Guidelines to the Current Implementation of the MAP Transformation System

Sophie Renault

IASI-CNR, Viale Manzoni 30, 00185 Roma, Italy
e-mail: renault@iasi.rm.cnr.it

Abstract

In this note, we give the instructions to get, install and use the current implementation of the MAP transformation system. Since the system is still under development, the note is likely to become obsolete soon, but it should indicate where to get an updated version of it. A report presenting the design, implementation and use of the system will be available soon.

1 Introduction

The MAP transformation system is a tool which is designed to support the interactive transformation of logic programs using the unfold/fold methodology [4]. It has been developed at IASI Institute of the National Research Council, Rome, Italy, on the basis of an existing implementation presented in [5]. The main novelty in the new implementation is a graphical user interface (GUI) through which transformations are specified and applied by means of mouse clicks, button presses, menu selections, dialog boxes etc. The core of the system, i.e. the transformation engine, is written in SICStus Prolog [6] and the GUI is written in Tcl/Tk [3]. When a transformation rule is invoked, if the conditions for its application the system transforms the current program into a new one. Several such transformation steps may be performed interactively, until the desired program is derived. Some strategies are also available, which perform in a fully automatical way a sequence of transformation steps. MAP provides the user with many facilities such as backtracking, navigating, printing, saving and restoring a derivation.

2 Location and installation of the system

The system can be found on `adpu.iasi.rm.cnr.it` in `/home/save_map/`. Take the file named "newmap.tar.gz" and put it in the directory in which you want to install it. Then uncompress and unpack it as usual:

```
> gunzip newmap.tar.gz
> tar xvf newmap.tar
```

This will create a directory `MAP/` containing subdirectories and files as described now.

3 Directory structure

The directory `Doc/` contains the file `guidelines.ps` which is the last updated version of this note.

The code of the system is split into the directories `src_tcl/` and `src_sics/`. `src_tcl/` contains the tcl/tk code, that is the code corresponding to the GUI. `src_sics/` contains the SICStus prolog code. The directory `src_tcl/` contains the following files:

- `mapMain_c.tcl`: global variables and directives for the compilation of the other files contained in the same directory.

- `control_c.tcl`: control of the main widgets of the sytem.
- `display.tcl`: layout of the system.
- `rules.tcl`: calls to the transformation rules.
- `help_rules.tcl`: on-line help.
- `basics.tcl`: low level procedures used by the other files.

The directory `src_sics/` contains the following files:

- `srcMain.pl`: directives for the compilation of the other files contained in the same directory.
- `general_calls.pl`: control calls to the various transformation rules and updating of the derivation tree.
- `environment.pl`: predicates which update the derivation tree.
- `definition_rule.pl`: predicates for the rules "define" and "restrict".
- `unfolding_rule.pl`: predicates for the rule "unfold".
- `folding_rule.pl`: predicates for the rule "fold".
- `goal_replacement.pl`: predicates for the rules "replace goal", "add goal", "delete goal".
- `case_split.pl`: predicates for the rule "case split".
- `generalize.pl`: predicates for the rule "generalize".
- `clause_replacement.pl`: predicates for the rules "add_clauses", "delete_clauses", "replace_clauses", "move_clause".

The other directories are `Histories/`, `Programs/`, `Sessions/` and `Theories/`.

- the directory `Programs/` contains the user programs intended to be transformed, that is, those programs which can be selected to begin a transformation sequence. The menu proposed to the user when he issues the entry "New" in the menu "Derivation" is made out of the files available in `Programs/`.
- the directory `Theories/` contains *theories*, that is, sets of laws which are intended to be used for the replacement rule. In order to make a law of a theory available for the replacement rule, the associated theory must be first selected. This can be done by issuing the entry "Load" in the menu "Theory". Laws are rewrite rules using the following syntax: $A_1, \ldots, A_n \rightarrow B_0, \ldots, B_m$, where $A_1, \ldots, A_n, B_0, \ldots, B_m$ are literals.
- the directory `Histories/` contains files associated with the history of a derivation that have been saved by issuing the entry "History" in the menu "Save" (or the entry "Save/History" in the menu "Derivation"). These files are text files providing a description of the corresponding saved derivation.
- the directory `Sessions/` contains files associated with a derivation tree that has been saved by issuing the entry "Derivation" in the menu "Save" (or the entry "Save/Derivation" in the menu "Derivation"). These files are intended to be restored by issuing the enrty "Load" in the menu "Derivation" and are not supposed to be modified by the user.

4 Running the system

The current implementation of the system has been done in SISCtus prolog 3 #6 and Tcl/Tk 8.0. It uses the SICStus prolog interface to Tcl/Tk. An on-line manual of Tcl/Tk can be found at the URL: `http://www.beedub.com/book/2nd/booktoc.mif.html`.

To run the system, you must first define the variable "NEWMAP_DIR" to indicate where the "MAP" directory is. For instance, at the unix prompt, I would issue the command:

```
> setenv NEWMAP_DIR /home2/users/renault/PROLOG/MAP
```

because the MAP directory is installed in my home directory under PROLOG/.

Then in the MAP directory, run a SICStus prolog session and at the prolog prompt issue the following goal:

```
| ?- [map_c].
```

This will compile the source code of the system. After the compilation, you may want to save the compiled state in order to avoid to recompile the code each time you want to use the sytem. You can do it as follows:

```
| ?- save(map).
```

This will save the compiled state of the system into file "map", and therefore can be restored directly at the unix level by issuing the command "map". Note that saved states might be uncompatible from a machine to another, and therefore you may want to have several saved state with different names for the different machines on which you intend to use the system. For instance "map_adpu" for the system compiled (and saved) on adpu, and "map_merlino" for the system compiled (and saved) on merlino.

The top level goal to be issued at the SICStus prolog level is then

```
| ?- map.
```

This will open a window with various buttons and menus. To start a derivation, issue the entry "New" in the menu "Derivation". This will open a dialog box with the list of all files in the directory "Programs". To load an initial program, double click on it, and press the "Load" button on the dialog box. The initial program will appear on the screen. Press "Theory" to load a set of laws to be used by the "replace goal" rule. Press "Rules" to have the menu of all available transformation rules. Press "Help" to enter the on-line help.

References

[1] R. M. Burstall and J. Darlington. A transformation system for developing recursive programs. Journal of the ACM, 24(1):44–67, January 1977.

[2] J. W. Lloyd. Foundations of Logic Programs. Springer Verlag, Second edition, 1987.

[3] John Ousterhout. Tcl and the Tk Toolkit. Addison-Wesley, 1994.

[4] A. Pettorossi and M. Proietti. Transformation of logic programs: Foundations and techniques. Journal of Logic Programming, 19,20:261–320, 1994.

[5] S. Renault. A System for Transforming Logic Programs. RR-97.04, Department of Computer Science, University of Rome Tor Vergata, 1997.

[6] M. Carlsson and J. Widen. SICStus Prolog user's manual. Research report 88007B. Sweedish Institute of Computer Science, Kista, Sweeden, 1988.

[7] H. Tamaki and T. Sato. Unfold/fold transformation of logic programs. In S.-Å. Tärnlund, editor, Proceedings of the Second International Conference on Logic Programming, Uppsala, Sweden, pages 127–138. Uppsala University, 1984.