# Lisp in Summer Projects Submission

| | |
|---|---|
| **Submission Date** | 2013-10-01 17:38:26 |
| **Full Name** | Emanuele Acri |
| **Country** | Italy |
| **Project Name** | prolog-talk |
| **Type of software** | command-line/terminal app |
| **General category** | other |
| **LISP dialect** | Commmon Lisp |
| **GitHub URL** | https://github.com/crossbowerbt/prolog-talk |
| **Did you start this project?** | Yes, all the code is written by me |
| **Project Description** | I want to describe my project in this form. |
| **Purpose** | Implement a parser for natural English language and provide a free grammar dictionary. |
| **Function** | The project parses English sentences and outputs parse trees.<br><br>The user write a sentence in English and receive a tree representation, which contains grammatical and syntactical informations.<br><br>A grammar dictionary is also provided, 110000 word (not counting plurals and conjugations of verbs), for AI programming. |
| **Motivation** | Few dictionaries are available without royalties. The project provides a decent one.<br><br>In future more programs will accept natural language as input, and the project provides an easily scriptable parser for it.<br><br>Personal objectives: |

| | |
|---|---|
| | Improve my knowledge of linguistics, and apply my studies in a concrete program.<br><br>The concept of macro-generated prolog was intriguing. |
| **Audience** | The dictionary is for programmers of AI applications. It will remain free to use. The format is easily adaptable even for different programming languages.<br><br>The parser is a base to build more complex tools, some of which are described in the Conclusion section. |
| **Methodology** | The main program is based on a dictionary that contains word categories (a word can be in more that one category) and forms (i.e. plural for nouns, conjugations for verbs).<br><br>Gambol provides an implementation of prolog in common lisp, where facts and rules can call lisp functions and vice-versa.<br><br>Using these resources, a hierarchal network of finite state automata (FSA) parse English sentences into syntactically correct syntagms.<br><br>"Hierarchal" since they can be nested into each other. This makes the structure roughly equivalent to context free grammars.<br><br>"Syntactically correct" because there is no semantic knowledge in the system. Semantic ambiguity is possible.<br><br>---<br><br>Implementation:<br><br>- A simple language describes the automata. It's implemented using a macro that generates "lispy" prolog rules. I choose prolog because automatic backtracking is suited for the task.<br><br>- FSA used are not basic ones: I added jumps, to simplify networks, and a system similar to "difference lists" to manage sentence gaps.<br><br>On this last point, let me clarify with an example:<br><br>"The fox which _ jumped onto the log is red."<br><br>The *dependent* sentence has no subject: there is gap.<br><br>Context free grammar are not able to manage gaps very well.<br>FSA used by the program are augmented with some instructions (the :if, :if-not, :set, :unset modifiers), to manage this.<br><br>To understand the program a little knowledge of how prolog works is required.<br><br>---<br><br>The main challenge was building the dictionary: 110000 words, not counting different forms, from a 1913 text. |

Programming a satisfying automata network proved a little challenging, like implementing a (simple) way to manage gaps.

**Conclusion**

Accomplishments:

- A decent English dictionary is now available, usable in a lot of situations.

- The parser works for a useful range of sentences, and it is good enough to build friendly user interfaces.

Limitations:

- The system is slow, and uses a too much memory:

Gambol is not optimized for large database, and do not support tail-call optimizations (the main part of the program is recursive).

It also use continuations to back-track. A lot of unnecessary state information is saved in memory.
(a possible solution is to write a simple prolog in lisp, with the objective of being directly optimizable by the compiler.)

- The parser can be extended to recognize more types of sentences, possibly even paragraphs.

- Adding Semantic and pragmatic informations can reduce ambiguity.

Future directions:

I plan to work on the following projects:

- A non literary translation tool: a based on the meaning of sentences, not on the single words.

- A Siri-like interface, that match user input, using specific features of the parse trees (thus permitting a flexible way for the user to give commands to the machine).

**Build Instructions**

The SBCL common lisp implementation is recommended.

The only dependency is gambol. It can be installed using quicklisp, or the old, but still working, asdf-install:

```
(require 'asdf)
(require 'asdf-install)
(asdf-install:install "gambol")
```

**Execution Instructions**

Inside the project directory:
sbcl --script ./prolog-talk.lisp

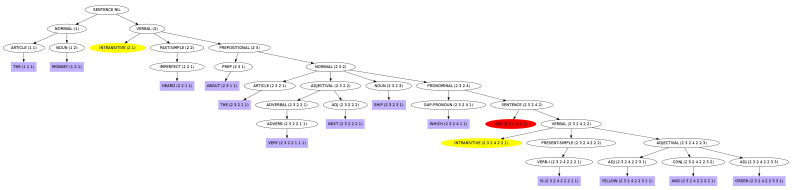
Then write your sentences when the prompt appears.

A detailed usage example is on the project site.

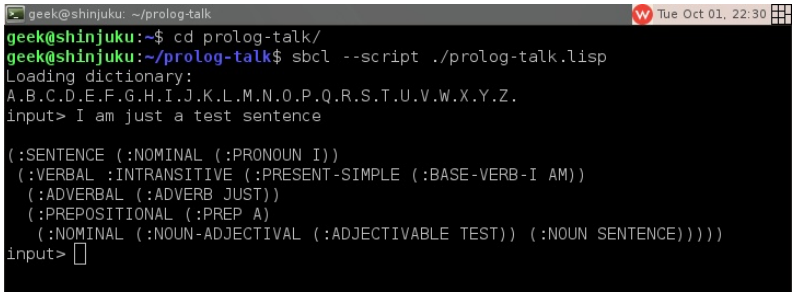
**Describe any bugs or caveats**

Do not try the "multiple trees" option for complex sentences, unless you have a *lot* of memory available.

**Screen shots**



[sentence.png]




[usage.png]

**Official**

I have read rules and have abided by them.
I am 18 years of age or older.
I am not living in Brazil, Quebec, Saudi Arabia, Cuba, Iran, Myanmar (Burma), North Korea, Sudan, or Syria.