiles according to different strategies in order to restore consistency. For details about the strategies, see [1, 2].

This manual is only a preliminary version.

2 Installation

PLRSF is mainly a prolog program written for the swi-prolog compiler in mind. It should not be complicated to port PLRSF on other prolog systems. If you consider writing a port, please drop me a line.

A this time, the instruction installations are directed toward swi-prolog installed on a linux system.

plrsf consist in two different applications :

— `plrsf` : a command line tool which computes the result of the merging of a profile consisting in a set of files.

— `plrsf-webdemo` : a web-based version of PLRSF, suitable as a demo.

2.1 Building the executables

The script file `build-executable.sh` is provided to build both programs `plrsf` and `plrsf-webdemo`. It accepts one option `-e`, which effect is to incorporate swi-prolog runtime libraries in the final executable. This is actually needed for 64 bits linux systems. YOu can omit this build option on 32 bit inux systems, except is you plan to copy the final executables on a machine on which swi-prolog is not installed.

2.2 Installing the executables

At this time the is no installation script. All you have to do is to copy the executable files `plrsf` and `plrsf-webdemo` and the starting script `run-plrsf-webdemo` in directory appearing in your `PATH`.

2.3 Configuring the installation

For the `plrsf` command line program, there is nothing to configure.

The web demonstrator need a little more work to be fully operational.

First of all edit the script `run-plrsf-webdemo` and adjust the `PLRSF_EXE` variable to point on the path of the `plrsf-webdemo` executable. Adjust the `SWI_HOME_DIR` to point to the directory containing swi-prolog runtime libraries.

The next step depends on your installation choice :

— If you only want to perform the demo on the computer where you installed it, look at the section **configuring webdemo**.

— If you want to configure the demo so it may be accessible on the WEB, please perform the configuration as indicated in the next section (yet to come)

2.4 Configuring the webdemo for public access

2.4.1 configuring the webdemo

Wether you want to use the web demo locally or run it publicly, you have to perform this configuration step. Edit the source file `webdemo.pl` and modify the following line :

```
user:file_search_path(document_root,'/home/wurbel/src/prolog/plrsf').
```

Modify the path so it points to the plrsf installation directory. Recompile `plrsf`.

From now on, you can use the webdemo locally. If you want to configure a public access for the webdemo, please go through the following sections.

2.4.2 Using apache reverse proxy setup

2.4.3 Using apache rewrite engine and reverse proxy

2.4.4 Using apache VirtualHosts

This is prefferably the best solution, but it needs administrative access.

3 Running plrsf

3.1 running the command line tool

The command line tool syntax is as follows (you can print this help by invoking `plrsf --help` at the shell prompt) :

```
plrsf [options] files...
--clasp-path      -c  term=path(clingo)  clasp path. Accepted values are either a pathname, re
                                           or absolute, or a term of the form path(exe), where
                                           is the name of the executable. The executable is th
                                           searched among the directories specified in the PAT
                                           environment variable.
--clasp-ver       -v  integer=4          clasp version. Accepted values are 3 and 4. Default i
                                           meaning that the version of clasp is 4 or up. Aggre
                                           litterals and choice constructs have a different sy
                                           depending on the version number.
--output          -o  atom=user_output   output destination. This can be either a file name or
                                           user_output, which stands for standard output.
--mode            -m  atom=strong        Potential reoved sets mode:
                                             weak: weak potential removed sets (based on SE mo
                                             strong: strong potendtial removed sets (based on
--strategy        -s  atom=sigma         merging strategy, one of
                                             sigma:   sigma strategy
                                             card:    card strategy
                                             max:     max strategy
                                             gmax:    gmax strategy
                                             inclmin: inclusion-minimal potential removed sets
                                             all:     all potential removed sets
--results         -r  atom=all           requested results, one of
                                             all:    all belief bases resulting from the
                                                     merging
                                             arsets: only print the atoms characterizing the r
                                                     (debugging purpose mainly)
                                             rsets:  all removed sets.
--program-output  -p  atom=none          ASP program output, one of
                                             none:        program is temporary.
                                             user_output: ASP program is written on standard o
                                             ATOM:        ASP program is written to the file w
                                                          name is ATOM.
```

The files contain the belief profile and the integrity constraints. Each file contain a belief base, it must start with a fact `kbname/1` asserting the belief base name.

The file containing the integrity constraints must begin with the fact `kbname(ic)`.

For example, suppose we want to run the `archeo` example which is in the `test` directory. This example contain 3 belief bases and a set of integrity constraints. To perform the merging using the strategy Σ and showing all the possible resulting belief bases, using weak merging, the command is :

```
plrsf -sigma -weak test/archeo-*.pl
```

3.2 Running the web demo

3.2.1 running the web demo locally

To run the web demonstrator on your machine, launch the `run-plrsf-webdemo` script. Then, open a web browser, and browse the following url : `http://localhost:5000/plrsfdemo/`

The demo is self-documented.

Références

[1] Julien Hué, Odile Papini, and Eric Würbel. Merging belief bases represented by logic programs. In Claudio Sossai and Gaetano Chemello, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 10th European Conference, ECSQARU 2009, Verona, Italy, July 1-3, 2009. Proceedings*, volume 5590 of *Lecture Notes in Computer Science*, pages 371–382. Springer, 2009.

[2] Julien Hué, Odile Papipni, and Eric Würbel. Extending belief base change to logic programs with asp. In *Trends in Belief Revision and Argumentation Dynamics*, Studies in Logic. 3 december 2013.