PLRSF V0.99

Éric Würbel

jeudi 28 mai 2015

Table des matières

1 Introduction **1**

2 Installation **2**

2.1 Configuring the installation 2
2.2 Building the executables 2
2.3 Installing the executables 2
2.4 Configuring the webdemo for public access 2
2.4.1 Using apache reverse proxy setup 3
2.4.2 Using apache rewrite engine and reverse proxy 3
2.4.3 Using apache VirtualHosts 3

3 Running plrsf **3**

3.1 running the command line tool 3
3.2 Running the web demo 4
3.2.1 Running the web demo locally 4
3.2.2 Running the web demo through apache 4

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 withdrawal 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 HT-model [3].

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].

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, at least the command line version of PLRSF [1]. If you consider writing a port, please drop me a line.

At present, the installation instructions 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 Configuring the installation

Concerning 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. Then, 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.

2.2 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.

2.3 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 a directory appearing in your `PATH`.

2.4 Configuring the webdemo for public access

When configured and built, the webdemo listens for http requests on port 5000 of your local machine. However, a public access to the PLRSF webdemo requires requests being sent on port 80 (standard web TCP port). It's a bad idea to modify the PLRSF webdemo to make it listen on port 80, as on server machines this port is generally handled by a web server software like apache.

The following sections describe three different methods enabling an apache server to relay incoming http requests on port 80 to port 5000, and to relay back the responses of the PLRSF service.

1. The web demo will be trickier to port on other prolog systems, not to say almost impossible, as it relies on the http client and server support provided by a swi-prolog module.

2.4.1 Using apache reverse proxy setup

This is the simplest way to configure PLRSF for public web access. This approach uses apache reverse proxy mechanism. Ensure the modules `proxy` and `proxy_http` are loaded. Then add two simple rules to the server configuration. Below is an example that makes a `plrsfdemo` server on port 5000 (which are the default values coded in the source file) available from the main Apache server at port 80.

```
ProxyPass        /pldoc/ http://localhost:4000/pldoc/
ProxyPassReverse /pldoc/ http://localhost:4000/pldoc/
```

2.4.2 Using apache rewrite engine and reverse proxy

2.4.3 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, relative or absolute, specified as a term file(pathname), or a term of the form path(exe), where exe is the name of the executable. The executable is then searched among the directories specified in the PATH environment variable.

`--clasp-ver -v integer=4` clasp version. Accepted values are 3 and 4. Default is 4, meaning that the version of clasp is 4 or up. Aggregate litterals and choice constructs have a different syntax depending on the version number.

`--output -o atom=user_output` output destination. This can be either a file name or the atom $\text{user}_{\text{output}}$, which stands for standard output.

`--mode -m atom=strong` Potential reoved sets mode : weak : weak potential removed sets (based on SE models) strong : strong potendtial removed sets (based on answer sets)

`--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 removed sets. (debugging purpose mainly) rsets : all removed sets.

`--program-output -p atom=none` ASP program output, one of none : program is temporary. $\text{user}_{\text{output}}$: ASP program is written on standard output. ATOM : ASP program is written to the file whose 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.

3.2.2 Running the web demo through apache

Providing that you configured and compiled the web demo executable properly, and that your apache web server has been configured to relay the requests addressed to the plrsf service to the plrsf executable, using one of the three described techniques, running the web demonstrator boils down to :

— running the `run-plrsf-webdemo` script. This script runs the `plrsf-webdemo` daemon. It can be safely detached from the terminal ;

— running apache.

The full automation of the starting of the service is out of the scope of this documentation at the moment. It requires the creation of a starting script for the *init* service or equivalent (*upstart*, *Ruinit*, etc.).

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.

[3] David Pearce. A new logical characterisation of stable models and answer sets. In Jürgen Dix, Luís Moniz Pereira, and Teodor C. Przymusinski, editors, *Non-Monotonic Extensions of Logic Programming, NMELP '96, Bad Honnef, Germany, September 5-6, 1996, Selected Papers*, volume 1216 of *Lecture Notes in Computer Science*, pages 57–70. Springer, 1996.