RUNNING λProlog

1 Finding λProlog

1.1 On Computer Science Department HP Machines

The interpreter is called `Terzo`. (The name means "fourth" in Italian, and this is the fourth implementation of λProlog.) (It lives in `/usr/unsup/bin`, which should be in your PATH by default.)

1.2 On Project Vincent

On Project Vincent, execute `add sml` to gain access to the SML directory `/home/sml` and have the executable and manual page available. (This is not a misprint, the interpreter is in the SML directory.) The interpreter is called `Trezo`.

2 Working with the Interpreter

The interpreter's prompt is `Terzo>` . Input is terminated with a period (.); you can continue typing on separate lines. If the interpreter prompts with several spaces and `>`, then it is expecting more input. As in prolog, the interpreter doesn't evaluate your input until you type the period (.).

To exit λProlog, use the `#quit` command. Don't forget the period (as usual). Typing control-D will also work if you are at the top level.

To interrupt the interpreter, type control-C.

To read in a file of definitions (a module) in a file "foo.mod", type `#load "foo.mod".` to the interpreter (don't forget the period). You can use `#load "foo.mod".` to reload a module you changed. (This erases the previous definitions and reads in the new ones.)

To give a query, use the syntax

```
#query <module-name> <term>.
```

where `<module-name>` is the name of some module (like `foo`), and `<term>` is a term in the λProlog grammar. You can also start an interactive query loop for a module by writing

```
#query <module-name>.
```

When you are done, type `#end.` to get back to the top level input loop.

When you give λProlog a query, it may respond with the message "solved" or some substitution will be printed out. If it is not prompting you with `?-`, it is waiting to see if you want more solutions to your query. You can either hit return to get another prompt, or type a semicolon (`;`) to see if there are more solutions.

3 Errors

An error such as

```
std_in:4.1-4.2Ignoring illegal character #
std_in:4.2-4.14: Error: Illegal declaration: not parsing module.
```

means that you typed in an illegal character. The error message points out where the error is; it's in the fourth line of your input, either character 1 or 2. Type your input again.

An error of the form

```
std_in:8.1-8.6: Error: Undeclared constant timse.
```

means either you made a spelling error (in this case using "decsribe" instead of "describe") or that the constant you want needs to be defined (in a module, and loaded) first.

If you have other syntax errors, sometimes adding parentheses helps. Using a LISP-like style, `(f x)` may sometimes help the parser.

To prevent waiting for occasional core dumps, run `Terzo` from a directory in which you have an empty file named `core` that you cannot write (Unix permission 0); or execute (for csh):

```
limit coredumpsize 0
```

(I put this csh command in my `.login` file.)

4 Where to Find More

After λProlog starts, type `#help.` (don't forget the period) for more information. This tells you what help topics are available. For example, you can type `#help load.` to find more about the `#load` command. The library of λProlog code available for loading is in `/usr/unsup/Terzo/lib/`. The grammar is in the directory `/usr/unsup/Terzo/doc/`. The macrosyntax is in the file `grammar`. The microsyntax is in the file `lexical`.

The web page for Terzo is

```
http://www.cse.psu.edu/~dale/lProlog/terzo/index.html
```

The web page for λProlog itself is

```
http://www.cse.psu.edu/~dale/lProlog/
```

5 Other Interpreters

An older interpreter for λProlog is `elp`. It also lives in `/usr/unsup/bin` on the Com S machines.

If `elp` puts the cursor 4 spaces to the right, then it is expecting more input. Try typing a period (.). To exit `elp`, use the `bye.` command. Typing control-D will also work if you are at the top level. To interrupt the interpreter, type control-C.

To read in a file of definitions (a module) in a file "foo.mod", type `use "foo".` You can use `load "foo".` to reload a module you changed. (Doing another `use` does not read the file again, but `load` erases the previous definitions and reads in the new ones.)

After `elp` starts, type `help.` for more information. This tells you what help is available. For example, you can type the query `describe load.` to find more about the `load` primitive. There are some examples in `/usr/unsup/lpsml/elp/examples`, and the library of λProlog code available for loading is in `/usr/unsup/lpsml/elp/lib`. The grammar is in the directory `/usr/unsup/lpsml/elp/grammar`. The macrosyntax is in the file `elp.grm`. The microsyntax is in the file `elp.lex`.