SLs: User's Guide

(Still being written)

Irène Durand

Université de Bordeaux I
33405 Talence, France
idurand@labri.fr

1 Introduction

SLS is a tool for handling Slanguages (containing swords) and Sexpressions (expressions over Slanguages). [4, 3, 1, 2]

The graphical interface (still under construction) is written using FreeCLIM, the free implementation of the CLIM specification. New functionnalities could be easily integrated upon demand.

2 Installation

The system has been compiled to run on a PC x86 under Linux and also on a Power PC under MacOSX. Other architectures can be made available on demand.

The graphical interface is an X11 client so requires an X11 server.

The user should get the file SLS.tgz appropriate for your system and install the system with the following commands.

```
$ tar xzvf SLS.tgz     # unless done automatically by your system
$ cd SLS
$ ./INSTALL
```

The system is started by typing

```
./sls &
```

Use the `Quit` button or the `Quit` command to exit SLS.

The directory `SLS` contains a `Data` directory were the specifications are stored.

3 Specification Files

The user may specify a set of SLS objects (swords, slanguages, sexpressions) related to a common alphabet of letters in a specification file whose name should have the extension `.txt`.

An SLS specification file starts with the definition of a signature, eventually followed by th e definition of a set of variables. Next we may have in any order definitions of TRSs, automata, termsets, each one of them associated with a distinct name.

Example: `trains.txt`

```
Problem trains
 ([{a,b,e}] . ([{a}] X [{b}] X [{e}]))
 (([{a}] X [{b}]) . [{a}{b}])
 ((([{a}] X [{c}]) . [{a}{c}{d}{d}]) U (([{a}] X [{c}]) . [{a,c}{d}{d}]))
 ([{b}{b,d,f}] . ([{f}] X [{d}]))
 (([{e}] X [{d}]) . [{d,e}])

Sword [{a}{{a}]
Lexpr ([{a}{a}] X [{b}{b}])
Lexpr ([{a,b}] J [{b_1,c}])
```

3.1 Comments

Any part of a line located after the character `;` is considered a comment and will be ignored.

3.2 Names

The name of a letter or a language should not contain the following characters `;`, `:`, `(`, `)`, `"`, `-` unless the name is surrounded by `" "`.

3.3 Letters, iletters sletters

A *letter* has a *name*. An *iletter* is a letter indexed with an integer. A *sletter* is a set of iletters.

3.4 Swords

An *Sword* is introduced by the `Sword` keyword followed by the Sword. Swords are sequences of Sletters surround by squared brackets ([]).

3.5 Lexpr

An *Lexpr* is introduced by the `Lexpr` keyword followed the Lexpr surrounded by parentheses.

3.6 Problem

An *Problem* is introduced by the `Problem` keyword followed the name of the problem followed by a list of Lexprs which are the constraints of the problem.

3.7 Slanguage

A *Slanguage* is introduced by the `Slanguage` keyword followed by the name of the Slanguage and its swords.

4 Using SLS

4.1 Generalities

The top pane of the SLS window is an interactor pane from which the user interacts with the system. It prompts the user for commands and arguments.

At any time the user can get **help** about the current possibilities by clicking on the right-button.

Commands are accessible either thru the command line using **completion** (achieved with the `<TAB>` key or from the menus items. Some commands are also available from buttons.

All commands accessible are accessible by typing the command name (written on the item) to the *Command* prompt. The latter is recommended as thru completion it will save the user a lot of typing. Completion works for commands, data filenames, automaton names, trs names, termset names.

The menu are there to remind the user of every possible command or for people who don't like to type. In any case, at the moment, arguments of commands need to be typed in the interactor pane.

- The *Read* commands are used to read SLS objects into the current specificationc directly from the command line.
- The *Load* commands are used to load SLS objects into the current specification from a file with the `.txt` extension.
- The *Save* commands are used to save SLS objects into a file.
- The *Retrieve* commands are used to set the current SLS object with an already computed object.
- The *Alphabet* (resp. *Ialphabet*) commands give the alphabet (resp. ialphabet) of the current SLS object of the correponding type.

4.2 File Menu

From this menu, one can change the directory where specifiations are loaded *Change Data Directory* , clear the bottom result pane *clear* or quit the application *Quit* .

