Trale

1 Links

Here you can learn how to install Trale and test its functionality on a simple grammar. To learn more about Trale and Trale grammars you should consult the Trale manual: `http://utkl.ff.cuni.cz/~rosen/public/trale-manual.pdf`. Or see the Trale download web page with more links, including more detailed documentation at `http://milca.sfs.uni-tuebingen.de/A4/Course/trale/`.

The textbook `http://milca.sfs.uni-tuebingen.de/A4/Course/PDF/gramandpars.pdf` provides a good introduction to grammar writing in Trale.

2 How to install Trale

There are three ways, the easiest way (§ 2.1 — no installation) can be used with any Intel-based machine that allows system booting from CD.

2.1 How to install Trale on a Windows system, or how to avoid installing it on any system

If you have Sicstus Prolog and Cygwin, you can try, but you are largely on your own, see `http://milca.sfs.uni-tuebingen.de/A4/Course/trale/` and the help files included with the distributions. Note that only the new Trale package handles UTF-8 characters.

The much easier way (at least to install) is to use the bootable CD "Grammix". This CD persuades your PC to run Knoppix, a Linux system. To use it, some experience with Linux is a plus. You can do it even if you have Linux on your PC, but then you might prefer to save yourself the trouble of rebooting, using a different system and fiddling with a memory stick, at the cost of spending a while installing Trale on your system (see § 2.3).

You can download the ISO image of the CD from `http://hpsg.fu-berlin.de/Software/Grammix/` and burn it, or you can ask your kind teacher :-), who has a few copies available. Note that while using this CD, you should have a flash memory stick to save your files and your PC must allow for CD system booting (this is a BIOS option). A few hints on how to use this CD:

1. Make sure your PC allows booting a system from CD. To enter the BIOS setup screen, you should press a key while the PC is starting up, before any system is loaded. This key is often F12. In the BIOS setup, find an item specifying the order of system boot devices and make sure the highest priority is assigned to CD. Save the new BIOS setup.
2. Insert the Grammix CD into your CD drive. Let your PC boot from the CD. Restart your PC, if you have a system running.
3. After a while you should see a Knoppix intro screen in German. At the
   ```
   boot:
   ```
   prompt below type
   ```
   knoppix lang=us
   ```
   and press Enter.
4. After another while you should see the HPSG grammar development desktop with sample grammars in German. You can change the language to English and explore this environment. To use and develop grammars of your own, click on the Home icon below and create a new folder for your grammars and other data. **This folder disappears when you log off!**
5. Insert your flash memory stick into a USB port. Its icon should appear on the desktop. Now you can synchronize its content with that of your temporary Knoppix. **Don't forget to save the content of your Knoppix folder onto the flash memory before you log off!**
6. To run Trale with your grammar, open a terminal window and cd to the folder with the grammar, usually consisting of the two files signature and theory. Proceed as described below in § 3.

2.2 How to install Trale on a Linux system with Sicstus Prolog

This way, you'll use Trale compiled on your system (which runs faster for larger grammars and data). See `http://milca.sfs.uni-tuebingen.de/A4/Course/trale/` and the help files included with the distributions. Note that only the new Trale package handles UTF-8 characters. For instructions concerning the Qt4 package see below.

2.3 How to install Trale on a Linux system without Sicstus Prolog

1. Create a folder Gram1, download into it the files signature and theory from Moodle, LTGF, Topic 4, Trale: Grammar 1: `http://moodle.ff.cuni.cz/mod/resource/view.php?id=5608`.
2. Go to `http://hpsg.fu-berlin.de/Software/Trale/` and download the file standalone-trale.tbz.
3. Follow the instructions 1–3 on the Standalone Trale web page. Then:
4. `cd` to Gram1 and try this:
   ```
   trale -sag
   ```
   There can be two results:

   A: If you see a response similar to that below and a new window named GRISU, you are done. Skip the steps 5– below. You can now compile the grammar and parse a string — see below in § 3.