4.3 Spec Menu

This menu gathers all commands related to the current Specification.

4.4 Sword Menu

This menu gathers all commands related to the current Sword..

Size One can obtain the length of the current Sword by using the *Size* command.

4.5 Lexpr

This menu gathers all commands related to the current Lexpr.

Language

4.6 Language

This menu gathers all commands related to the current Slanguage

Cardinal One can obtain the cardinal of the current Slanguage by using the *Cardinal* command.

5 How to exit if SLS crashes

In that case, you should end up in the Lisp debugger. Just type *(cl::quit)* to exit Lisp.

6 Example of a session

Command: ***Load Spec***
spec filename: ***trains.txt***
Command: ***Solve***
Command: ***Language Lexpr***
Command: ***Cardinal Language***
Command: ***Membership To Language***

7 Snapshots of a session

Figures 1, 2, 3, 4and 5 at the end of the document show some snapshots of the graphical interface.

References

1. Jean-Michel Autebert, Matthieu Latapy, and Sylviane R. Schwer. Le treillis des chemins de delannoy. *Discrete Mathematics*, 258(1-3):225–234, 2002.
2. Jean-Michel Autebert and Sylviane R. Schwer. On generalized delannoy paths. *SIAM J. Discrete Math.*, 16(2):208–223, 2003.
3. Sylviane R. Schwer. Temporal granularity enlightened by knowledge. In *NLDB*, pages 30–41, 2000.
4. Sylviane R. Schwer. Reasoning with intervals on granules. *J. UCS*, 8(8):793–807, 2002.

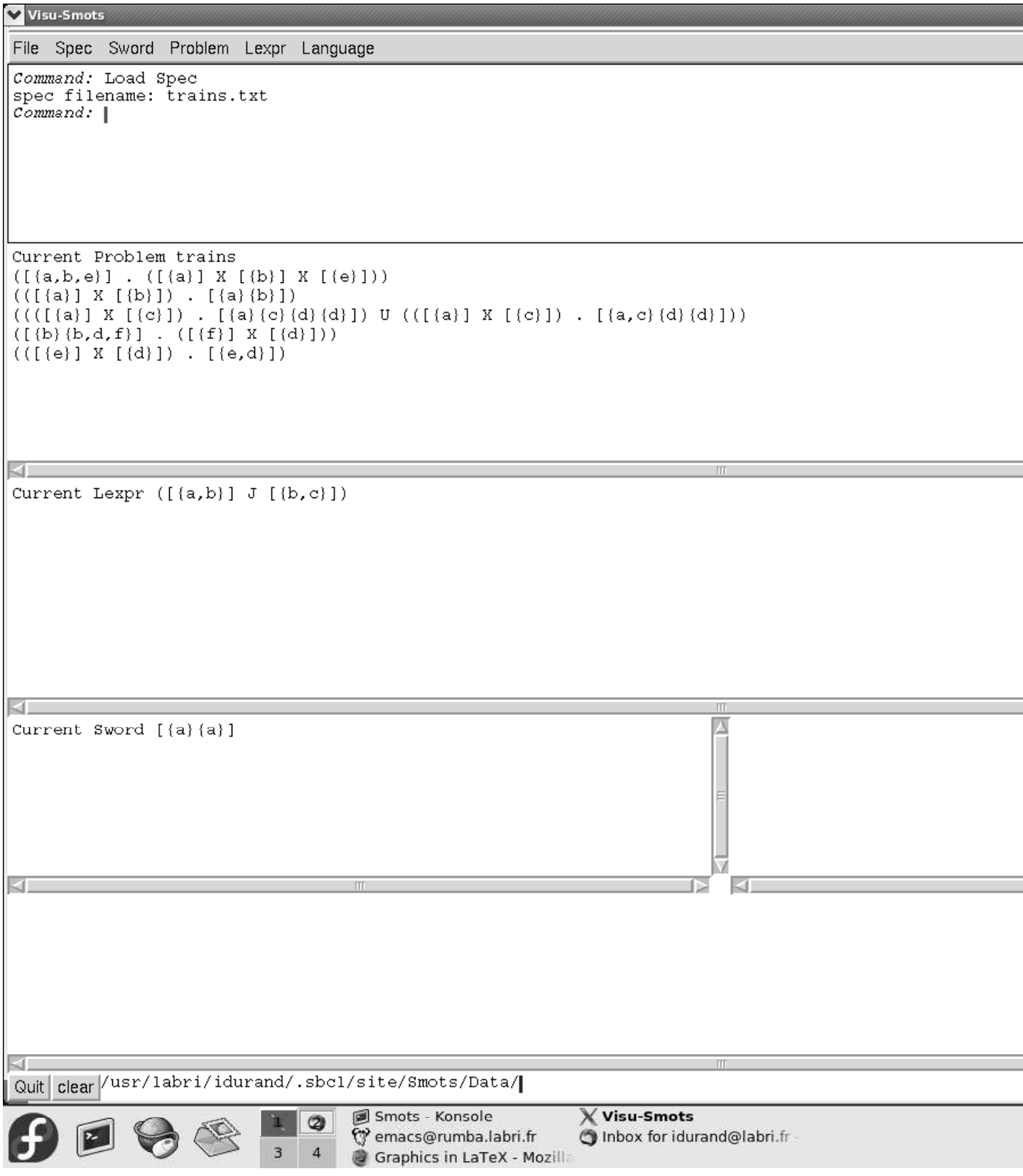

Fig. 1. Operations on the current term and the current automaton