   ```
   Using Trale system found in /usr/local/standalone-trale
   Starting Trale (no saved state) from sources and the grisu interface
      Starting grisu interface .... started on 5002 of host localhost
      Starting standalone version of Trale (standalone-trale.Linux)
   Loading Interface connection specifications (options -g):

   Establishing connection to interface on port 5002 of localhost.
   Connection established.

   TRALE Milca environment (version 2.7.12)
   Copyright (C) 2002/3 Project MILCA A4
   PIs: Detmar Meurers (OSU), Gerald Penn (Univ. Toronto),
        Frank Richter (Univ. Tubingen)
   All rights reserved

   | ?-
   ```

   B: If you see
   ```
   Starting grisu interface .......................
   ```
   going on and on, then there is more work to do:

5. Go to `http://milca.sfs.uni-tuebingen.de/A4/Course/trale/` and download the file grale-v-1-0-2-src.tar.bz2. Unpack it.
6. Unless you have it installed already, install the development package Qt, version 4. The easiest way is to install the package using your Linux distribution procedure. For example, in Fedora you say this:
   ```
   yum install qt4-devel
   ```
   Alternatively, but with some risk of problems, you can download and install Qt4 from here:
   `http://wftp.tu-chemnitz.de/pub/Qt/qt/source/qt-x11-opensource-src-4.3.4.tar.gz`.
7. `cd` to the source folder of Grale (see Step 5) and do:
   ```
   /usr/lib/qt4/bin/qmake
   ```
   (or similar, depending on where your Qt4 resides)
   Then do:
   ```
   make
   ```
   If you encounter a problem, consult the file GRALE.HOWTO in the Grale source folder.
8. Create a symbolic link to the grale executable in a folder included in a search path of your system. For example:
   ```
   cd /usr/local/bin ln -s ../grale-v-1-0-2-src/grale
   ```
9. `cd` to the Trale folder (standalone-trale, see Step 3) and liberalize your access rights to the file trale, e.g. like this:
   ```
   chmod a+rwx trale
   ```
   Then open the file for editing, and modify the instruction which graphical interface should be used. Change
   ```
   # What to call to start the graphical interface
   FRONTEND=grisu
   ```
   into
   ```
   # What to call to start the graphical interface
   FRONTEND=grale
   ```
   Save.
10. Repeat Step 4. You should now see Result A.

3 How to run Trale

1. `cd` to a folder with the files signature and theory
2. Say:
   ```
   $ trale -sag
   ```
   or, if you have Sicstus Prolog and don't use the standalone version:
   ```
   $ trale -sg
   ```
3. You can list all paremeters for Trale with
   ```
   $ trale -h
   ```
   If you are comfortable with emacs, call Trale without the parameter `s`, or replace `s` by `x`. If you use UTF-8 encoding in your grammars, add `u` (not available with the old compiled version of Trale). You shouldn't omit `g` with the standalone version of Trale, the plain text output is not properly formatted.
4. `| ?- c.` — compile a grammar
5. `| ?- rec.` — parse a string as shown

   `|: pepa pase kozu` — type a string
6. A window with a parse tree should appear. You can click on a node in the tree to see the structure and on various parts of the structure to see more or less of it.

   Trale asks you if you want to see more trees by a prompt `ANOTHER?`. If can respond by pressing Enter or `Y` you'll see another parse. Of course, only if your grammar has determined that the string has more parses.

   Each parse is listed in the GRALE (GRISU) window. You can recall any previous parse by double clickling on the corresponding line.
7. `| ?- halt.` — to quit
8. More commands:

   `| ?- rec([pepa,pase,kozu]).` — parse a string

   `| ?- rule` Rulename`.` — show a rule Rulename

   `| ?- lex` Word`.` — show a lexical entry Word
9. Inspecting signature:

   `| ?- show_approp(`Type`).`

   `| ?- show_subtypes(`Type`).`

   `| ?- show_all_subtypes(`Type`).`

   `| ?- show_supertypes(`Type`).`

   `| ?- show_all_supertypes(`Type`).`
10. Parsing numbered examples:

    `| ?- test(`Nr`).`

    `| ?- test([`From, To`]).`

    `| ?- test(`all`).`

    `| ?- testt(...).` — same as above without producing structures