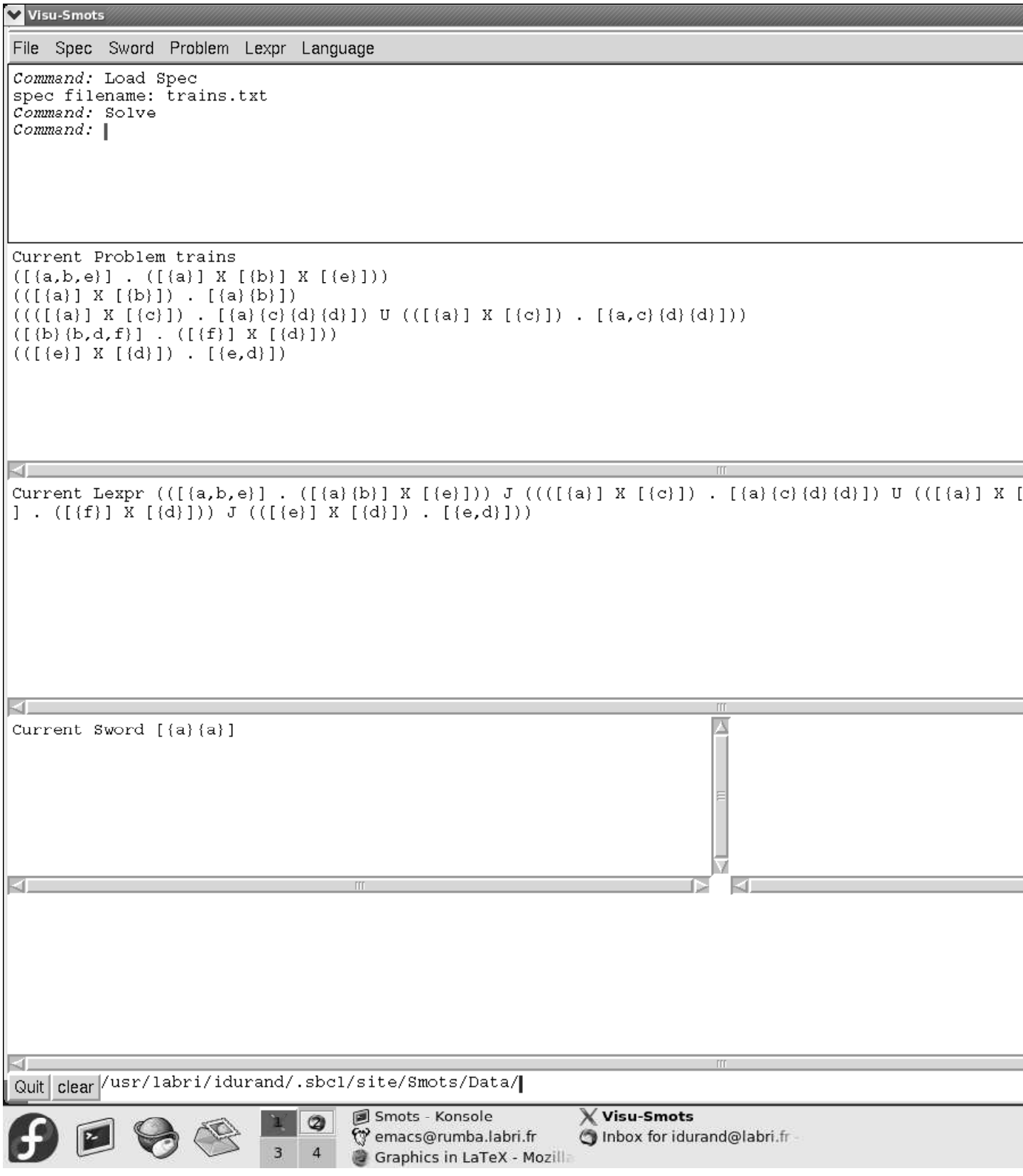

Fig. 2. Boolean operations on automata

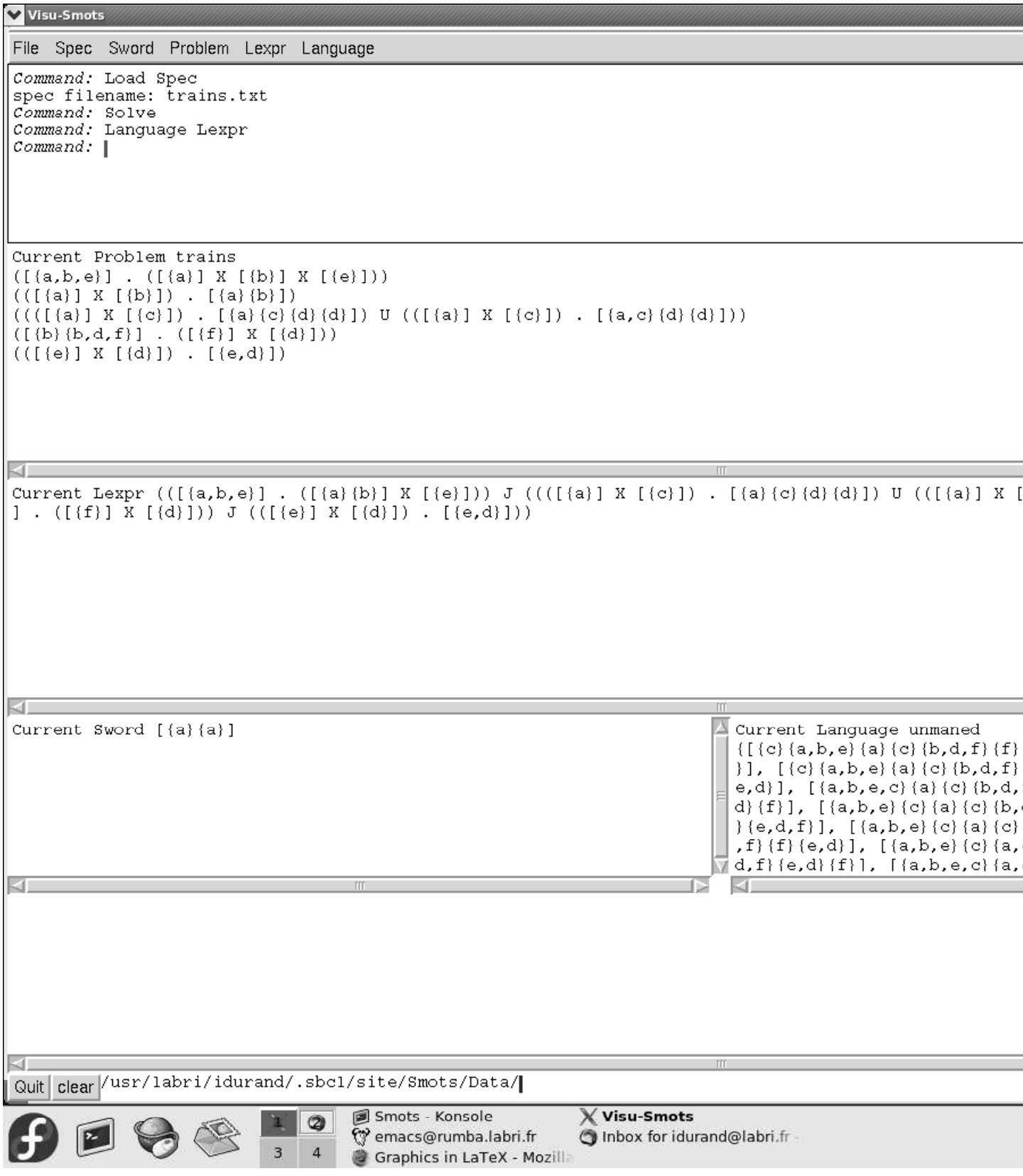

Fig. 3. Call by need queries

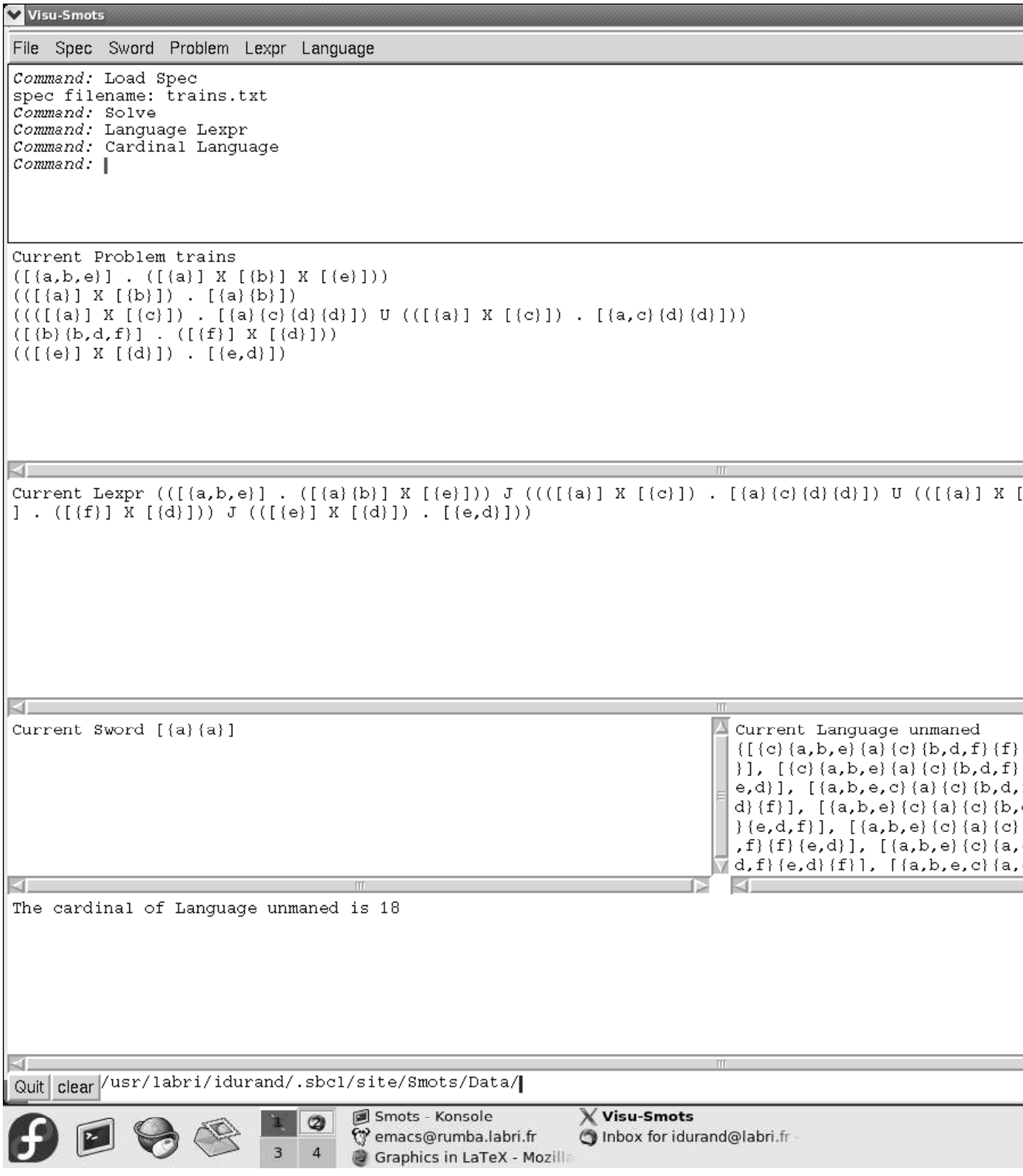

Fig. 4. Call by need queries

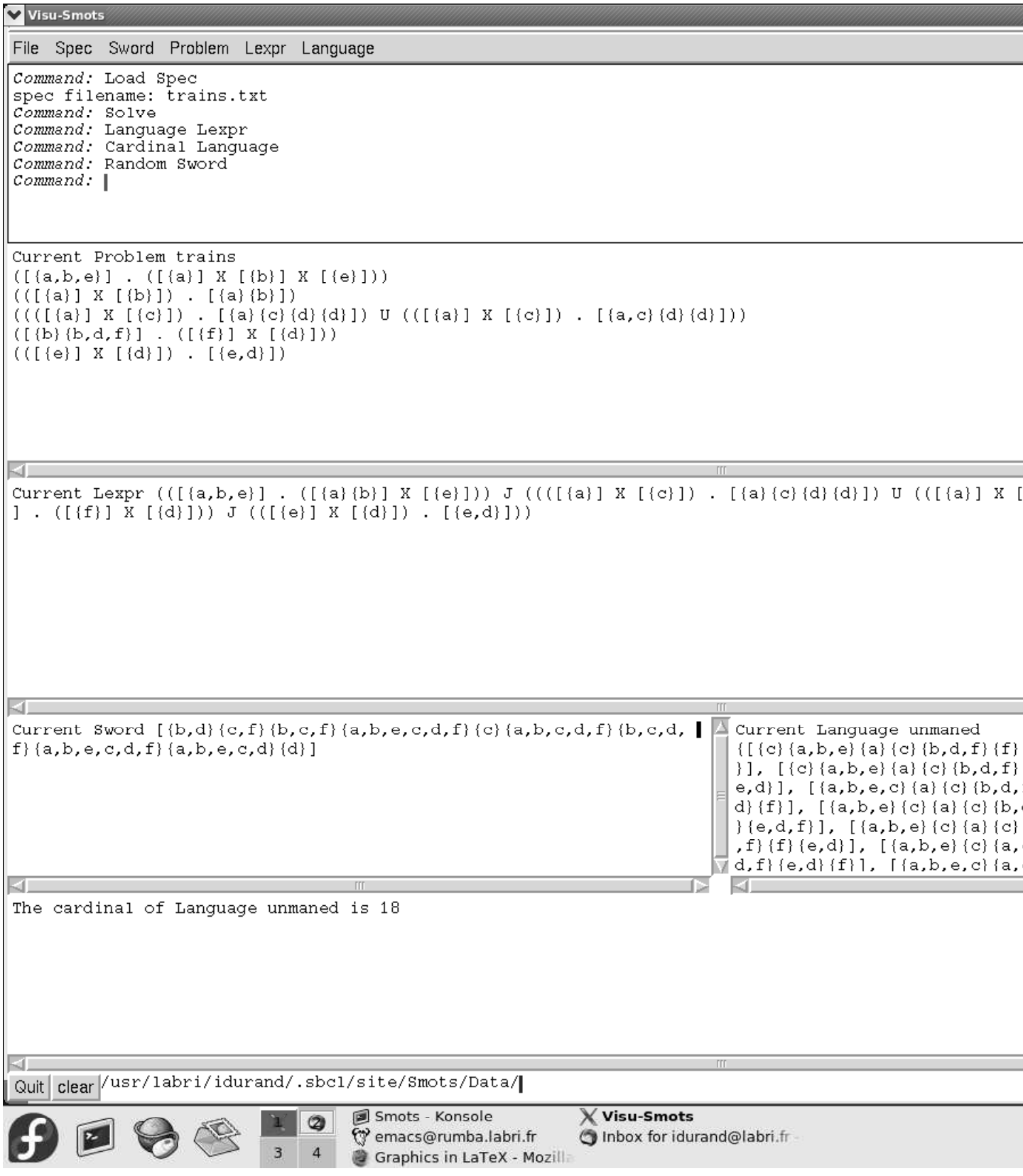

Fig. 5. Call by need queries